FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Backend systému pro ukládání bezpečnostních dat
Student:	Filip Pokorný
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je serverová aplikace pro efektivní ukládání a dotazování bezpečnostních dat (Cyber Threat Intelligence). Samotné získávání bezpečnostních dat je mimo rozsah této práce.

Postupujte v těchto krocích:
Proveďte analýzu požadavků na funkcionalitu aplikace.
Na základě analýzy proveďte důkladný návrh.
Dle návrhu implementujte vhodný backend.
Pro vámi implementovaný kód připravte testování alespoň v ukázkovém rozsahu.
Navrhněte budoucí možnosti rozvoje vašeho řešení.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 23. října 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Backend systému pro ukládání bezpečnostních dat

Bc. Filip Pokorný

Katedra informační bezpečnosti
Vedoucí práce: Ing. Jiří Hunka

10. května 2022

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

Praha dne 10. května 2022 . . . . . .

Odkaz na tuto práci

Pokorný, Filip. *Backend systému pro ukládání bezpečnostních dat.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Tato bakalářská práce se zabývá vývojem nového backendového systému pro ukládání a vyhledávání bezpečnostních dat (Cyber Threat Intelligence). Obsahem je analýza současného řešení pro ukládání a vyhledávání v bezpečnostních datech získávaných z nástroje IntelMQ a služby Shodan. Na základě této analýzy jsou sestaveny požadavky na nový systém. Podle požadavků je proveden návrh nového systému a jeho následná implementace. Závěrečná část obsahuje dokumentaci nově vytvořeného systému a náměty pro budoucí rozvoj.

Klíčová slova backend, bezpečnostní data, threat intelligence, vyhledávání, ukládání, IntelMQ, Shodan, Threatstate, Python, FastAPI, Beanie, Pydantic, pyparsing, MongoDB, Docker

Abstract

This bachelor thesis deals with development of new backend system for storing and searching security data (Cyber Threat Intelligence). It contains an analysis of currently used solution for storing and searching security data received from IntelMQ system and Shodan service. Based on the analysis a list of requirements for the new system is compiled. The requirements are used for designing the new system and it's implementation. The final part contains the documentation of the newly developed system and suggestions for future development.

Keywords backend, security data, threat intelligence, search, storage, IntelMQ, Shodan, Threatstate, Python, FastAPI, Beanie, Pydantic, pyparsing, MongoDB, Docker

Obsah

Seznam obrázků

Seznam výpisů kódu

Úvod

S narůstajícím zájmem o problematiku počítačové bezpečnosti také přibývají různé projekty a iniciativy, které produkují zajímavé informace o bezpečnosti počítačů připojených k Internetu (tzv. Cyber Threat Intelligence). Povaha takových informací je různá v závislosti na konkrétním projektu - některé projekty se například snaží zmapovat všechny běžící služby přístupné z Internetu. Jiný projekt se může zaměřit pouze na pár vybraných služeb. Další projekt se naopak soustředí na škodlivý kód a mapuje takové počítače, které jsou nakažené a připojují se k velícímu Command and Control serveru. Odlišný způsob zvolí například projekt provozující fiktivní a zdánlivě zranitelné cíle pro útočníky (honeypoty), které umožní analyzovat útočníkovo chování. Výstupem takových a jím podobných projektů jsou často velmi zajímavá data, která jsou v mnoha případech zdarma zveřejněna (a pravidelně aktualizována) a která má smysl sbírat pro další analýzu. Pokud bychom data z každého takového projektu brali jako jeden díl skládačky, tak jejich hromadných sběrem a skládáním dohromady můžeme vytvořit ucelenou znalost o každém počítači připojeném k Internetu. Ta pak může posloužit jak práci bezpečnostních analytiků, tak také koncovým správcům sítí, kteří získají vnější vhled do své sítě a mohou díky tomu odhalit bezpečnostní rizika či jiné nedostatky.

V rámci mého zaměstnání takový systém pro hromadný sběr dat z různých zdrojů provozujeme a rozvíjíme již od roku 2017. S postupem času a rošiřováním zdrojů, ze kterých data sbíráme, však začala použitelnost systému narážet na nedostatky ve způsobu ukládání a sjednocení formátu dat, které nelze snadno odstranit. Tyto nedostatky se promítly i do možnosti v datech jednoduše vyhledávat, což ztížilo práci analytiků. Takto vznikla myšlenka vytvoření zcela nového systému pro efektivnější ukládání a vyhledávání sbíraných dat.

Cílem této práce bude vytvořit backend, jehož součástí bude nový způsob ukládání a vyhledávání sbíraných dat, který umožní vytvoření ucelené znalosti o počítačích připojených k Internetu a to na základě vstupních dat získaných sběrem z různých bezpečnostních projektů. Samotný sběr vstupních

dat je mimo rozsah této práce. Nejprve se v této práci budu zabývat analýzou současného stavu a jeho nedostatků, podle kterých sestavím požadavky na nový systém a provedu srovnání konkurenčních řešení. Na základě této analýzy vytvořím návrh nového systému, který poslouží jako podklad pro následnou implementaci. V závěrečné části práce se budu věnovat dokumentaci používání nového systému a námětům pro navazující vývoj.

Pracovní název mnou vyvíjeného systému je Threatstate.

Kapitola 1

Analýza

V první části této kapitoly se věnuji popisu současného stavu a identifikaci jeho nedostatků. Na základě těchto zjištění pak v další části této kapitoly formuluji ucelený seznam funkčních a nefunkčních požadavků, které budou stěžejní pro návrh nového systému. V poslední části kapitoly se věnuji porovnání s vybranými konkurenčními řešeními.

1.1 Současné řešení

Současné řešení je sestaveno z několika součástí. První z nich je nástroj IntelMQ, který slouží pro samotný sběr bezpečnostních dat a jejich obohacení. Z nástroje IntelMQ jsou data převzata nástrojem Logstash, který umožňuje dodatečné úpravy dat a zajišťuje jejich vložení do databázového systému Elasticsearch. Pro vizualizaci a vyhledávání v datech je použit nástroj Kibana. Podrobnější popis jednotlivých součástí se nachází v následujících odstavcích. Tento systém slouží zejména pro práci bezpečnostních analytiků, kteří díky němu mohou snáze identifikovat bezpečnostní rizika a hrozby v prostředí Internetu.

IntelMQ Jedná se o open-source nástroj, který je používán a kolektivně vyvíjen zejména evropskými CERT/CSIRT týmy.[1] Slouží k automatizovanému sběru bezpečnostních dat z různých zdrojů (web, mail, rsync, API třetích stran aj.) a v různých formátech (csv, json, formáty třetích stran aj.), získaná data transformuje do vlastního jednotného formátu (tzv. Event) a následně tato data umožňuje obohacovat (geolokace IP adres, detekce TOR exit nodů, doplnění WHOIS informací aj.). Přednostmi nástroje IntelMQ jsou schopnost sbírat data z široké škály zdrojů dat třetích stran a jeho vysoká přizpůsobitelnost, díky které je možné vytvoření velmi individuální konfigurace sbírání zdrojových dat a jejich způsobu zpracování a obohacování.

Elasticsearch, Logstash, Kibana Někdy také hromadně označované jako ELK Stack je kolekce open-source produktů od společnosti Elasticsearch B.V.,

které však obsahují také placené funkcionality. Stěžejním z nich je databázový systém Elasticsearch. Jedná se o distribuovaný databázový systém s vyhledávacími a analytickými nástroji.[2] Logstash slouží pro sběr a transformaci dat z různých zdrojů a jejich následnému předání k dalšímu zpracování či uložení (v našem současném řešení Logstash vkládá data do Elasticsearch databáze). Posledním je nástroj Kibana, který umožňuje uživatelsky pohodlné vyhledávání a vizualizaci dat uložených v Elasticsearch databázi a její správu skrze grafické webové prostředí.[3]

1.1.1 Nedostatky současného řešení

Současné řešení je s drobnými úpravami používáno od roku 2017. V průběhu času docházelo k rozšiřování počtu zdrojů dat. Některé z nich však poskytují data takového formátu či povahy, která pro naše účely nelze sbírat a zpracovávat nástrojem IntelMQ, protože je není možné bez významných ztrát převést do jednotného formátu používaného nástrojem IntelMQ. Jedním takovým významným zdrojem dat je komerční služba Shodan, která pravidelně skenuje všechny stroje připojené k Internetu a zjišťuje přístupné služby. Tento stav vedl k vytvoření dodatečných skriptů a použití dalších drobných nástrojů, které umožňují data uložit do Elasticsearch databáze, bohužel však odděleně od dat zpracovaných nástrojem IntelMQ. Díky nástroji Kibana je sice v datech stále možné vyhledávat a vytvářet vizualizace, ovšem práci bezpečnostního analytika to značně ztěžuje, protože musí vyhledávat v různých Elasticsearch indexech a ucelenou informaci například o konkrétní IP adrese sestavit ručně.

Dalším nedostatkem je absence logiky, která by řídila ukádání dat do databáze. Současné řešení ukládá data do databáze jako proud záznamů bez jakékoliv reflexe již existujících záznamů. Nástroj IntelMQ způsob dlouhodobého ukládání zpracovávaných dat nad rámec tohoto přímočarého přístupu neimplementuje. Jedná se o velmi jednoduchý způsob ukládání, který je však nešetrný k práci se systémovými zdroji a značně ztěžuje práci analytika, který tak musí procházet velké množství záznamů a opět manuálně sestavovat ucelenou informaci.

Třetím nedostatkem současného řešení je způsob zpracování dat nástrojem IntelMQ, který sbíraná data zpracovává proudově a reprezentuje je jako jednotlivé nesouvisející události. Pokud by například zpracovával seznam velkého množství pokusů o přihlášení ze stroje X k službě SSH na stroji Y, nedokázal by vyhodnotit, že se jedná o útok hrubou silou.

1.1.2 Účastníci

Pojem účastník popisuje skupinu uživatelů, která má v systému určenou sadu oprávnění umožňující jejich interakci se systémem. Současně používané řešení rozlišuje celkem tři uživatelské role: správce systému, bezpečnostní analytik a systémový uživatel.

1.1.2.1 Správce systému

Správcem systému je osoba, která je zodpovědná za provoz systému. Správci je umožněna interakce se systémem ve všech podobách, jakými ho mohou používat i ostatní role. Navíc je mu umožňěno spravovat evidenci uživatelů.

1.1.2.2 Bezpečnostní analytik

Tato role slouží pro osoby, které mohou v systému zadávat dotazy pro vyhledání uložených bezpečnostních dat.

1.1.2.3 Systémový uživatel

Role systémového uživatele je určena pro umožnění přístupu externímu systému pro vkládání bezpečnostních dat k uložení.

1.1.3 Evidence uživatelů

Tato část popisuje funkce související s evidencí uživatelů.

1.1.3.1 UC1 Zobrazit seznam uživatelů

Osoby v rolích správce systému mohou zobrazit seznam uživatelů evidovaných v systému. U každého zobrazeného uživatele tak zjistí jeho uživatelské jméno a úroveň oprávnění v systému.

1.1.3.2 UC2 Vytvořit nového uživatele

Osoby v rolích správce systému mohou vytvořit nového uživatele systému. Ten bude zařazen do evidence uživatelů. Nově vytvořenému uživateli je při vytvoření přiděleno uživatelské jméno, heslo a úroveň oprávnění.

1.1.3.3 UC3 Upravit existujícího uživatele

Osoby v rolích správce systému mohou upravit existujícího uživatele evidovaného v systému. Úprava existujícího uživatele umožňuje změnu uživatelského hesla a nebo jeho úrovně oprávnění v systému.

1.1.3.4 UC4 Odstranit existujícího uživatele

Osoby v rolích správce systému mohou odstranit existujícího uživatele evidovaného v systému.

1.1.4 Evidence záznamů získaných z nástroje IntelMQ

Tato část popisuje funkce související s evidencí jednotlivých záznamů získaných z nástroje IntelMQ.

1.1.4.1 UC5 Vytvořit nový záznam

Osoby v rolích správce systému a nebo systémového uživatele mohou v evidenci vytvořit nový záznam. Nejsou vynucována žádná pravidla, která by určovala podobu těchto záznamů. Očekává se však, že budou ve formátu používáném nástrojem IntelMQ[4].

1.1.4.2 UC6 Vyhledat záznamy

Osoby v rolích správce systému a nebo bezpečnostního analytika mohou v evidenci vyhledávat záznamy podle libovolných atributů.

1.1.4.3 UC7 Úpravit záznam

Osoby v rolích správce systému mohou upravit existující záznamy v evidenci.

1.1.4.4 UC8 Odstranit záznam

Osoby v rolích správce systému mohou odstranit existující záznamy z evidence.

1.1.5 Evidence záznamů získaných ze služby Shodan

Tato část popisuje funkce související s evidencí jednotlivých záznamů získaných ze služby Shodan.

1.1.5.1 UC9 Vytvořit nový záznam

Osoby v rolích správce systému a nebo systémového uživatele mohou v evidenci vytvořit nový záznam. Nejsou vynucována žádná pravidla, která by určovala podobu těchto záznamů. Očekává se však, že budou ve formátu používáném službou Shodan[5].

1.1.5.2 UC10 Vyhledat záznamy

Osoby v rolích správce systému a nebo bezpečnostního analytika mohou v evidenci vyhledávat záznamy podle libovolných atributů.

1.1.5.3 UC11 Úpravit záznam

Osoby v rolích správce systému mohou upravit existující záznamy v evidenci.

1.1.5.4 UC12 Odstranit záznam

Osoby v rolích správce systému mohou odstranit existující záznamy z evidence.

1.1.6 Shrnutí

V analýze současného řešení je popsáno jeho technické řešení, nedostatky a jeho stežejní funkcionality, které svým uživatelům nabízí. Z těch je zřejmý zmíněný nedostatek, vyplývající z oddělených evidencí podle původu dat, který nutí bezpečnostní analytiky vyhledávat informace ve dvou různých evidencích.

Současné řešení obsahuje i další funkcionality, které vyplývají z obecných možností současně používaných nástrojů, ty však nejsou předmětem této práce a proto nebyly v analýze zahrnuty. Příkladem takových funkcionalit jsou například funkcionality spojené se správou databázového systému Elasticsearch.

1.2 Formulace požadavků

V této podkapitole jsou formulovány požadavky na nový systém, které by měl splňovat. Jsou sestaveny na základě popsaných nedostatků a autorových zkušeností se současným řešením a to jak z pohledu uživatele, tak i z pohledu administrátora. Dále zohledňují používáné prostředí pro provoz systémů v rámci zaměstnání autora této práce, kde bude systém provozován. Požadavky jsou rozděleny na funkční požadavky, které jasně specifikují jakou funkčnost bude nový systém uživatelům nabízet, a na nefunkční požadavky, které vymezují obecné charakteristiky systému. Nefunkční požadavky jsou dále rozděleny do kategorií dle akronymu FURPS, který rozlišuje funkční požadavky a čtyři další kategorie (použitelnost, spolehlivost, výkon a podporovatelnost či rozšiřitelnost).[6]

1.2.1 Funkční požadavky

1.2.1.1 FP1 Evidence uživatelů

Systém umožní přidávat, upravovat a odebírat uživatele, kteří mohou k systému přistupovat. U uživatelů se bude evidovat uživatelské jméno, heslo a role v systému (úroveň oprávnění). Úrovně oprávnění budou rozlišovat minimálně práva na zápis (vkládání dat), práva pro čtení (vyhledávání dat) a práva na správu evidence uživatelů.

- Priorita: nízká
- Složitost: střední

1.2.1.2 FP2 Evidence strojů připojených k Internetu

Systém eviduje informace o strojích připojených k Internetu a to včetně jejich historického stavu. Jedná se zejména o informace jako IP adresa, běžící služby přístupné z Internetu, geografické umístění, identifikované zranitelnosti,

operační systém, doménové jméno a další. Informace jsou evidovány jednotně bez ohledu na původ dat. Historický stav bude udržován s přesností na dny.

- Priorita: vysoká
- Složitost: vysoká

1.2.1.3 FP3 Evidence událostí týkajících se strojů připojených k Internetu

Systém eviduje události, týkající se strojů připojených k Internetu. Každá událost obsahuje informaci o konkrétním typu události (např. útok hrubou silou), čas události, zdrojovou IP adresu, cíl a případné specifické informace podle konkrétního typu události. Události jsou evidovány jednotně bez ohledu na původ dat.

- Priorita: vysoká
- Složitost: střední

1.2.1.4 FP4 Vyhledávání stavu strojů připojených k Internetu

Systém umožní uživatelům s odpovídajícím oprávněním vyhledání informací o stavu strojů připojených k Internetu a umožní jim přístup ke všem evidovaným informacím včetně historického stavu.

- Priorita: vysoká
- Složitost: vysoká

1.2.1.5 FP5 Vyhledávání událostí týkajících se strojů připojených k Internetu

Systém umožní uživatelům s odpovídajícím oprávněním vyhledání informací o událostech týkajících se strojů připojených k Internetu a umožní jim přístup ke všem evidovaným informacím.

- Priorita: vysoká
- Složitost: vysoká

1.2.1.6 FP6 Vložení a zpracování dat ve formátu používaném službou Shodan

Systém umožní uživatelům s odpovídajícím oprávněním vložení dat ve formátu používaném službou Shodan. Vložená data budou systémem zpracována a uložena podle jejich charakteru do odpovídající evidence.

- Priorita: vysoká
- Složitost: vysoká

1.2.1.7 FP6 Vložení a zpracování dat ve formátu používaném nástrojem IntelMQ

Systém umožní uživatelům s odpovídajícím oprávněním vložení dat ve formátu používaném nástrojem IntelMQ. Vložená data budou systémem zpracována a uložena podle jejich charakteru do odpovídající evidence.

- Priorita: vysoká
- Složitost: vysoká

1.2.2 Nefunkční požadavky

1.2.2.1 NP1 Přístup skrze REST API

Součástí systému bude REST API, které umožní se systémem komunikovat. Toto API může být také v budoucnu využito například pro vývoj webového grafického uživatelského rozhraní či textového uživatelského rozhraní (pro použití z příkazové řádky) a nebo pro integraci systému s jinými nástroji.

- Kategorie: použitelnost
- Priorita: vysoká
- Složitost: střední

1.2.2.2 NP2 Vývoj v jazyce Python

Předpokládá se, že použití systému bude v praxi úzce spjato také s použitím nástroje IntelMQ. Požadavek na použití programovacího jazyka Python má opodstatnění zejména kvůli budoucí spolupráci s vývojáři nástroje IntelMQ, na kterém se autor této práce také částečně podílí a který je celý implementován právě v jazyce Python. Toto zmenší bariéru toho, aby se přispěvatelé do open-source projektu IntelMQ mohli v budoucnu podílet také na dalším rozšiřování systému.

- Kategorie: podporovatelnost
- Priorita: vysoká
- Složitost: nízká

1.2.2.3 NP3 Podpora pro běh v operačním systému GNU/Linux

Systém musí být uzpůsoben běhu v operačním systému GNU/Linux. Tento požadavek vychází z používáného prostředí pro provoz systémů v rámci zaměstnání autora této práce.

- Kategorie: podporovatelnost
- Priorita: střední
- Složitost: nízká

1.2.2.4 NP4 Provoz v prostředí Dockeru

Způsob provozu systému je nedílnou součástí této práce. Požadavek na provoz v prostředí Docker vychází z používaného prostředí pro provoz systémů v rámci zaměstnání autora této práce. Je tedy požadováno, aby byl systém připravený pro nasazení do Dockerového prostředí.

- Kategorie: podporovatelnost
- Priorita: vysoká
- Složitost: nízká

1.2.2.5 NP5 Rozšiřitelnost

Mezi funkčními požadavky je schopnost zpracovávat data z nástroje IntelMQ a služby Shodan. Pro budoucí vývoj je však velmi pravděpodobné, že se tento seznam zdrojů dat dále rozšíří. Pří návrhu je tedy nutné zohlednit budoucí implementaci zpracování dalších zdrojů dat, která by měla být bezproblémově integrovatelná do již existujícího kódu.

- Kategorie: rošiřitelnost
- Priorita: střední
- Složitost: nízká

1.2.2.6 NP6 Instalace

Pro přípravu prostředí, ve kterém bude systém spuštěn, bude připraven nástroj, který přípravu provede. Příprava prostředí bude závislá na podporovaném typu operačního systému.

- Kategorie: podporovatelnost
- Priorita: nízká

1.2.2.7 NP7 Logování

Systém bude vytvářet záznamy o stavu a chybách za běhu programu.

- Kategorie: podporovatelnost
- Priorita: střední
- Složitost: vysoká

1.2.2.8 NP8 Konfigurovatelnost

Nastavení pro běh systému bude možné měnit pomocí konfiguračního souboru nebo jiným způsobem.

- Kategorie: podporovatelnost
- Priorita: střední
- Složitost: střední

1.2.2.9 NP9 Použití open-source technologií

V rámci vývoje systému musí být použity pouze takové technologie, které mají otevřený zdrojový kód a jejichž použití není zpoplatněno. Tento požadavek je vznesen v zájmu budoucích uživatelů systému a vychází také z filozofie používáných systémů v rámci zaměstnání autora této práce.

- Kategorie: podporovatelnost
- Priorita: vysoká
- Složitost: nízká

1.3 Srovnání konkurečních řešení

Existují řešení, která mají určitý překryv s touto prací, neobjevil jsem však žádné, které by se zaměřovalo na úplně stejnou problematiku a splňovalo formulované požadavky. Náměty na podobná řešení jsem sbíral na základě vlastní zkušenosti a informací od spřátelených organizací, které se sběrem bezpečnostních dat také zabývají. Pokud bych se zaměřil na open-source řešení, určitě stojí za zmínku například Malware Information Sharing Platform (MISP)[7] nebo Your Everyday Threat Intelligence (YETI)[8]. V obou případech se jedná o platformy sloužící k vytvoření uceleného přehledu bezpečnostních informací, avšak zaměřují se především na poznatky o malwaru (a s ním spojených tzv. Indicators of Compromise) a jejich použití pro mnou popsaný problém tak není ideální. Další za zmínku stojí komerční služby Censys[9] nebo Shodan[10]. Jejich řešení nabízí funkcionalitu téměř odpovídající

zadaným požadavkům. Data ze služby Shodan dokonce budou použita jako jeden ze zdrojů dat pro mé řešení. Přesto je několik zásadních rozdílů mezi těmito službami a požadovaným řešením: obě jsou komerční, obě mají uzavřený zdrojový kód a jsou poskytovány pouze jako webová služba. Nelze je tedy použít na vlastní infrastruktuře a ani rozšířit či přizpůsobit vlastním potřebám.

Kapitola 2

Návrh

2.1 Zvolené technologie

Tato část se zabývá popisem zvolených technologií. Výběr technologií vychází ze zadaných požadavků, znalostí a zkušeností autora a prostředí, ve kterém bude systém nasazen.

2.1.1 Python

Programovací jazyk Python je použit na základě požadavku NP2. Jedná se o programovací jazyk, který se podle průzkumu portálu StackOverflow v roce 2021 umístil na předních místech v žebříčku nejpopulárnějších programovacích jazyků.[11] Mezi obory, ve kterých nachází uplatnění, jednoznačně patří také použití ve webových aplikací, pro jejichž vývoj existuje mnoho používaných frameworků, z nichž se některé (Flask, Django a FastAPI) umístily ve zmíněném průzkumu v žebříčku nejpoužívanějších webových frameworků.[12] V neposlední řadě se jedná o programovací jazyk, se kterým má autor této práce předchozí zkušenost.

2.1.1.1 Knihovna FastAPI

FastAPI je výkonný webový aplikační framework, který využívá moderní Python a knihovnu Pydantic, díky kterým je možné deklarovat a validovat datové typy. FastAPI byl vyvinut s ohledem na obvykle používaná vývojová prostředí pro programování v jazyce Python, ve kterých je tak úplná podpora automatického doplňování kódu, což značně usnadňuje vývoj.[13]

Je postaven na na ASGI (Asynchronous Server Gateway Interface) frameworku Starlette, který propojuje Pythoní webové servery s asynchronně běžícími Python aplikacemi. Díky tomu je FastAPI a patří mezi ty nejrychlejší Python webové aplikační frameworky.

Další velmi užitečnou vlastností je automatická generace dokumenace dle specifikace OpenAPI.[14]

2.1.1.2 Knihovna Beanie

Beanie je poměrně mladá knihovna, která slouží jako tzv. Object-Document Mapper tedy zprostředkovává mapování a ukládání Python objektů do dokumentů databázového systému MongoDB.[15] Oproti tradičním a déle vyvíjeným knihovnám přináší podporu asynchronního Python kódu, což je v kombinaci s použitím knihovny FastAPI velmi užitečná vlastnost.

2.1.1.3 Knihovna pyparsing

Pyparsing je knihovnou nabízející alternativní přístup k tvorbě a vyhodnocování jednoduchých gramatik oproti přístupu tradičních nástrojů *lex* a *yacc* či použití regulárních výrazů. Poskytuje sadu Python tříd pro vytvoření gramatiky přímo v kódu a nevyžaduje tak znalost speciální syntaxe pro popis gramatik.[16]

2.1.2 MongoDB

NoSQL databázový systém MongoDB ukládá data ve formátu podobném JSONu do tzv. dokumentů a nevyžaduje striktně definované schéma.[17] Splňuje požadavek NP10 a existuje pro něj kvalitní integrace pro použití v programovacím jazyce Python, v tomto konkrétním případě zmíněná knihovna Beanie.

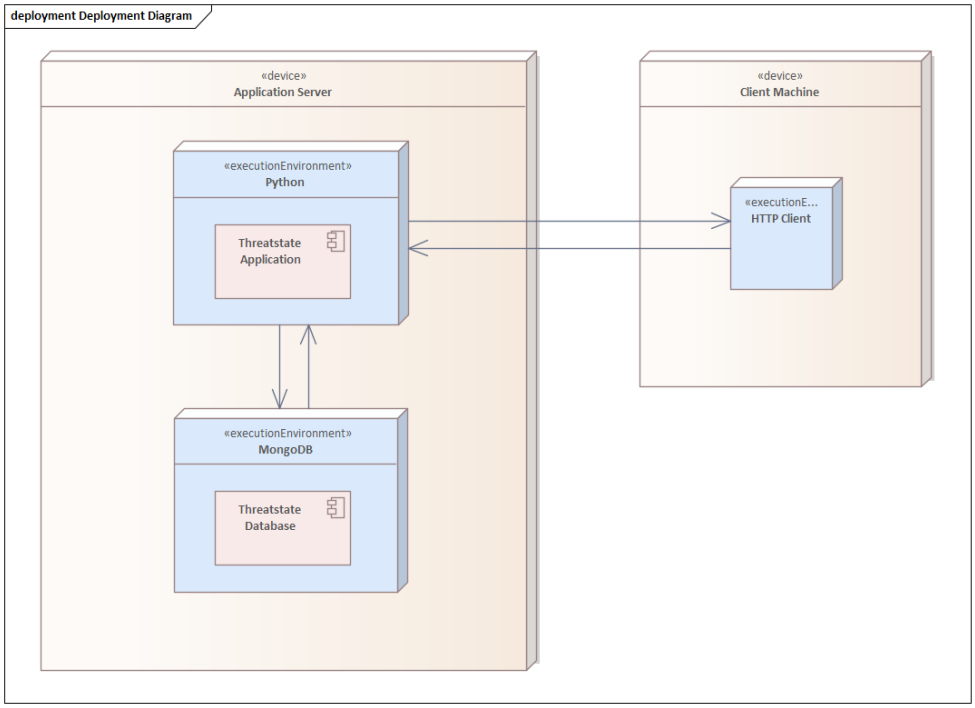
2.1.3 Docker

Jedná se o rozšířenou moderní technologii pro vývoj, distribuci a provoz aplikací. Umožňuje snadné oddělení aplikačního prostředí od infrastruktury, ve které jsou aplikace provozovány, a urychluje tak nasazení do produkce. Aplikace včetně jejich prostředí a závislostí jsou v systému Docker zabaleny a spouštěny v izolovaném prostředí, kterému se říká kontejner.[18] Součástí této práce tedy bude také sada konfigurací, které umožní snadné sestavení a spuštění systému Threatstate a jeho závislostí v podobě Docker kontejnerů.

2.2 Architektura

2.2.1 Fyzická architektura

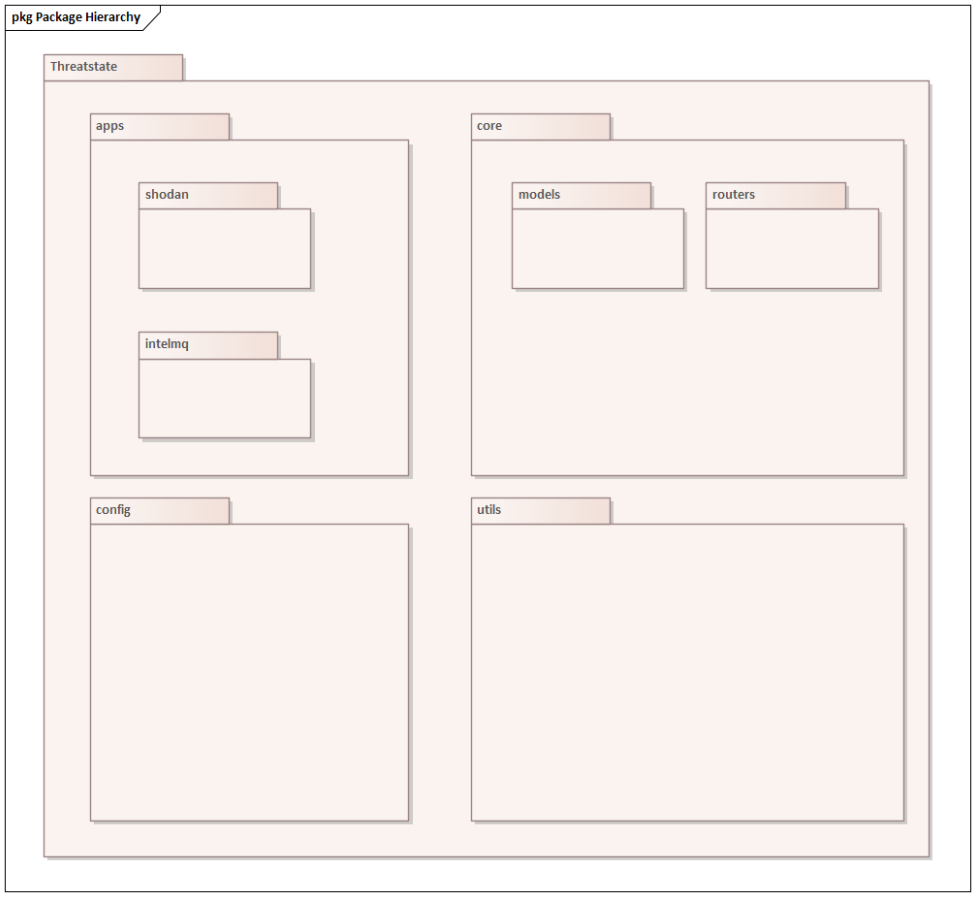
Návrh architektury systému Threatstate není složitý. Skládá se ze dvou komponent. První komponentou je aplikace implementovaná v programovacím jazyce Python, která se stará o veškerou logiku. Druhou komponentou je pak databázový systém MongoDB, který slouží jako trvalé uložiště dat zpracovaných aplikací v první komponentě.



Obrázek 2.1: Znázornění fyzické architektury pomocí diagramu pro nasazení

2.2.2 Logická architektura

Návrh logické architektury je v podobě jednoho Python modulu, který obsahuje několik dalších submodelů. Stěžejním bude submodul *core*, který bude obsahovat všechny implementované funkcionality. Dalším je submodul *config*, který obsahuje kód týkající se celkové konfigurace systému. Submodul *utils* bude obsahovat pomocný kód různorodého charakteru, pro použití v jiných částech aplikace. Posledním je pak submodul *apps*, který obsahuje kód týkající se zpracování různých vstupních dat a je koncipován tak, aby byl v budoucnu rozšiřitelný o vstupy z další zdrojů.



Obrázek 2.2: Znázornění logické architektury pomocí diagramu balíčků

2.3 API

Tato podkapitola popisuje vlastnosti navrženého REST API a způsob komunikace s jednotlivými koncovými body, které bude používáno pro komunikaci se systémem Threatstate. Veškerá komunikace probíhá zasíláním dat ve formátu JSON.

2.3.1 Bezpečnost

2.3.1.1 Autentizace

Bezpečnost ja zajištěna použitím základní HTTP autentizace. Uživatelé jsou tak nuceni s každým požadavkem zaslat přihlašovací údaje. Uživatelská hesla jsou uložena v databázi a hashována algoritmem Argon2id podle doporučení organizace Open Web Application Security Project (OWASP).[19] Nedílnou

součástí zabezpečení přihlašovacích údajů je však také jejich bezpečný přenos. Ten je možné zajistit například TLS šifrovaním HTTP provozu.

2.3.1.2 Autorizace

Další součástí zabezpečení jsou různé úrovně oprávnění uživatelů. Uživatelé mohou nabýt tří různých oprávnění a to ke čtení (umožní vyhledávání), zápisu (umožní vkládat data) a správě uživatelů (umožní přídávat či odebírat uživatele a nastavovat jejich heslo či úroveň oprávnění). Ve výchozím stavu systém bude obsahovat jednoho superuživatele. Ten disponuje všemi třemi oprávněními a je ho tedy nutné použít pro případné vytvoření dalších uživatelů.

2.3.1.3 Ošetření vstupů

Další důležitou složkou zabezpečení je ošetření uživatelských vstupů na chybná data (např. uživatelem omylem zadaný překlep, ale i útočníkem úmyslně připravená chybná data). Systém všechny uživatelské vstupy bude ošetřovat striktně definovanou podobou vstupů. Nesplní-li uživatelský vstup tuto podobu, není vstup dále zpracováván a je vrácena standartní odpověď s HTTP stavovým kódem 422.

2.3.2 Standartní odpovědi

Každý z koncových bodů reaguje na požadavky odpovědí s odpovídajícím HTTP stavovým kódem. Pokud koncový bod nespecifikuje vlastní odpoveď (obsahující například požadovaná data), použije se standartní odpověď obsahující klíč *detail*, jehož hodnotou je řetězec s libovolným slovním vysvětlením. V případě úspěchu toto pole vždy obsahuje řetězec OK.

Speciálním případem standartní odpovědi je odpověď s HTTP stavovým kódem 422 (dle RFC4918 tzv. Unprocessable Entity[20]), která signalizuje chybná vstupní data. Ta obsahuje také klíč *detail* jehož hodnotou je však pole s dodatečnými informacemi, popisujícími která část vstupu byla chybná.

Výpis kódu 2.1: Ukázková standartní odpověď s HTTP stavovým kódem 200

```
{
  "detail": "OK"
}
```

Výpis kódu 2.2: Ukázková standartní odpověď s HTTP stavovým kódem 403

```
{
  "detail": "Permission denied"
}
```

Výpis kódu 2.3: Ukázková standartní odpověď s HTTP stavovým kódem 422

```
{
  "detail": [
```

```
    {
      "loc": [
        "body",
        "source.ip"
      ],
      "msg": "field required",
      "type": "value_error.missing"
    }
  ]
}
```

2.3.3 Správa uživatelů

Následující podkapitoly popisují koncové body sloužící ke správě uživatelů a řízení jejich přístupu formou nastavování oprávnění.

2.3.3.1 GET /users

Koncový bod, který umožní uživatelům s oprávněním spravovat uživatele získat seznam všech existující uživatelů v systému. Tento koncový bod nemá žádné volitelné parametry.

Výpis kódu 2.4: Ukázková odpověď koncového bodu GET /users

```
[
  {
    "username": "threatstate",
    "permissions": [
      "READ",
      "MANAGE_USERS",
      "WRITE"
    ]
  },
  {
    "username": "filip",
    "permissions": [
      "READ",
      "WRITE"
    ]
  }
]
```

2.3.3.2 GET /user/{username}

Koncový bod, který umožní uživatelům s oprávněním spravovat uživatele získat informace o konkrétním uživateli systému. Jediným povinným parametrem tohoto koncového bodu je uživatelské jméno.

Výpis kódu 2.5: Ukázková odpověď koncového bodu GET /user/threatstate

```
{
```

```
  "username": "threatstate",
  "permissions": [
    "READ",
    "MANAGE_USERS",
    "WRITE"
  ]
}
```

2.3.3.3 PUT /user/{username}

Tento koncový bod umožňuje uživatelům s oprávněním spravovat uživatele vytvořit nového uživatele nebo upravit vlastnosti existujícího uživatele. V případě vytváření nového uživatele je povinným parametrem tohoto koncového bodu jeho uživatelské jméno, heslo a seznam oprávnění. V případě úpravy existujícího uživatele je povinným parametrem pouze uživatelské jméno, volitelnými pak nový seznam oprávnění a nebo nové heslo. Odpověď tohoto koncového bodu je totožná jako v případě GET /user/{username}.

Výpis kódu 2.6: Ukázkový požadavek na koncový bod PUT /user/john

```
{
  "password": "supersecret",
  "permissions": [
    "READ"
  ]
}
```

2.3.3.4 DELETE /user/{username}

Koncový bod umožňující uživatelům s oprávněním spravovat uživatele smazat existujícího uživatele ze systému. Jediným povinným parametrem tohoto koncového bodu je uživatelské jméno. Systém na tento požadavek odpovídá standartní odpovědí.

2.3.4 Vkládání dat

Tato část obsahuje popis koncových bodů, které slouží pro vkládání dat ve formátu třetích stran a jejich zpracování systémem Threatstate.

2.3.4.1 POST /intelmq/insert

Uživatelé s oprávněním zapisovat mohou na tento koncový bod zaslat požadavek obsahující data ve formátu používaném nástrojem IntelMQ (tzv. Event). Ten je založen na formátu JSON a definuje jména jednotlivých polí a jejich datový typ. Kompletní popis datového formátu nástroje IntelMQ je součástí příloh této práce. Oproti poněkud laxním minimálním požadavkům na obsah Eventu vyžadovaným nástrojem IntelMQ[21], vyžaduje systém Threatstate přítomnost minimálně tří polí: *source.ip* (zdrojová IP adresa události),

classification.type (typ události dle povoleného výčtu nástroje IntelMQ) a *time.observation* (čas, kdy byla událost zpracována nástrojem IntelMQ). Systém na tento požadavek odpovídá standartní odpovědí.

Výpis kódu 2.7: Ukázkový požadavek na koncový bod POST /intelmq/insert

```
{
    "classification.taxonomy": "malicious-code",
    "classification.type": "infected-system",
    "destination.port": 25,
    "feed.accuracy": 100,
    "feed.name": "spamhaus-cert",
    "malware.name": "gamut",
    "protocol.transport": "tcp",
    "source.abuse_contact": "abuse@o2.cz",
    "source.asn": 5610,
    "source.geolocation.cc": "CZ",
    "source.geolocation.city": "pardubice",
    "source.geolocation.latitude": 50.0512,
    "source.geolocation.longitude": 15.7769,
    "source.ip": "88.103.230.145",
    "source.network": "88.102.0.0/15",
    "source.port": 38564,
    "time.observation": "2022-03-12T12:02:14+00:00",
    "time.source": "2022-03-12T10:32:15+00:00"
}
```

2.3.4.2 POST /shodan/insert

Uživatelé s oprávněním zapisovat mohou na tento koncový bod zaslat požadavek obsahující data ve formátu používaném službou Shodan (tzv. Banner). Ten je založen na formátu JSON a definuje jména jednotlivých polí a jejich datový typ. Specifikace datového formátu služby Shodan je součástí příloh této práce. Minimální požadavky, vyžadující přítomnost a obsah vybraných polí, odpovídá zmíněné specifikaci. Systém na tento požadavek odpovídá standartní odpovědí.

Výpis kódu 2.8: Ukázkový požadavek na koncový bod POST /shodan/insert

```
{
    "_shodan": {
        "id": "7383056c-d513-4b43-8734-b82d897888e6",
        "options": {},
        "ptr": true,
        "module": "dns-udp",
        "crawler": "9d8ac08f91f51fa9017965712c8fdabb4211dee4"
    },
    "hash": -553166942,
    "os": null,
    "opts": {
        "raw": "34ef818200010000000000007766572736f6e046269"
    },
```

```
    "ip": 134744072,
    "isp": "Google",
    "port": 53,
    "hostnames": [
        "dns.google"
    ],
    "location": {
        "city": null,
        "region_code": null,
        "area_code": null,
        "longitude": -97.822,
        "country_code3": null,
        "country_name": "United States",
        "postal_code": null,
        "dma_code": null,
        "country_code": "US",
        "latitude": 37.751
    },
    "dns": {
        "resolver_hostname": null,
        "recursive": true,
        "resolver_id": null,
        "software": null
    },
    "timestamp": "2021-01-28T07:21:33.444507",
    "domains": [
        "dns.google"
    ],
    "org": "Google",
    "data": "\nRecursion: enabled",
    "asn": "AS15169",
    "transport": "udp",
    "ip_str": "8.8.8.8"
}
```


2.3.5 Vyhledávání

Tato část popisuje koncové body sloužící pro vyhledávání dat v systému Threatstate. Pro vyhledávání je použit vlastní dotazovací jazyk inspirovaný dotazovacím jazykem služby Shodan a dotazovacím jazykem webové aplikace Kibana (Kibana Query Language). To by mělo usnadnit přechod uživatelům současného řešení.

Vlastní dotazovací jazyk, pojmenovaný Threatstate Query Language (TQL), umožňuje vyhledávat skrze všechny atributy dotazovaných objektů a podporuje datové typy *string* a *integer*. Vyhledávací výraz se skládá z názvu atributu a vyhledávané hodnoty, mezi kterými se nachází dvojtečka, která vyjadřuje rovnost. Nerovnost je možné vyjádřit pomocí klíčového slova *NOT* před vyhledávacím výrazem. Vyhledávací výrazy je možné spojovat logickými operátory pro vyjádření konjunkce a diskjunkce vyjádřené klíčovými slovy *AND* a *OR*. Pořadí operací je možné měnit použitím závorek. Pokud vy-

hledávaná hodnota obsahuje mezeru, dvojtečku nebo jiný speciální znak, je nutné jí uzavřit apostrofy nebo uvozovkami. To platí také pro vyhledávání řetězce, který obsahuje pouze číslice. Speciálním případem je vyhledávání podle IP adresy, které umožňuje vyhledávat i v adresních rozsazích.

Výpis kódu 2.9: Ukázky dotazů zapsaných v Threatstate Query Language

```
ip:"::/0" AND org:"Amazon.com, Inc."

(tags:compromised OR services.port:21) AND NOT ip:155.143.28.0/24
```

2.3.5.1 POST /host/search

Tento koncový bod umožňuje uživatelům s oprávněním číst vyhledávání informací o strojích připojených k Internetu evidovaných v systému.

Povinným parametrem je vyhledávací dotaz v poli *query* (datového typu *string*). Jeho obsahem musí být validní TQL vyhledávací dotaz. V rámci tohoto parametru je také splněn požadavek na vyhledávání v historických datech. Ten je možné specifikovat jako vyhledávací výraz `date:2021-06-21`, který omezí výsledky vyhledávání na konkrétní datum.

Volitelným parametrem je pak *count* (datového typu *integer*), který umožňuje nastavit maximální počet získaných výsledků. Pokud není parametr *count* nastaven, použije se výchozí hodnota 10.

Odpoveď systému v případě úspěchu obsahuje pole objektů Host seřazených sestupně podle jejich datumu. Popis objektu Host je součástí kompletní specifikace dle OpenAPI, která je součástí obsahu přiloženého CD. V případě neúspěchu systém zašle standartní odpověď.

Výpis kódu 2.10: Ukázkový požadavek na koncový bod POST /host/search

```
{
  "count": 1,
  "query": "ip:104.42.225.122 AND date:2021-06-21"
}
```

Výpis kódu 2.11: Ukázková odpověď koncového bodu POST /host/search

```
[
  {
    "_id": "626d7d004540a42916a9ef0c",
    "ip": "104.42.225.122",
    "date": "2021-06-21",
    "hostnames": [
      "freefinder.me"
    ],
    "tags": [
      "malicious",
      "c2"
    ],
    "location": {
```

```
      "country_code": "US",
      "country_name": null,
      "city": "San Jose"
    },
    "cloud": null,
    "isp": null,
    "org": null,
    "asn": "8075",
    "os": null,
    "devicetype": null,
    "abuse": "abuse@microsoft.com",
    "vulnerabilities": [
      "CVE-2021-44678"
    ],
    "services": [
      {
        "port": 80,
        "data": null,
        "details": [],
        "time": "2021-06-21T06:34:53+00:00",
        "transport_protocol": "tcp",
        "application_protocol": "unknown",
        "tls": null
      }
    ]
  }
]
```


2.3.5.2 POST /action/search

Tento koncový bod umožňuje uživatelům s oprávněním číst vyhledávání informací o aktivitách strojů připojených k Internetu evidovaných v systému.

Povinným parametrem je vyhledávací dotaz v poli *query* (datového typu *string*). Jeho obsahem musí být validní TQL vyhledávací dotaz. V rámci tohoto parametru je také splněn požadavek na vyhledávání v historických datech. Ten je možné specifikovat jako vyhledávací výraz `date:2021-06-21`, který omezí výsledky vyhledávání na konkrétní datum.

Volitelným parametrem je pak *count* (datového typu *integer*), který umožňuje nastavit maximální počet získaných výsledků. Pokud není parametr *count* nastaven, použije se výchozí hodnota 10.

Odpoveď systému v případě úspěchu obsahuje pole objektů Action seřazených sestupně podle jejich datumu. Popis objektu Action je součástí kompletní specifikace dle OpenAPI, která je součástí obsahu přiloženého CD. V případě neúspěchu systém zašle standartní odpověď.

Výpis kódu 2.12: Ukázkový požadavek na koncový bod POST /action/search


```
{
  "count": 1,
  "query": "type:malware-communication AND date:2021-06-21"
}
```

Výpis kódu 2.13: Ukázková odpověď koncového bodu POST /action/search


```
[
  {
    "_id": "626e8937fefe13cb98d347d4",
    "type": "malware-communication",
    "target": {
      "ip": "104.42.225.122",
      "network": "104.40.0.0/13",
      "organization": null,
      "country_code": "US",
      "port": 80,
      "application": null,
      "user": null,
      "password": null,
      "platform": null,
      "fqdn": "freefinder.me"
    },
    "time": "2021-06-21T06:34:53+00:00",
    "date": "2021-06-21",
    "ip": "46.13.178.102",
    "port": 1359,
    "application_protocol": null,
    "transport_protocol": "tcp"
  }
]
```


2.3.5.3 POST /vulnerability/search

Tento koncový bod umožňuje uživatelům s oprávněním číst vyhledávání informací o zranitelnostech a slouží jako doplněk k vyhledávání informací o strojích připojených k Internetu. V případě zranitelnosti u některého z evidovaných strojů umožní tento koncový bod uživateli získat dodatečné informace o konkrétní zranitelnosti.

Povinným parametrem je vyhledávací dotaz v poli *query* (datového typu *string*). Jeho obsahem musí být validní TQL vyhledávací dotaz.

Volitelným parametrem je pak *count* (datového typu *integer*), který umožňuje nastavit maximální počet získaných výsledků. Pokud není parametr *count* nastaven, použije se výchozí hodnota 10.

Odpoveď systému v případě úspěchu obsahuje pole objektů Vulnerability. Popis objektu Vulnerability je součástí kompletní specifikace dle OpenAPI, která je součástí obsahu přiloženého CD. V případě neúspěchu systém zašle standartní odpověď.

Výpis kódu 2.14: Ukázkový požadavek na koncový bod POST /vulnerability/search


```
{
  "count": 1,
  "query": "id:CVE-2012-3499"
}
```

Výpis kódu 2.15: Ukázková odpověď koncového bodu POST /vulnerability/search


```
[
  {
    "_id": "CVE-2012-3499",
    "name": null,
    "cvss": 4.3,
    "cvss_vector_string": null,
    "description": "Multiple cross-site scripting (XSS)
       vulnerabilities in the Apache HTTP Server 2.2.x before
       2.2.24-dev and 2.4.x before 2.4.4 allow remote attackers
       to inject arbitrary web script or HTML via vectors
       involving hostnames and URIs in the (1) mod_imagemap, (2)
        mod_info, (3) mod_ldap, (4) mod_proxy_ftp, and (5)
       mod_status modules.",
    "references": [
      "http://rhn.redhat.com/errata/RHSA-2013-0815.html",
      "http://httpd.apache.org/security/vulnerabilities_24.html",
      "http://rhn.redhat.com/errata/RHSA-2013-1207.html",
      "http://httpd.apache.org/security/vulnerabilities_22.html",
      "http://support.apple.com/kb/HT5880",
      "http://www.securityfocus.com/bid/64758",
      "http://www.debian.org/security/2013/dsa-2637",
      "http://rhn.redhat.com/errata/RHSA-2013-1208.html",
      "http://www.securityfocus.com/bid/58165",
      "http://rhn.redhat.com/errata/RHSA-2013-1209.html",
      "http://marc.info/?l=bugtraq&m=136612293908376&w=2",
    ]
  }
]
```

Kapitola 3

Implementace

Stěžejní částí implementace jsou třídy, které reprezentují evidované objekty *Host*, *Action* a *Vulnerability* se všemi jejich atributy. Tyto třídy mají díky použitým knihovnám několik důležitých vlastností. První z nich je snadná převoditelnost instancí těchto tříd do podoby datového formátu JSON, kterým systém komunikuje s uživateli. Tato vlastnost je doprovázena i opačnou schopností, kterým je snadné vytvoření instance třídy předáním atributů ve formátu JSON. Druhou důležitou vlastností, která vychází z použití knihovny Beanie, je uložení instancí těchto tříd do databáze. MongoDB používá pro ukládání dat do databáze formát BSON, který je postaven na formátu JSON, ale s tím zásadním rozdílem, že BSON je binární formát.[22] Knihovna Beanie zajišťuje převod instancí nativních Python tříd do formátu BSON pro uložení v MongoDB databázi. Samozřejmě disponuje opět také opačnou funkcionalitou, kterou je vytvoření Python objektů z dat uložených v databázi.

Požadavek na udržování historického stavu je podstatný zejména u evidovaného objektu *Host*. To je implementováno skrze atribut *date*, kterým je vyjádřeno, jaký den jeho atributy reflektují. Pro další dny je pak vytvořen nový objekt, jehož atributy opět reprezentují informace z dalšího dne. Pokud máme informace o konkrétním stroji a jeho stavu pro každý den v roce, bylo by to zaznamenáno formou jednoho objektu *Host* pro každý takový den.

Další důležitou částí je implementace třídy *Settings*. Vytvoření instance této třídy zajišťuje načtení konfigurace ze tří možných způsobů: načtení konfiguračního souboru, načtění obsahu proměnných prostředí nebo použití přepínačů na příkazové řádce. Při načítání konfigurace ze všech způsobů se respektuje jejich priorita. Konfigurační soubor má nejmenší prioritu. Jeho obsah je tedy možné přetížit pomocí použití proměnných prostředí. Stejně tak přepínače v příkazové řádce mají vyšší prioritu než použití proměnných prostředí. Třída *Settings* pro většinu konfiguračních direktiv definuje také výchozí hodnotu, která se použije v případě, že není nastavena uživatelem. Po načtení konfigurace dochází v rámci nově vytvořené instanci třídy *Settings* také k inicializaci logovacího systému.

Koncové body pro vkládání dat z nástroje IntelMQ a ze služby Shodan jsou implementovány v odděleném modulu *apps*, respektive jeho podmodulech *intelmq* a *shodan*. Tento kód je tak zcela oddělen od stěžejních částí systému, které pouze importuje pro použití. Kód koncových bodů pro vkládání různých vstupů se skládá zejména z popisu vstupních dat, která má systém přijmout, a použije se zejména pro jejich validaci. Dále má kód za úkol vytvoření instancí tříd Host, Action či Vulnerability, do jejichž atributů obsáhne přijatá data. Všechny tři zmíněné třídy implementují metodu *consume*, kterou obsluha koncových bodů na závěr zavolá a která zajistí vytvoření nových objektů v databázi a nebo jejich aktualizaci, pokud již v databázi existují.

Pro vyhledávání jsou implementovány tři třídy *HostSearch*, *ActionSearch* a *VulnerabilitySearch*, jejichž instance reprezentují uživatelem zadané parametry vyhledávání. Pro vlastní dotazovací jazyk TQL byl implementován vlastní datový typ, který vychází z nativní implementace datového typu *string*, přidává však dodatečné funkcionality, kterými je ověření správnosti vyhledávacího dotazu a atribut *parsed*, který vrátí parsovaný vyhledávací dotaz.

3.1 Vybrané problémy

3.1.1 Validování dat

Existenci knihovny Pydantic pro validování dat považuji za úžasný vynález, který vývojáři značně usnadňuje práci. Pro základní použití stačí definovat třídu (tzv. model), která dědí z Pydantic třídy BaseModel a deklarovat její atributy s anotacemi datového typu (a to i za použití vlastních složitějších datových typů). Každému atributu je možné nastavit výchozí hodnotu nebo funkci, která výchozí hodnotu vytvoří, slovní popis atributu, alternativní název a mnoho dalšího. Při inicializaci instance takové třídy dojde k validaci všech vstupních dat, která v případě nevalidních dat vyvolá vyjímku.

Použití knihovny Pydantic je základním stavebním kamenem celé této práce. Možnost pohodlné validace dat je nesmírně užitečná pro zpracování uživatelských vstupů, které mohou být chybou nebo úmyslně vadné. Aby však bylo možné data validovat, je třeba pro ně definovat zmíněný Pydantic model, který je bude popisovat. Stěžejními a komplexními vstupy, které systém Threatstate zpracovává, jsou data ve formátu používaném nástrojem IntelMQ a služby Shodan. Tvorba Pydantic modelu pro data ve formátu používaném nástrojem IntelMQ byla poměrně přímočará, jedná se celkem o osmdesát atributů, které jsou popsány v dokumentaci nástroje IntelMQ.[23] Komplikovanější případ bylo vytvoření Pydantic modelu pro data ve formátu služby Shodan. Ten rozlišuje, v době psaní této práce, devadesát sedm různých protokolů a produktů. Pro každý z nich existuje řada atributů popisující vlastnosti daného protokolu či produktu, které služba Shodan zjišťuje. Všechny jsou popsány v dokumentaci služby Shodan[5], kde k nim lze nalézt také strojově čitelné schéma podle specifikace OpenAPI. Bylo tak možné použít

Python nástroj *datamodel-code-generator*, který umožňuje automaticky vygenerovat Pydantic model na základě takového schématu. Vygenerovaný model, který obsahuje téměř devatenáct set řádků kódu, bylo z vícero důvodů nutné následně manuálně upravit. Prvním z nich je důvod, že datový typ některých atributů uvedených ve zveřejněném schématu služby Shodan je nepřesný, respektive nedostačující pro učely validace. Jako příklad bych zmínil atribut *timestamp*, jehož datový typ schéma definuje jako *string*. Validace, že časové razítko je datového typu řetězec, je pro účel této práce nedostačující. Druhým důvodem je, že služba Shodan schéma průběžně rozšiřuje o další protokoly a produkty, které je schopna zjišťovat. Prvotně automaticky vygenerovaný model je tak nutné přizpůbosit novým rozšířením služby Shodan, nelze však provést opětovné automatické generování kvůli již provedeným manuálním úpravám. Navíc jsem v průběhu vývoje narazil i na situaci, kdy data ze služby Shodan obsahovala atributy, které v danou chvíli ani nebyly popsány ve schématu.

Vytvoření Pydantic modelu, který by byl užitečný pro validaci dat ve formátu používaném službou Shodan, se ukázalo jako komplexní a pracný problém. Vytvořený model je navíc nutné pravidlně aktualizovat, aby reflekoval nová rozšíření implementovaná služou Shodan. V neposlední řadě by takový model mohl najít uplatnění i v jiných projektech mimo tuto práci. Proto jsem se rozhodl oddělit vytvořený Pydantic model do samostatného Python modulu pojmenovaném *pydantic-shodan*, který je v současné chvíli možné stáhnout a nainstalovat z centrálního repozitáře Python Package Index (PyPI).

3.1.2 Implementace datového typu IP adresa

Jedním z prvních implementačních problémů, se kterými jsem věděl, že se budu muset vypořádat, bylo vyhledávání v datech podle IP adresy nebo celého síťového rozsahu (například vyhledání informací o všech strojích v síťovém rozsahu 217.31.207.0/24). IP adresa je stěžejním identifikátorem, když mluvíme o počítačích připojených k Internetu a databázový systém MongoDB bohužel nativně nepodporuje takto specifický datový typ. Pokud však budeme s IP adresou pracovat jako s číslem, můžeme snadno hledat rovnosti, tedy vyhledávat podle konkrétní adresy. Pro vyhledávání podle síťového rozsahu je však situace o něco komplikovanější. Zadaný síťový rozsah můžeme reprezentovat jako dvě IP adresy: první je IP adresa sítě (tedy taková adresa, kde jsou všechny bity na pozici masky sítě nastaveny na nulu) a druhá je IP adresa broadcastu (adresa, kde jsou všechny bity na pozici masky sítě nastaveny na jedničku). Výsledkem hledání podle síťového rozsahu budou všechny adresy, které jsou jako číslo větší nebo rovné, než IP adresa sítě a zároveň menší nebo rovné, než IP adresa broadcastu. MongoDB sice podporuje číslo jako datový typ, avšak pouze ve formě 32bitového nebo 64bitového čísla se znaménkem. Pro IPv4 adresu, která má délku 32 bitů, by taková velikost stačila a můžeme ji

do databáze uložit. Pro IPv6 adresu, která má délku 128 bitů, bohužel stačit nebude a nelze jí proto uložit do databáze jako jedno velké číslo. Proto jsem se rozhodl IPv6 adresu rozložit na čtyři čísla o délce 32 bitů. MongoDB naštěstí podporuje datový typ pole, což jsem pro řešení použil a v databázi je tak IPv6 adresa reprezentována jako pole čtyř čísel. Vyhledávání podle adresy či rozsahu je následně řešeno podobně jako u IPv4. Před vyhledáváním je třeba vyhledávánou IPv6 adresu také převést do podoby pole čtyř čísel, která budou při vyhledávání porovonávána s hodnotami uloženými v databázi. Pro vyhledávání podle IPv6 adresního rozsahu jsou opět určena nejnižší adresa sítě a nejvyšší adresa broadcastu. Obě adresy jsou převedeny do podoby pole čtyř čísel a vyhledávání probíhá porovnáváním, kde výsledkem jsou adresy, které mají všechny čtyři čísla rovná nebo vyšší než adresa sítě a nižší nebo rovné adrese broadcastu.

Výsledná implementace je provedena tak, aby rozšiřovala použitou knihovnu Beanie a nevyžadovala při použití navazujícím kódem žádný speciální přístup.

3.1.3 Vlastní dotazovací jazyk pro vyhledávání

Řešení dotazovacího jazyka pro vyhledávání bylo zajímavým implementačním problémem, které se několikrát ocitlo ve slepé uličce. Prvotní myšlenka implementace od základu vlastního parseru byla zavržena pro svoji komplexnost. Proto nakonec byla zvolena knihovna pyparsing, která umožnila popis vlastní jednoduché gramatiky, podle které také umožňuje parsovat vstupy. Výstupem parsování podle mnou navržené gramatiky je pole, které obsahuje jednotlivé řetězce nebo další rekurzivně zanořená pole.

Samotná definice gramatiky a její parser je umístěn v abstraktní třídě DocumentQuery. Z té potom dědí třídy, které jsou používány pro parsování uživatelem zadaných vyhledávacích dotazů pro jednotlivé vyhledatelné objekty Host, Action a Vulnerability. Při spuštění systému Threatstate se gramatika dotazovacího jazyka pro vyhledávání zmíněných objektů vytvoří dynamicky na základě atributů těchto objektů. To je možné díky tomu, že je gramatika definováná přímo v jazyce Python. Dynamická tvorba gramatiky umožňuje, aby například v budoucnu přidané atributy objektů Host, Action a Vulnerability byly hned po spuštění systému použitelné při vyhledávání bez zásahu do kódu vyhledávání nebo popisu gramatiky. Další přidaná hodnota tohoto řešení nastane v situaci, kdy uživatel zadá chybně název pole, podle kterého se má vyhledávat. Název pole obsahující překlep či jinou chybu je odhalen a uživatel je informován, že se jednalo a nevalidní dotaz. Nevalidní dotaz nebude dále zpracováván a nedojde k vyhledávání v databázi, které by zbytečně vyčerpávalo vypočetní výkon. Samotná abstraktní třída DocumentQuery dědí z nativního datového typu *string* a má implementovány také metody pro validaci, které vyžaduje knihovna Pydantic, aby bylo možné třídu

(resp. třídy, které z této abtraktní třídy dědí) použít pro anotaci datového typu.

V případě, že je uživatelem zadaný dotaz správný a podle definované gramatiky dojde k jeho úspěšnému parsování, je výstup parsování předán další metodě `build_mongo_query`, která, jak název napovídá, sestaví výsledný dotaz pro vyhledání v Mongo databázi. Následné ukázky demonstrují celý proces zpracování uživatelem zadaného dotazu.

Výpis kódu 3.1: Příklad uživatelem zadaného dotazu

```
NOT NOT ip:217.20.0.0/16 AND org:"Amazon.com, Inc." OR os:Linux
```

Výpis kódu 3.2: Python struktura po parsovaní uživatelem zadaného dotazu

```
[
    [
        [
            'NOT',
            ['NOT', 'ip:217.20.0.0/16']
        ],
        'AND',
        "org:'Amazon.com, Inc.'"
    ],
    'OR',
    'os:Linux'
]
```

Výpis kódu 3.3: Převedení parsovaného dotazu zadaného uživatelem do databázového dotazu MongoDB

```
{
  "$or": [
    {
      "$and": [
        {
          "ip": {
            "$size": 1,
            "$gte": 3641966592,
            "$lte": 3642032127
          }
        },
        {
          "org": "Amazon.com, Inc."
        }
      ]
    },
    {
      "os": "Linux"
    }
  ]
}
```

3.1.4 Interpretace dat z nástroje IntelMQ

Nástroj IntelMQ definuje zmíněný datový formát Event v podobě JSONu, pro který definuje celkem osmdesát atributů. Do těchto osmdesáti atributů se snaží mapovat sbíraná data. Pokud nějaká část sbíraných dat nelze mapovat do předem definovaných atributů, jsou uložena do dodatečných atributů s libovolným názvem začínajícím řetězcem *extra*. Obsah těchto dodatečných atributů je nedefinovaný a nestrukturovaný, protože ho nelze předvídat, ale zabraňuje ztrátě dat. Dále nastavuje stěžejní atribut *classification.type* na jednu z předem definovaných výčtových hodnot, která má vyjadřovat celkový charakter informace. Může se jednat například o hodnotu *vulnerable-system* nebo třeba *scanner*. Ostatní atributy pak slouží k popisu takové informace a doplňují konkrétní detaily. Jedním z těch nejdůležitejších pro systém Threatstate je informace o zdrojové IP adrese, která je vyjádřena atributem *source.ip*. Taková informace dává smysl pro oba zmíněné případy, kdy se jedná o zranitelný systém (v takovém případě bude chápána jako IP adresa zranitelného systému), tak pro případ skeneru (v tomto případě je zdrojová IP adresa chápána jako IP adresa stroje, který provádí skenování). Dalším důležitým atributem bude informace o cílové adrese, která je vyjádřena atributem *destination.ip*. Pro zmíněný případ skeneru se opět jedná o naprosto smysluplnou informaci, která vyjadřuje, jaký stroj byl cílem onoho skenování. Pokud by se však objevila informace o cílové IP adrese v případě zranitelného systému (jehož IP adresa již byla obsažena v atributu *source.ip*), jak takovou informaci budeme chápat? A jak takovou informaci interpretovat algoritmicky?
Z pohledu nástroje IntelMQ se technicky jedná o validí Event.

Dokumentace nástroje IntelMQ poskytuje určitou nápovědu při interpretaci nejobvyklejších klasifikačních typů[24], ta však nechává mnoho případů nezodpovězených. Dosud byla interpretace dat z nástroje IntelMQ manuální prací, kterou musel provádět bezpečnostní analytik. Systém Threatstate se tuto roli snaží převzít. Bezpečnostní analytik má však dodatečné možnosti interpretace oproti algoritmu systému Threatstate: vyčíst informace z nestrukturovaných dat a použití selského rozumu. Tento problém byl identifikován až v průběhu implementace této práce a jako její autor neznám jeho ideální řešení. Pro implementaci jsem se rozhodl vycházet ze způsobu, jakým nástroj IntelMQ zpracovává data v současně používaném řešení a za současné konfigurace. Má implementace také provadí řadu předpokladů založených na běžném chápání jednotlivých klasifikačních typů. Jsem si však vědom, že taková implementace může odporovat způsobu používání nástroje IntelMQ v jiných organizacích a snižuje tak její univerzálnost použití. Dále může vést k určité ztrátě informací, které by odporovaly mnou implementovanému způsobu interpretace.

3.2 Verzování

Pro verzování kódu je použit nástroj Git. Vzdálený repozitář byl po dobu tvorby této práce umístěn na GitLabu Fakulty informačních technologií dostupném na adrese gitlab.fit.cvut.cz/pokorfi1/threatstate. Pro budoucí vývoj bude využit veřejný repozitář, poskytovaný službou GitHubu, dostupný na webové adrese github.com/threatstate/threatstate.

3.3 Testování

Pro testování implementovaných funkcionalit je připravena sada jednotkových testů v ukázkovém rozsahu pro spuštění testovacím nástrojem *pytest*. Cílem jednotkových testů je ověřit správnost fungování jednotlivých částí zdrojového kódu. V rámci připravených jednotkových testů je ověřena správnost komunikace s jednotlivými koncovými body, správnost zpracování a vyhodnocení vyhledávacích dotazů, správnost implementace vlastních datových typů (zejména IP adresa) a správnost vyhledávání. Pro spuštění testů je nutný přístup do databázového systému MongoDB.

3.4 Sestavení

Pro přípravu systému na provoz v kontejnerizovaném prostředí Dockeru je připraven konfigurační soubor Dockerfile, který slouží k sestavení Docker obrazu. Soubor Dockerfile je součástí git repozitáře projektu Threatstate, jehož kořenový adresář je nutné použít jako tzv. build context[25].

Výpis kódu 3.4: Dockerfile pro sestavení Docker obrazu

```
# ------------ Python builder ------------
FROM python:3.9-bullseye as builder

ENV DEBIAN_FRONTEND=noninteractive \
    PATH=/opt/venv/bin:${PATH}

RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

WORKDIR /opt/threatstate/

COPY setup.cfg /opt/threatstate/setup.cfg

RUN python3 -m venv /opt/venv && \
    pip install pip-tools && \
    pip-compile --quiet --output-file=requirements.txt /opt/
        threatstate/setup.cfg && \
    pip uninstall -y pip-tools && \
```

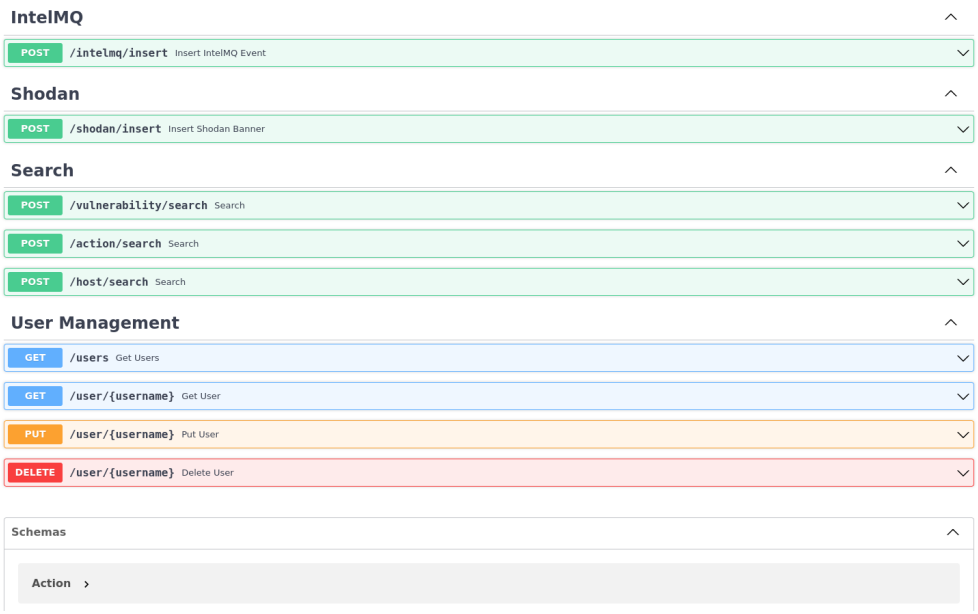
```
    pip install -r requirements.txt

# ------------ Runtime ------------
FROM python:3.9-bullseye as runtime

ENV DEBIAN_FRONTEND=noninteractive \
    PATH=/opt/venv/bin:${PATH}

RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

COPY --from=builder /opt/venv /opt/venv

COPY docker-entrypoint.sh /docker-entrypoint.sh

COPY setup.cfg /opt/threatstate/setup.cfg

COPY pyproject.toml /opt/threatstate/pyproject.toml

COPY threatstate /opt/threatstate/threatstate

RUN pip install /opt/threatstate

RUN threatstate setup && \
    chown threatstate:threatstate /docker-entrypoint.sh && \
    chmod 544 /docker-entrypoint.sh && \
    rm -rf /opt/threatstate

USER threatstate

RUN threatstate check

EXPOSE 8000

ENTRYPOINT ["/docker-entrypoint.sh"]

CMD ["threatstate", "start"]
```

Kapitola 4

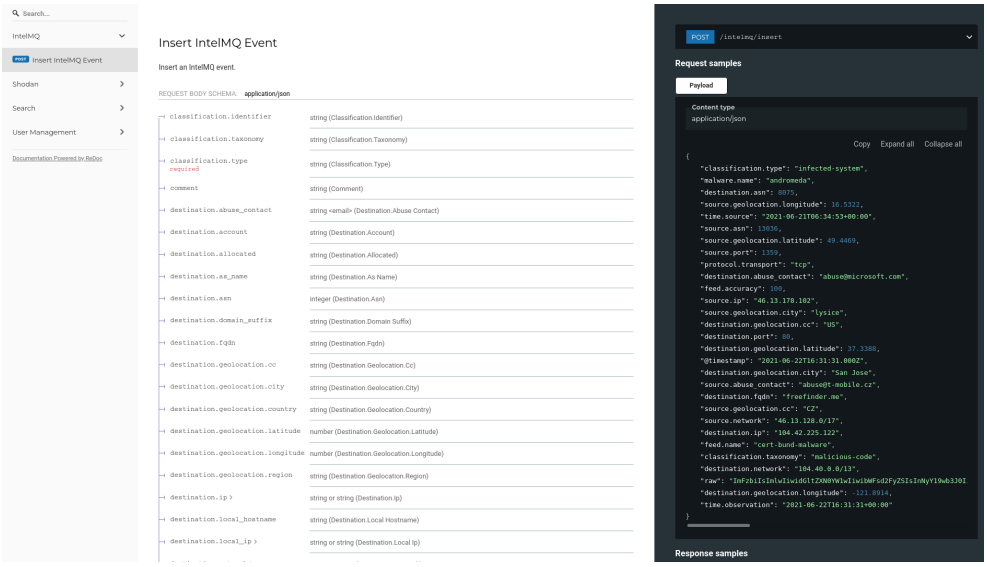
Dokumentace

Dokumentace systému Threatstate se skládá z několika částí. První z nich je automaticky generovaná online webová dokumentace koncových bodů, jejíž dynamické vytvoření zajišťuje použitý framework FastAPI při spuštění systému. Dokumentace tak okamžitě reflektuje změny při vývoji systému a je dostupná ve dvou různých grafických podobách. Protože je tato dokumentace spojena s během systému, umožňuje tak interakci s implementovanými koncovými body skrze jednoduché grafické webové rozhraní.

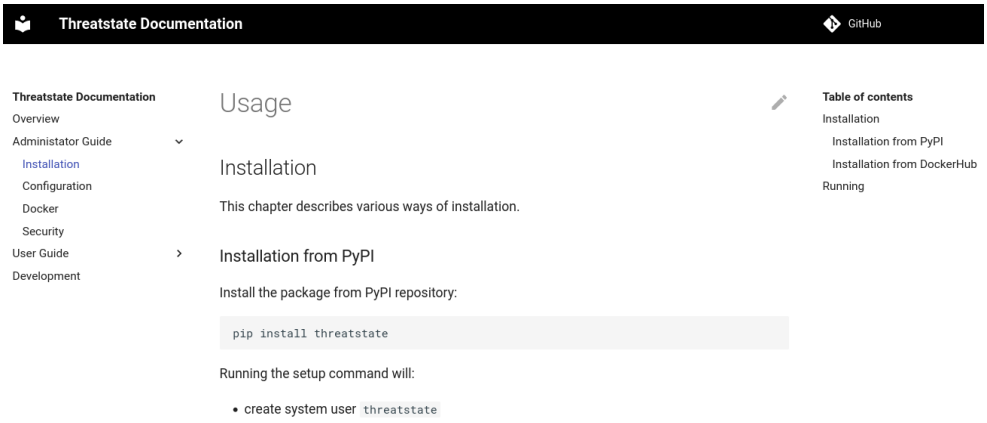
Druhou částí dokumentace je administrátorská příručka, která obsahuje sadu návodů pro instalaci, konfiguraci a spuštění systému. Ta je ručně sepsána v anglickém jazyce ve formátu Markdown a následně převedena nástrojem *mkdocs* do podoby statické webové stránky. Webová stránka s administrátorskou příručkou v anglickém jazyce je zveřejněna na webové adrese threatstate.github.io. Administrátorská příručka v českém jazyce je obsahem následující podkapitoly.



Obrázek 4.1: Ukázka automaticky generované dokumentace



Obrázek 4.2: Ukázka automaticky generované dokumentace v alternativní podobě



Obrázek 4.3: Ukázka anglické administrátorské příručky v podobě webové stránky

4.1 Administrátorská příručka

4.1.1 Instalace a spuštění

Tato část popisuje různé způsoby instalace a spuštění systému Threatstate.

4.1.1.1 Docker

Pro běh v kontejnerizovaném prostředí Dockeru je připravena ukázková konfigurace. Pro spuštění je potřeba mít nainstalovaný nástroj *docker* a *docker-compose*. Postup instalace těchto nástrojů je mimo rozsah této práce. Ukázková konfigurace stáhne předpřipravené obrazy kontejnerů z centrálního repozitáře DockerHub a spustí kromě samotného systému Threatstate také databázový systém MongoDB.

Výpis kódu 4.1: Ukázková konfigurace docker-compose.yml

```
version: '3.1'

services:

  threatstate:
    image: threatstate/threatstate
    restart: always
    ports:
      - 80:8000
    environment:
      THREATSTATE_DATABASE_HOST: mongo
      THREATSTATE_DATABASE_PORT: 27017
      THREATSTATE_DATABASE_USERNAME: root
      THREATSTATE_DATABASE_PASSWORD: supersecret
```

```
mongo:
  image: mongo
  restart: always
  environment:
    MONGO_INITDB_ROOT_USERNAME: root
    MONGO_INITDB_ROOT_PASSWORD: supersecret
```

4.1.1.2 Python Package Index (PyPI)

Druhou možností instalace je stažení Python modulu Threatstate z repozitáře Python Package Index. Tato možnost závisí na přítomnosti nástroje *pip* v operačním systému, který zprostředkovává stažení a instalaci Python modulů. Ten je v operačním systému Linux obvykle součástí balíčku *python3-pip*. Modul threatstate je pak možné nainstalovat následujícím příkazem:

```
pip install threatstate
```

Následujícím příkazem dojde k vytvoření systémového uživatele *threatstate*, vytvoření výchozí konfigurace umístěné v `/etc/threatstate/threatstate.conf` a vytvoření adresáře `/var/log/threatstate` pro umístění logů.

```
sudo threatstate setup
```

Výchozí konfiguraci je nutné upravit a nastavit správné přihlašovací údaje k databázovému systému MongoDB. Postup instalace databázového systému MongoDB je mimo rozsah této práce. Po úpravě konfigurace je možné spustit systém Threatstate následujícím příkazem:

```
sudo -u threatstate threatstate start
```

4.1.2 Konfigurace

Tato kapitola popisuje různé způsoby, kterými lze systém Threatstate konfigurovat. Protože systém umožňuje konfiguraci několika různými způsoby, které mají odlišnou prioritu, je možné získat výpis výsledné konfigurace pomocí následujícího příkazu:

```
threatstate config
```

4.1.2.1 Konfigurační soubor

Konfigurační soubor v `/etc/threatstate/threatstate.conf` je výchozím umístěním. Konfigurační soubor využívá formát YAML pro větší uživatelskou přívětivost. Výchozí cestu ke konfiguračnímu souboru lze změnit použitím přepínače `-f` nebo `--file`. Příklad použití:

```
threatstate --file mojekonfigurace.conf start
```

4.1.2.2 Proměnné prostředí

Dalším způsobem, kterým lze konfigurovat systém Threatstate, je skrze proměnné prostředí. Tato vlastnost je užitečná a široce rozšířená zejména pro konfiguraci aplikací běžících v Dockeru. Konfigurace skrze proměnné prostředí má vyšší prioritu než hodnoty z konfiguračního souboru. Příklad použití:

```
THREATSTATE_PORT=4444 threatstate start
```

4.1.2.3 Přepínače

Posledním způsobem konfigurace je použití přepínačů v příkazové řádce. Ty umožňuje pouze omezené možnosti konfigurace, mají však nejvyšší prioritu. Příklad použití:

```
threatstate start --port 4444
```

Kapitola 5

Navazující vývoj

V této kapitole se zabývám náměty pro navazující vývoj systému, které jsem sbíral v průběhu implementace současné verze. Obsahují náměty, které se týkají uživatelské přívětivosti, rozšíření funkčnosti nebo vylepšeného zpracovaní některých krajních případů.

5.1 Rozšíření dotazovacího jazyka pro vyhledávání

Současná podoba dotazovacího jazyka pro vyhledávání lze rozšířit v několika směrech. Prvním z nich je umožnit tzv. fulltextové vyhledávání, které by umožnilo vyhledávat zadanou hodnotu napříč všemi atributy vyhledávaného objektu. U tohoto způsobu vyhledávání je nutné brát v potaz potenciálně vysokou zátěž systémových zdrojů.

Méně ambiciózní je pak rozšíření vyhledávání o možnost vyjádřit pouze existenci a neprázdnost určitého atributu. Podoba vyhledávacího výrazu by pak mohla být například `os:*`, jehož výsledkem by byly všechny objekty, který takový atribut mají vyplněný.

Další možností rozšíření dotazovacího jazyka, respektive jeho způsobu vyhodnocování, se týká validace datových typů. Současná podoba umožňuje například vyhledávat číslo v atributu, o kterém systém ví, že je datového typu *string*. Takový dotazovací dotaz je v současné implementaci validní, bude podle něj sestaven databázový dotaz, který však nepřinese žádný výsledek a pouze zbytečně zatěžuje systémové zdroje. Toto by v budoucnu mohlo být řešeno například pokusem o konverzi datového typu a v případě neúspěchu odesláním odpovědi, že zadaný dotaz není validní.

Posledním námětem, týkající se rozšíření dotazovacího jazyka, je zvýšení jeho uživatelské přívětivosti. Některé atributy vyhledávaných objektů mohou mít velmi dlouhá jména, která musí uživatel zadat, pokud chce podle takového atributu vyhledávat. V budoucnu by pro tyto názvy mohly být implementovány aliasy, které by uživateli umožnily psaní kratších vyhledávacích výrazů.

5.2 Rozdělení objektů na menší části

Součástí dat získaných ze služby Shodan jsou některé atributy, které mohou mít nepředvídatelnou velikost. Jedná se o velmi vzácné případy, které však mohou narazit na limit maximální povolené velikosti objektu ukládaného do databáze MongoDB, který je v současnosti 16MB. Jako příklad mohu zmínit sken HTTP služby, ve kterém služba Shodan ukládá také data obrázku na webové stránce použitého jako Favicon, který obvykle nabývá velikosti v řádech jednotek až desítek kilobajtů. V jednom ze zkoumaných případů však měl tento obrázek velikost přes 12MB. Je tak zcela reálné, že v těchto vzácných případech dojde k překročení maximální velikosti objektu a ten tak nebude do databáze uložen. To povede ke ztrátě dat, nicméně nedojde k ohrožení konzistence dat v systému, protože ten se ze své povahy snaží o kumulaci maximálního možného množství dat, ale nespolehá se na jejich existenci v systému. V současnosti považuji tento postup, kdy v případě překročení limitu velikosti dojde k zahození dat, za přijatelný. Je na zvážení, zda-li v budoucnu limit navýšit a data ukládat jako zvláštní přílohu například na disk. Ovšem i v takovém případě bude nutné nastavit limit a definovat postup v případě jeho překročení.

5.3 Zpracování vstupních dat z dalších zdrojů

Požadavky získané během analýzy jasně určily, že systém má být schopen zpracovávat data ve formátu používaném nástrojem IntelMQ a službou Shodan. Je možné, že v budoucnu bude potřeba systém rozšířit o schopnost zpracování dalších vstupních formátů.

5.4 Filtrování rezervovaných rozsahů IP adres

V současné implementaci není řešen případ, kdy by se získaná data týkala strojů s IP adresou z rezervovaných rozsahů IP adres. Je tak možné, aby systém obsahoval data o stroji, jehož udaná IP adresa se ve skutečnosti na Internetu nemůže vůbec objevit. Tento problém vyžaduje hlubší analýzu a zvážení možných scénářů řešení.

5.5 Implementace grafického webového rozhraní

Výstupem této práce je backendový systém s REST aplikačním rozhraním pro komunikaci. Toto rozhraní mohou využít i uživatelé, ale jeho uživatelská přívětivost není pro pravidelné používání komfortní. Implementované aplikační rozhraní je vhodné spíše pro komunikaci s dalšími systémy. Tím by v buducnu mohla být například webová aplikace, která implementuje grafické rozhraní pro zvýšení uživatelské přívětivosti při používání.

Závěr

Cílem této práce bylo vytvořit nový backend pro ukládání a vyhledávání bezpečnostních dat, který by splňoval požadavky sestavené na základě analýzy současného řešení a odstranil by jeho identifikované nedostatky. Tento cíl byl z větší části splněn.

Nedostatek absence logiky v současném řešení, která by řídila ukládání sbíraných dat, je nově vytvořeným backendem odstraněn. Ten už neukládá vstupní data jako proud záznamů, ale implementuje objekty, jejichž atributy jsou vyplněny na základě vstupních dat. Implementovaná logika ukládání prokázala, že je možné data z nástroje IntelMQ a data ze služby Shodan ukládat jednotně a s reflexí již existujících dat. Je přesvědčením autora, že tato nová logika ukládání je dostatečně abstraktní, aby bylo možné v budoucnu rozšířit systém Threatstate o vstupní data z dalších zdrojů.

Na základě nového způsobu ukládání dat byl implementován také nový způsob vyhledávání v datech a to za použití vlastního dotazovacího jazyka. Nový způsob ukládání a vyhledávání odstraňuje nedostatek, kdy bezpečnostní analytik musel vyhledávat v oddělených evidencích podle původu dat a procházet velké množství nezávislých záznamů, aby sestavil celkovou požadovanou informaci.

Posledním nedostatkem, který byl v rámci analýzy popsán, je vyhodnocení ukládaných dat, tedy aby například řada pokusů o přihlášení byla vyhodnocena jako útok hrubou silou. Tento nedostatek nebyl odstraněn, nebyl však ale ani součástí sestavených požadavků. Práce se v první řadě zaměřila na vytvoření nového způsobu ukládání a vyhledávání dat, jejich komplexní vyhodnocování je však mimo rozsah této práce.

Nově vytvořený backend by měl být přínosem pro práci bezpečnostních analytiků. Pro další vyhodnocení jeho funkčností bude nasazen v testovacím provozu v rámci zaměstnání autora této práce. Díky tomuto vyhodnocení a zpětné vazby od uživatelů budou sestaveny náměty pro další rozvoj. Jedním z uživatelů systému bude i autor této práce, který má zájem na tom, aby byl systém opravdu přínosem a byl dále rozvíjen nad rámec této práce.

Vedlejším přínosem práce je také několik příspěvků do open-source projektů, které byly v rámci implementace použity, a ve kterých byla objevena chyba či námět ke zlepšení.

Bibliografie

1. CERT.AT. *IntelMQ Documentation: Introduction* [online] [cit. 2021-12-19]. Dostupné z: `https://intelmq.readthedocs.io/en/maintenance/user/introduction.html`.
2. ELASTICSEARCH B.V. *Elastic: What is Elasticsearch?* [Online] [cit. 2022-05-05]. Dostupné z: `https://www.elastic.co/what-is/elasticsearch`.
3. ELASTICSEARCH B.V. *Elastic: What is Kibana?* [Online] [cit. 2022-05-05]. Dostupné z: `https://www.elastic.co/what-is/kibana`.
4. CERT.AT. *IntelMQ Documentation: Data Format* [online] [cit. 2022-03-12]. Dostupné z: `https://intelmq.readthedocs.io/en/maintenance/dev/data-format.html`.
5. SHODAN.IO. *Shodan Datapedia: Overview* [online] [cit. 2022-03-12]. Dostupné z: `https://datapedia.shodan.io/`.
6. DEFINITONS.NET. *FURPS* [online] [cit. 2022-05-08]. Dostupné z: `https://www.definitions.net/definition/FURPS`.
7. MISP PROJECT. *MISP Project: MISP features and functionalities* [online] [cit. 2022-05-05]. Dostupné z: `https://www.misp-project.org/features/`.
8. CHOPITEA, Thomas. *YETI Platform: Introducing the Yeti* [online] [cit. 2022-05-07]. Dostupné z: `https://yeti-platform.github.io/introducing-yeti`.
9. CENSYS, INC. *Censys: Attack Surface Management Leaders* [online] [cit. 2021-12-19]. Dostupné z: `https://censys.io/data-and-search/`.
10. SHODAN.IO. *Shodan: Monitor* [online] [cit. 2021-12-19]. Dostupné z: `https://monitor.shodan.io/`.
11. STACK EXCHANGE INC. *Developer Survey 2021: Most popular technologies* [online] [cit. 2022-04-27]. Dostupné z: `https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies`.

12. STACK EXCHANGE INC. *Developer Survey 2021: Web framworks* [online] [cit. 2022-04-27]. Dostupné z: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks.
13. COLVIN, Samuel. *FastAPI: Features* [online] [cit. 2022-04-27]. Dostupné z: https://fastapi.tiangolo.com/features/.
14. OPENAPI INITIATIVE. *OpenAPI Specification v3.1.0* [online] [cit. 2022-04-30]. Dostupné z: https://spec.openapis.org/oas/v3.1.0.
15. RIGHT, Roman. *Beanie Documentation: Overview* [online] [cit. 2022-04-27]. Dostupné z: https://roman-right.github.io/beanie/.
16. MCGUIRE, Paul T. *Pyparsing Module: PyParsing 3.0.9 Documentation* [online] [cit. 2022-04-30]. Dostupné z: https://pyparsing-docs.readthedocs.io/en/latest/pyparsing.html.
17. MONGODB, INC. *What is MongoDB?* [Online] [cit. 2022-05-08]. Dostupné z: https://www.mongodb.com/what-is-mongodb.
18. DOCKER, INC. *Docker Documentation: Overview* [online] [cit. 2022-03-12]. Dostupné z: https://docs.docker.com/get-started/overview/.
19. CHEATSHEETS SERIES TEAM. *OWASP Cheat Sheet Series: Password Storage* [online] [cit. 2022-04-27]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html.
20. DUSSEAULT, L. *RFC4918: HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)* [online] [cit. 2022-04-30]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc4918.
21. CERT.AT. *IntelMQ Documentation: Data Format: Minimum recommended requirements for events* [online] [cit. 2022-03-12]. Dostupné z: https://intelmq.readthedocs.io/en/maintenance/dev/data-format.html#minimum-recommended-requirements-for-events.
22. MONGODB, INC. *Explaining BSON with Examples* [online] [cit. 2022-05-08]. Dostupné z: https://www.mongodb.com/basics/bson.
23. CERT.AT. *IntelMQ Documentation: Harmonization Fields* [online] [cit. 2022-03-12]. Dostupné z: https://intelmq.readthedocs.io/en/maintenance/dev/harmonization-fields.html.
24. CERT.AT. *IntelMQ Documentation: Data Format: Meaning of source and destination identities* [online] [cit. 2022-05-05]. Dostupné z: https://intelmq.readthedocs.io/en/maintenance/dev/data-format.html#meaning-of-source-and-destination-identities.
25. DOCKER, INC. *Docker Documentation: docker build* [online] [cit. 2022-04-30]. Dostupné z: https://docs.docker.com/engine/reference/commandline/build/#extended-description.

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

HTTP Hypertext Transport Protocol

JSON JavaScript Object Notation

BSON Binary JavaScript Object Notation

TOR The Onion Router

TQL Threatstate Query Language

Příloha B

Datový formát používaný nástrojem IntelMQ

Následující tabulka obsahuje popis atributů datového formátu používaného nástrojem IntelMQ. Je převzata z dokumentace nástroje IntelMQ[4] a ponechána s původními vysvětlivkami v anglickém jazyce, aby nedošlo k nepřesnostem v důsledku překladu.

`classification.identifier`	The lowercase identifier defines the actual software or service (e.g. heartbleed or ntp_version) or standardized malware name (e.g. zeus). Note that you MAY overwrite this field during processing for your individual setup. This field is not standardized across IntelMQ setups/users.
`classification.taxonomy`	We recognize the need for the CSIRT teams to apply a static (incident) taxonomy to abuse data. With this goal in mind the type IOC will serve as a basis for this activity. Each value of the dynamic type mapping translates to a an element in the static taxonomy. The European CSIRT teams for example have decided to apply the eCSIRT.net incident classification. The value of the taxonomy key is thus a derivative of the dynamic type above. For more information about check ENISA taxonomies.

`classification.type`	The abuse type IOC is one of the most crucial pieces of information for any given abuse event. The main idea of dynamic typing is to keep our ontology flexible, since we need to evolve with the evolving threatscape of abuse data. In contrast with the static taxonomy below, the dynamic typing is used to perform business decisions in the abuse handling pipeline. Furthermore, the value data set should be kept as minimal as possible to avoid type explosion, which in turn dilutes the business value of the dynamic typing. In general, we normally have two types of abuse type IOC: ones referring to a compromised resource or ones referring to pieces of the criminal infrastructure, such as a command and control servers for example.
`comment`	Free text commentary about the abuse event inserted by an analyst.
`destination.abuse_contact`	Abuse contact for destination address. A comma separated list.
`destination.account`	An account name or email address, which has been identified to relate to the destination of an abuse event.
`destination.allocated`	Allocation date corresponding to BGP prefix.
`destination.as_name`	The autonomous system name to which the connection headed.
`destination.asn`	The autonomous system number to which the connection headed.
`destination.domain_suffix`	The suffix of the domain from the public suffix list.
`destination.fqdn`	A DNS name related to the host from which the connection originated. DNS allows even binary data in DNS, so we have to allow everything. A final point is stripped, string is converted to lower case characters.
`destination.geolocation.cc`	Country-Code according to ISO3166-1 alpha-2 for the destination IP.
`destination.geolocation.city`	Some geolocation services refer to city-level geolocation.
`destination.geolocation.country`	The country name derived from the ISO3166 country code (assigned to cc field).
`destination.geolocation.latitude`	Latitude coordinates derived from a geolocation service, such as MaxMind geoip db.
`destination.geolocation.longitude`	Longitude coordinates derived from a geolocation service, such as MaxMind geoip db.
`destination.geolocation.region`	Some geolocation services refer to region-level geolocation.
`destination.geolocation.state`	Some geolocation services refer to state-level geolocation.
`destination.ip`	The IP which is the target of the observed connections.

destination.local_hostname	Some sources report a internal hostname within a NAT related to the name configured for a compromized system.
destination.local_ip	Some sources report a internal (NATed) IP address related a compromized system. N.B. RFC1918 IPs are OK here.
destination.network	CIDR for an autonomous system. Also known as BGP prefix. If multiple values are possible, select the most specific.
destination.port	The port to which the connection headed.
destination.registry	The IP registry a given ip address is allocated by.
destination.reverse_dns	Reverse DNS name acquired through a reverse DNS query on an IP address. N.B. Record types other than PTR records may also appear in the reverse DNS tree. Furthermore, unfortunately, there is no rule prohibiting people from writing anything in a PTR record. Even JavaScript will work. A final point is stripped, string is converted to lower case characters.
destination.tor_node	If the destination IP was a known tor node.
destination.url	A URL denotes on IOC, which refers to a malicious resource, whose interpretation is defined by the abuse type. A URL with the abuse type phishing refers to a phishing resource.
destination.urlpath	The path portion of an HTTP or related network request.
event_description.target	Some sources denominate the target (organization) of a an attack.
event_description.text	A free-form textual description of an abuse event.
event_description.url	A description URL is a link to a further description of the the abuse event in question.
event_hash	Computed event hash with specific keys and values that identify a unique event. At present, the hash should default to using the SHA1 function. Please note that for an event hash to be able to match more than one event (deduplication) the receiver of an event should calculate it based on a minimal set of keys and values present in the event. Using for example the observation time in the calculation will most likely render the checksum useless for deduplication purposes.
extra	All anecdotal information, which cannot be parsed into the data harmonization elements. E.g. os.name, os.version, etc. Note: this is only intended for mapping any fields which can not map naturally into the data harmonization. It is not intended for extending the data harmonization with your own fields.
feed.accuracy	A float between 0 and 100 that represents how accurate the data in the feed is
feed.code	Code name for the feed, e.g. DFGS, HSDAG etc.

feed.documentation	A URL or hint where to find the documentation of this feed.
feed.name	Name for the feed, usually found in collector bot configuration.
feed.provider	Name for the provider of the feed, usually found in collector bot configuration.
feed.url	The URL of a given abuse feed, where applicable.
malware.hash.md5	A string depicting an MD5 checksum for a file, be it a malware sample for example.
malware.hash.sha1	A string depicting a SHA1 checksum for a file, be it a malware sample for example.
malware.hash.sha256	A string depicting a SHA256 checksum for a file, be it a malware sample for example.
malware.name	The malware name in lower case.
malware.version	A version string for an identified artifact generation, e.g. a crime-ware kit.
misp.attribute_uuid	MISP - Malware Information Sharing Platform & Threat Sharing UUID of an attribute.
misp.event_uuid	MISP - Malware Information Sharing Platform & Threat Sharing UUID.
output	Event data converted into foreign format, intended to be exported by output plugin.
protocol.application	e.g. vnc, ssh, sip, irc, http or smtp.
protocol.transport	e.g. tcp, udp, icmp.
raw	The original line of the event from encoded in base64.
rtir_id	Request Tracker Incident Response ticket id.
screenshot_url	Some source may report URLs related to a an image generated of a resource without any metadata. Or an URL pointing to resource, which has been rendered into a webshot, e.g. a PNG image and the relevant metadata related to its retrieval/generation.
source.abuse_contact	Abuse contact for source address. A comma separated list.
source.account	An account name or email address, which has been identified to relate to the source of an abuse event.
source.allocated	Allocation date corresponding to BGP prefix.
source.as_name	The autonomous system name from which the connection originated.
source.asn	The autonomous system number from which originated the connection.
source.domain_suffix	The suffix of the domain from the public suffix list.
source.fqdn	A DNS name related to the host from which the connection originated. DNS allows even binary data in DNS, so we have to allow everything. A final point is stripped, string is converted to lower case characters.
source.geolocation.cc	Country-Code according to ISO3166-1 alpha-2 for the source IP.

source.geolocation.city	Some geolocation services refer to city-level geolocation.
source.geolocation.country	The country name derived from the ISO3166 country code (assigned to cc field).
source.geolocation.cymru_cc	The country code denoted for the ip by the Team Cymru asn to ip mapping service.
source.geolocation.geoip_cc	MaxMind Country Code (ISO3166-1 alpha-2).
source.geolocation.latitude	Latitude coordinates derived from a geolocation service, such as MaxMind geoip db.
source.geolocation.longitude	Longitude coordinates derived from a geolocation service, such as MaxMind geoip db.
source.geolocation.region	Some geolocation services refer to region-level geolocation.
source.geolocation.state	Some geolocation services refer to state-level geolocation.
source.ip	The IP observed to initiate the connection.
source.local_hostname	Some sources report a internal hostname within a NAT related to the name configured for a compromised system.
source.local_ip	Some sources report a internal (NATed) IP address related a compromised system. N.B. RFC1918 IPs are OK here.
source.network	CIDR for an autonomous system. Also known as BGP prefix. If multiple values are possible, select the most specific.
source.port	The port from which the connection originated.
source.registry	The IP registry a given ip address is allocated by.
source.reverse_dns	Reverse DNS name acquired through a reverse DNS query on an IP address. N.B. Record types other than PTR records may also appear in the reverse DNS tree. Furthermore, unfortunately, there is no rule prohibiting people from writing anything in a PTR record. Even JavaScript will work. A final point is stripped, string is converted to lower case characters.
source.tor_node	If the source IP was a known tor node.
source.url	A URL denotes an IOC, which refers to a malicious resource, whose interpretation is defined by the abuse type. A URL with the abuse type phishing refers to a phishing resource.
source.urlpath	The path portion of an HTTP or related network request.
status	Status of the malicious resource (phishing, dropzone, etc), e.g. online, offline.
time.observation	The time the collector of the local instance processed (observed) the event.
time.source	The time of occurrence of the event as reported the feed (source).
tlp	Traffic Light Protocol level of the event.

Příloha C

Datový formát používaný službou Shodan

Následující tabulka obsahuje popis základních atributů datového formátu používaného službou Shodan. Je převzata z dokumentace služby Shodan[5] a ponechána s původními vysvětlivkami v anglickém jazyce, aby nedošlo k nepřesnostem v důsledku překladu. Dokumentace služby Shodan bohužel neuvádí vysvětlivky ke všem používaným atributům. Atributy jednotlivých protokolů a produktů, které služba Shodan zjišťuje, nejsou pro svůj rozsah v této příloze uvedeny.

`asn`	
`cpe`	CPE information in the old, deprecated format.
`cpe23`	CPE information in the 2.3 format.
`data`	
`device`	
`devicetype`	
`domains`	
`hash`	Numeric hash of the ”data”property which is helpful for finding other IPs with the exact same information.
`hostnames`	Hostnames for the IP based on the PTR/ reverse DNS information.
`html`	This property is deprecated. Use ”http.html”instead.
`ip`	Numeric IP address which can be more efficient for storing/ indexing.
`ip_str`	String representation of the IP address. This is most likely what you want to use.
`info`	
`ipv6`	
`isp`	
`link`	
`mac`	

`opts`	Stores experimental data before it has been finalized into a top-level property.
`org`	Name of the organization that manages the IP
`os`	Operating system
`platform`	
`port`	Yes
`product`	Name of the software that powers the service.
`tags`	array of Tag
`timestamp`	Date and time that the banner was collected in UTC time.
`title`	This property is deprecated. Use http.title instead.
`transport`	
`uptime`	
`vendor`	
`version`	
`vulns`	

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `thesis.pdf` ..... text práce ve formátu PDF
- `src`
  - `openapi.json` ..... kompletní popis systému dle specifikace OpenAPI
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `threatstate` ..... repozitář se zdrojovými kódy implementace
  - `pydantic-shodan` .. repozitář se zdrojovými kódy podpůrného modulu