VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

UŽIVATELSKÉ ROZHRANÍ SYSTÉMU PRO SPRÁVU ZAŘÍZENÍ V SÍTI

USER INTERFACE FOR ASSET MANAGEMENT SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DÁVID BENKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN ŽÁDNÍK, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

155232

Ústav: Ústav počítačových systémů (UPSY)
Student: **Benko Dávid**
Program: Informační technologie
Název: **Uživatelské rozhraní systému pro správu zařízení v síti**
Kategorie: Uživatelská rozhraní
Akademický rok: 2023/24

Zadání:

1. Seznamte se se systémem pro získávání informací o zařízeních ve vlastní síti, vyvíjeným ve sdružení CESNET.
2. Analyzujte potřeby a požadavky operátorů Security Operation Center na tento systém.
3. Pro systém navrhněte webové uživatelské rozhraní, které bude vhodným způsobem zobrazovat všechna data o zařízeních, na nich běžících službách, zjištěných zranitelnostech, a případně dalších entitách uložených v databázi systému, včetně jejich vzájemných vztahů. Rozhraní by dále mělo umožňovat vyhledávat entity podle zadaných kritérií a zobrazovat různá shrnutí či statistiky. Návrh průběžně konzultujte se členy CESNET-SOC a přizpůsobte ho jejich potřebám.
4. Navržené webové rozhraní implementujte, včetně případných potřebných úprav samotného asset management systému.
5. Ve spolupráci s vedoucím práce systém nasaďte a předejte k užívání týmu CESNET-SOC.
6. Diskutujte dosažené výsledky a možnosti budoucích úprav.

Literatura:

- Bartoš, Václav. (2019). NERD: Network Entity Reputation Database. ARES '19: Proceedings of the 14th International Conference on Availability, Reliability and Security. 1-7. 10.1145/3339252.3340512.
- Další literatura dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Žádník Martin, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 30.10.2023

Abstrakt

Táto práca sa zaoberá vývojom webového používateľského rozhrania systému Amfora pre správu zariadení v sieti, ktorý koncentruje údaje o IP adresách, službách a zraniteľnostiach z 9 systémov. Rozhranie je založené na trojvrstvovej architektúre s využitím REST API. Na implementáciu klientskej časti bol využitý framework Vue.js (jazyk TypeScript/JavaScript) a na implementáciu serverovej časti framework FastAPI (jazyk Python). Systém bol nasadený do siete CESNET a daný do užívania bezpečnostnému operačnému stredisku (SOC). Podarilo sa dosiahnuť výrazného zjednodušenia práce pri riešení bezpečnostných incidentov a monitoringu siete.

Abstract

This thesis focuses on the development of a web user interface for an asset management system called Amfora. Amfora gathers data regarding IP addresses, services and vulnerabilities from 9 other systems. The user interface is based on three tier architecture utilizing REST API. Frontend implementation uses the Vue.js framework (TypeScript/JavaScript language) and backend implementation uses the FastAPI framework (Python language). The system has been deployed into production use in the CESNET network and presented to the security operations center team in CESNET. Amfora has significantly simplified work in solving security incidents and network monitoring.

Kľúčové slová

Amfora, správa zariadení, webová aplikácia, REST API, DP^3, Vue.js, FastAPI, IP adresa, služba, bezpečnostné operačné stredisko, DNS, SNER, Shodan, ShadowServer, ADiCT, Auror, Mentat, ExaFS, PassiveDNS, OTRS, NERD, NetBox, FTAS

Keywords

Amfora, asset management, web application, REST API, DP^3, Vue.js, FastAPI, IP address, service, Security Operations Center, DNS, SNER, Shodan, ShadowServer, ADiCT, Auror, Mentat, ExaFS, PassiveDNS, OTRS, NERD, NetBox, FTAS

Citácia

BENKO, Dávid. *Uživatelské rozhraní systému pro správu zařízení v síti.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Žádník, Ph.D.

Uživatelské rozhraní systému pro správu zařízení v síti

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Martina Žádníka, Ph.D. Ďalšie informácie mi poskytol odborný konzultant Ing. Václav Bartoš, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

. . . . . .
Dávid Benko
6. mája 2024

Poďakovanie

Chcel by som sa poďakovať vedúcemu mojej práce, Ing. Martinovi Žádníkovi, Ph.D. a konzultantovi Ing. Václavovi Bartošovi, Ph.D. zo združenia CESNET za ich usmernenia, priebežné konzulácie a rady pri písaní práce. Rovnako by som sa chcel poďakovať pracovníkom bezpečnostného operačného strediska v združení CESNET — predovšetkým Ing. et Ing. Radkovi Krkošovi, Ph.D. — za spoluprácu pri špecifikácii požiadaviek a testovaní.

Obsah

1 Úvod **4**

2 Súčasný stav a služby **5**
2.1 Existujúce systémy využívané pri správe zariadení v sieti CESNET 5
2.1.1 SNER 6
2.1.2 ADiCT 7
2.1.3 Auror 8
2.1.4 Mentat 9
2.1.5 NERD 10
2.1.6 PassiveDNS 10
2.1.7 ExaFS 10
2.1.8 OTRS 12
2.1.9 FTAS 12
2.1.10 Shodan 13
2.1.11 ShadowServer 14
2.1.12 NetBox 14
2.2 Platforma DP^3 15
2.2.1 Dátový model platformy DP^3 15
2.2.2 Architektúra platformy DP^3 17
2.2.3 Spracovanie dát v platforme DP^3 18
2.3 Metódy implementácie moderných webových rozhraní 18
2.3.1 Frontend moderných webových rozhraní 19
2.3.2 Backend moderných webových rozhraní 21

3 Návrh systému Amfora **24**
3.1 Základ systému Amfora 24
3.2 Demo verzia používateľského rozhrania 25
3.3 Analýza potrieb bezpečnostného operačného strediska CESNET 28
3.4 Zhrnutie požiadaviek 29
3.5 Mockup používateľského rozhrania 29
3.6 Návrh architektúry systému Amfora 31

4 Implementácia serverovej časti webového rozhrania **33**

5 Implementácia klientskej časti webového rozhrania **40**
5.1 Popis zdrojového kódu klientskej časti webového rozhrania 40
5.2 Výsledné používateľské rozhranie 43

6 Nasadenie **49**

7 Testovanie **51**
7.1 Testovanie pomocou automatických testov 51
7.2 Používateľské testovanie 52
7.3 Testovanie responzivity na mobilných zariadeniach 52
7.4 Testovanie autentifikácie a autorizácie 52
7.5 Možné budúce rozšírenia 53

8 Záver **55**

Literatúra **56**

A Obsah priloženého pamäťového média **59**

Zoznam obrázkov

2.1 Architektúra systému SNER 7
2.2 Vysokoúrovňový pohľad na systém ADiCT 8
2.3 Architektúra systému Mentat 9
2.4 Princíp zberu dát za DNS resolverom 11
2.5 Architektúra systému ExaFS 12
2.6 Príklad sieťovej aktivity niekoľkých tokov v systéme FTAS 13
2.7 Používateľské rozhranie NetBox 14
2.8 Hodnota atribútu typu pozorovanie pri prekrývajúcich sa platnostiach dátových bodov v platforme DP^3 16
2.9 Architektúra platformy DP^3 18
2.10 Spracovanie prijatých dát vo worker procese platformy DP^3 19
2.11 Najpopulárnejšie webové frameworky jazyka Python v roku 2023 22

3.1 Prehľad IP adries v demo verzii používateľského rozhrania systému Amfora 26
3.2 Detail služby v demo verzii používateľského rozhrania systému Amfora . . . 27
3.3 Mockup používateľského rozhrania systému Amfora 30
3.4 Architektúra systému Amfora 32

4.1 Ukážka dokumentácie API rozhrania systému Amfora 36

5.1 Ukážka komponentu `DataWithSrc.vue` 41
5.2 Ukážka prehľadu IP adries v používateľskom rozhraní systému Amfora . . . 44
5.3 Ukážka prehľadu skupín IP adries v použ. rozhraní systému Amfora 45
5.4 Ukážka prehľadu služieb v používateľskom rozhraní systému Amfora 46
5.5 Ukážka detailu IP adresy v používateľskom rozhraní systému Amfora 47
5.6 Ukážka detailu služby v používateľskom rozhraní systému Amfora 48

6.1 Schéma nasadeného systému Amfora 49

7.1 Zobrazenie detailu IP adresy na mobilnom telefóne 53

Kapitola 1

Úvod

Komplexnosť počítačových sietí každoročne narastá, ako narastá aj počet používateľov, typov zariadení pripojených a komunikujúcich v rámci siete internet či rôznorodosť služieb, ktoré sa cez sieť poskytujú. Zároveň sa zvyšuje dôležitosť a nenahraditeľnosť elektronických komunikačných technológií v každodennom živote ľudí. Hoci si to väčšina používateľov neuvedomuje, bezpečnosť počítačových sietí sa vďaka tomu stáva čoraz náročnejším problémom. Obzvlášť náročný je monitoring a riešenie bezpečnostných incidentov v sieťach veľkých organizácií či univerzít.

Táto práca je súčasťou vývoja systému Amfora pre združenie CESNET. Úlohou systému Amfora je zjednodušenie práce operátorov bezpečnostného operačného strediska (SOC) pri monitoringu a riešení bezpečnostných incidentov v sieti CESNET. Zjednodušenie práce spočíva vo vylepšení spôsobu správy zariadení v sieti. Systém Amfora agreguje dáta o IP adresách, službách a zraniteľnostiach zo 7 existujúcich interných systémov (SNER, ADiCT, Mentat, PassiveDNS a ďalšie) a 2 externých systémov (Shodan, ShadowServer). Poskytuje unifikovaný prehľad o zariadeniach v sieti na jednom mieste, rieši problém náročného vyhľadávania a absencie informácií z externých zdrojov. Navyše implementuje podporu pre detekciu anomálií vo forme porovnávania reálneho stavu oproti definovanému.

Práca sa venuje analýze, návrhu, implementácii a nasadeniu (primárne) používateľského rozhrania systému Amfora. Keďže používateľské rozhranie nie je možné úplne oddeliť od zvyšku systému a vývoj prebiehal súčasne, spolupodieľal som sa aj na návrhu systému Amfora ako celku a na vývoji ostatných častí.

Práca je členená na viacero kapitol, ktoré postupne sprevádzajú celým procesom vývoja. Kapitola 2 sa venuje analýze súčasného stavu, aktuálne používaným systémom pri práci operátorov oddelenia SOC (sekcia 2.1), platforme DP^3 (sekcia 2.2) a technológiám používaným na vývoj moderných webových rozhraní (sekcia 2.3). V kapitole 3 je zhodnotený súčasný stav a adresovaný návrh systému Amfora, primárne používateľského rozhrania. Vysvetlený je celý proces návrhu od vytvorenia úvodnej demo verzie webového rozhrania a analýzy potrieb oddelenia CESNET SOC, až po vytvorenie mockup návrhu a návrhu architektúry. Následne je tento návrh implementovaný – v kapitole 4 serverová časť a v kapitole 5 klientská časť webového rozhrania. Webové rozhranie, rovnako ako celý systém Amfora, bolo nasadené do prevádzky v združení CESNET, čo je opísané v kapitole 6. V kapitole 7 je opísaný priebeh automatického testovania, testovania zabezpečenia a používateľského testovania na operátoroch oddelenia SOC, na základe ktorého boli vykonané úpravy a určené možnosti ďalšieho rozšírenia webového rozhrania systému Amfora.

Kapitola 2

Súčasný stav a služby

Táto kapitola je zhrnutím aktuálneho spôsobu práce operátorov oddelenia CESNET SOC[1] a nástrojov, ktoré pri práci používajú (sekcia 2.1). Následne sa zameriava na popis platformy DP[3], na ktorej bude systém Amfora založený (sekcia 2.2) a metódami implementácie webových rozhraní na klientskej aj serverovej strane (sekcia 2.3).

Združenie CESNET[2] je národným správcom e-infraštruktúry pre vedu, výskum a vzdelávanie v Českej republike. Bolo založené v roku 1996 vysokými školami a Akadémiou vied ČR. Dlhodobo prepája svojich členov vysokorýchlostnou sieťou, poskytuje prístup do internetu a do roamingovej služby *eduroam*, poskytuje prostriedky pre ukladanie a zdieľanie dát či výkonný hardware pre náročné výpočty a venuje sa aj vlastným výskumným aktivitám. [28]

V združení CESNET je vytvorené oddelenie SOC, ktoré zaisťuje dohľad na e-infraštruktúru CESNET, riešenie bezpečnostných incidentov a prevádzku CSIRT[3] tímu CESNET CERTS. [17]

Pre operátorov oddelenia CESNET SOC sú preto dôležité aktuálne, overené a čo možno najpodrobnejšie informácie o stave prevádzkovaných služieb. Keďže komunikácia prebieha po IP sieti, základom sú údaje, ktoré sa viažu k jednotlivým IP adresám a portom.

2.1 Existujúce systémy využívané pri správe zariadení v sieti CESNET

V rámci zberu požiadaviek na výsledný systém som sa zúčastnil niekoľkých diskusií so zástupcami oddelenia CESNET SOC a následne som vykonal pozorovanie práce operátorov SOC metódou *shadowing*. Celkovo som pozoroval prácu piatich operátorov. Získané poznatky spočívajú predovšetkým v používaných nástrojoch a postupoch, ktoré sú v krátkosti predstavené v nasledujúcich odstavcoch a podrobnejšie v podsekciách nižšie.

Oddelenie CESNET SOC využíva pri práci viacero interných systémov:

- **SNER** (podsekcia 2.1.1): enumeratívny skener siete (IP adresy, služby, verzie, zraniteľnosti)
- **ADiCT** (podsekcia 2.1.2): zhromažďuje a odvodzuje informácie o zariadeniach v sieti

[1] SOC: Security Operations Center (bezpečnostné operačné stredisko)
[2] CESNET: https://www.cesnet.cz/
[3] CSIRT: Computer Security Incident Response Team (tím reakcie na incidenty počítačovej bezpečnosti)

- **Auror** (podsekcia 2.1.3): skener TLS[4], HTTP[5] a ďalších protokolov
- **Mentat** (podsekcia 2.1.4): databáza bezpečnostných udalostí z rôznych zdrojov
- **NERD** (podsekcia 2.1.5): databáza škodlivých IP adries v celom internete
- **PassiveDNS** (podsekcia 2.1.6): databáza histórie DNS[6] záznamov
- **ExaFS** (podsekcia 2.1.7): nástroj na filtrovanie prevádzky v sieti pomocou ExaBGP[7] správ
- **OTRS** (podsekcia 2.1.8): systém pre správu procesu riešenia incidentov (angl. ticketing system)
- **FTAS** (podsekcia 2.1.9): monitoring sieťových tokov

Využité sú taktiež externé služby:

- **Shodan** (podsekcia 2.1.10): skener celého internetu (IP, služby)
- **ShadowServer** (podsekcia 2.1.11): skener celého internetu a databáza hrozieb v internete (botnety, DDoS, zraniteľnosti)

Operátor typicky potrebuje pri práci využívať viacero spomínaných systémov súčasne, pretože iba tak dokáže získať kompletný prehľad o situácii, ktorú rieši. Práca operátora preto často spočíva v manuálnom vyhľadávaní a porovnávaní údajov medzi niekoľkými oknami webového prehliadača.

Nasledujúce podsekcie bližšie opisujú systémy spomínané vyššie.

2.1.1 SNER

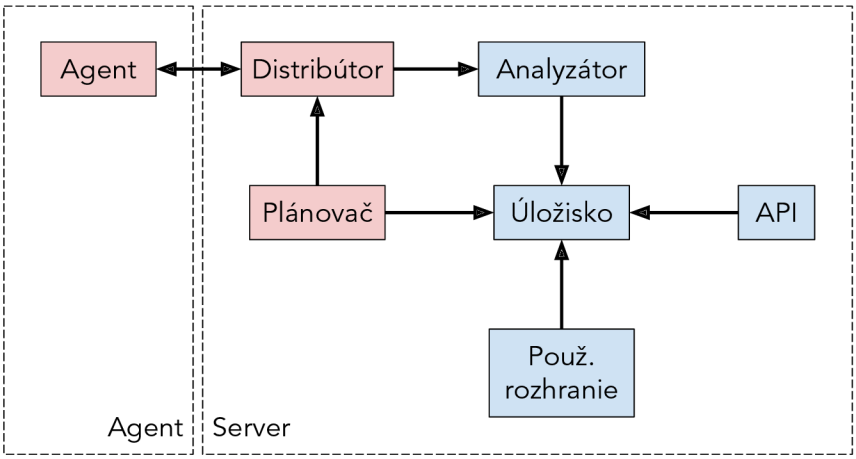
Systém SNER (Slow Network Recon Service) vykonáva enumeratívne skenovanie špecifikovaných sietí pre detekciu zariadení, služieb, ich verzií a zraniteľností. Vyvíja ho tím FLAB[8] združenia CESNET ako open-source projekt[9]. [9]

Systém SNER sa skladá zo 7 komponentov, ako ilustruje diagram 2.1. [10]

- **Agent** (agent): vykonáva skenovanie prostredníctvom jedného z pluginov (`nc`, `nmap`, TestSSL a ďalšie)
- **Distribútor** (scheduler): distribuuje záťaž medzi agentov
- **Plánovač** (planner): plánuje skenovanie, aby uložené dáta boli vždy dostatočne aktuálne
- **Analyzátor** (parser): spracováva dáta prichádzaujúce z agentov a ukladá ich
- **Úložisko** (storage): ukladá dáta do PostgreSQL[10] databázy (alebo do súborov)

[4] TLS: Transport Layer Security
[5] HTTP: Hypertext Transfer Protocol
[6] DNS: Domain Name System
[7] BGP: Border Gateway Protocol
[8] FLAB: forenzné laboratórium
[9] https://github.com/bodik/sner4
[10] PostgreSQL: https://www.postgresql.org/

- **Používateľské rozhranie** (UI): umožňuje používateľovi zobraziť zoznam zariadení, služieb a zraniteľností, ich detail a pridanie manuálnych poznámok
- **API**: rozhranie pre strojový prístup k uloženým dátam a na prepojenie s inými systémami



Obr. 2.1: **Architektúra systému SNER.** Červenou farbou sú vyznačené skenovacie komponenty, modrou farbou sú vyznačené dátové komponenty. Vlastná tvorba, vychádza z [10].

Pravdepodobne najdôležitejšou funkciou pre operátorov oddelenia SOC je vyhľadávanie software na zariadeniach podľa verzie. Typicky sa tak deje v reakcii na objavenie novej zraniteľnosti, po ktorej je potrebné príslušný software na zasiahnutých zariadeniach aktualizovať. Na vyhľadávanie využívajú webové rozhranie SNER.

2.1.2 ADiCT

Informácie v tejto podsekcii pochádzajú z bakalárskej práce Ondřeja Sedláčka [24], z interných wiki stránok projektu ADiCT[11] a z privátneho Github repozitára projektu[12].

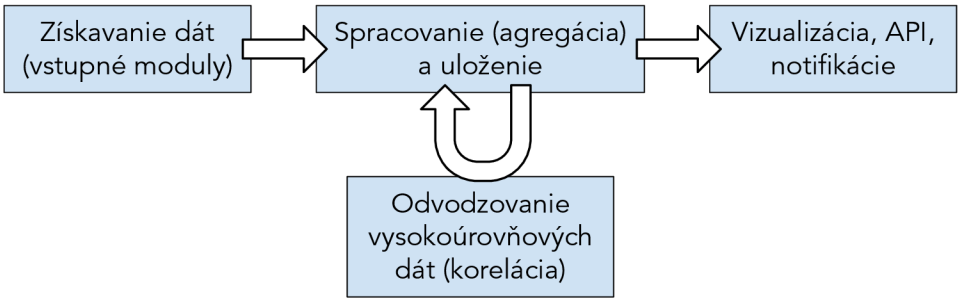
Projekt ADiCT (Asset Discovery, Classification and Tagging) slúži na automatické zhromažďovanie údajov o zariadeniach vo vlastnej sieti (v tomto prípade v sieti CESNET). V systéme ADiCT sa aktuálne sledujú IP adresy, MAC adresy a podsiete (angl. subnet).

Na rozdiel od iných systémov spomínaných v tejto bakalárskej práci (SNER, Auror, Shodan, ShadowServer), ktoré aktívne skenujú sieť, v projekte ADiCT ide predovšetkým o zber dát pasívnym monitorovaním prevádzky a odosielaním požiadaviek na externé systémy.

Keďže množstvo takto získaných dát je značné, ich interpretácia je netriviálna – obzvlášť, ak medzi dvoma zdrojmi existuje rozpor. Z tohto dôvodu ďalej v systéme ADiCT prebieha automatické odvodzovanie informácií, napríklad prostredníctvom metód fúzie dát navrhnutých v bakalárskej práci Ondřeja Sedláčka [24].

Na obrázku 2.2 je znázornený tok dát. ADiCT pracuje s dvoma typmi modulov – primárnymi a sekundárnymi. Získavanie dát realizujú primárne (vstupné) moduly prostredníctvom

[11] https://redmine.liberouter.org/projects/adict/wiki
[12] https://github.com/CESNET/ADiCT



Obr. 2.2: **Vysokoúrovňový pohľad na systém ADiCT.** Prevzaté z obrázku 2.1 v [24].

HTTP API rozhrania. V súčasnosti obsahuje projekt ADiCT 25 vstupných modulov[13], medzi ktoré patrí napríklad monitor aktivity IP adresy (`ip_activity`), modul mapujúci IP adresy na MAC adresy (`ip_mac`), detektor operačného systému podľa TCP/IP hlavičiek (`os_by_tcp`) či modul sťahujúci denné hlásenia zo služby ShadowServer[14] (`shadowserver`). Systém následne prijaté dáta ukladá do databázy a ďalej spracúva prostredníctvom sekundárnych (odvodzovacích) modulov. Spúšťajú sa v reakcii na príchod určitého typu dát, v reakcii na udalosť v systéme alebo periodicky. Sekundárne moduly slúžia na odvodzovanie vysokoúrovňových informácií alebo zisťovanie ďalších dát z ďalších zdrojov. Ku všetkým informáciám obsiahnutým v systéme ADiCT je možné pristupovať prostredníctvom HTTP API a webového rozhrania.

Projekt ADiCT je založený na platforme DP^3, ktorá bude podrobne opísaná v sekcii 2.2. DP^3 je všeobecná databáza entít umožňujúca príjem a odvodzovanie dát na základe primárnych a sekundárnych modulov, a bola vyvinutá združením CESNET práve pre projekt ADiCT. Od augusta 2023 využíva ADiCT novú verziu platformy DP^3, z ktorej sa stal open-source projekt[15].

V čase písania tejto práce je ADiCT chápaný viac experimentálne ako produkčne. Jeho vývoj stále aktívne prebieha s cieľom zistiť, čo všetko a s akou presnosťou je možné zistiť a odvodiť z dostupných zdrojov.

2.1.3 Auror

Táto podkapitola vychádza z dostupných interných informácií o systéme Auror[16] a z jedinej verejnej referencie[17] na tento systém, ktorú bolo možné dohľadať v čase písania tejto práce.

Auror je systém pre sieťové skenovanie zabezpečenia webových aplikácií. Ide o projekt interne vyvíjaný a nasadený v združení CESNET. Pre prístup k dátam poskytuje webové používateľské rozhranie a webové API rozhranie. Na skenovanie využíva niekoľko nástrojov:

- **nmap**[18]: známy generický skenovací nástroj
- **MASSCAN**[19]: rýchly skener pre protokol TCP (Transport Control Protocol)

[13] https://github.com/CESNET/ADiCT/tree/4d7906f1316ffe044a8fc25fe90fb47f9074ccd7/input_modules (privátny repozitár)
[14] ShadowServer: https://www.shadowserver.org/
[15] DP^3: https://github.com/CESNET/dp3
[16] https://auror.cesnet.cz/ a ďalšie
[17] https://csirt.cesnet.cz/en/auror-scanner
[18] nmap: https://nmap.org/
[19] MASSCAN: https://github.com/robertdavidgraham/masscan

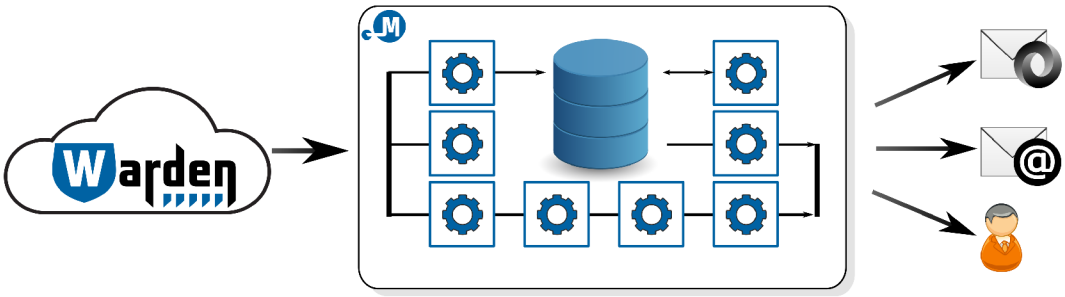
- **testssl.sh**[20]: nástroj pre príkazový riadok overujúci podporu SSL/TLS (Secure Sockets Layer/Transport Layer Security) sieťovej služby
- **DNSViz**[21]: sada nástrojov pre analýzu systému DNS (Domain Name System)

Pomocou nich zisťuje hodnoty HTTP hlavičiek, ponúkané verzie protokolu SSL/TLS, ponúkané šifrovacie metódy SSL/TLS, prípadné zraniteľnosti, validitu certifikátov a simuluje rôzne typy klientských zariadení. Získané informácie následne vyhodnocuje a v prípade protokolu SSL/TLS priraďuje službe hodnotenie (známku) `A+`, `A`, `A-`, `B`, `M` alebo `T`. Známky `A+`, `A`, `A-` a `B` sú sémanticky podobné hodnoteniu v škole, `M` vyjadruje, že doménové meno v certifikáte nezodpovedá doménovému menu skenovaného servera a `T` upozorňuje, že certifikát nie je dôveryhodný.

Pre operátorov oddelenia CESNET SOC je cenné vyhodnotenie SSL/TLS, ktoré môže poukázať na zraniteľnosti a problémy pri nasadení, ktoré je potrebné vyriešiť.

2.1.4 Mentat

Systém Mentat je databázou bezpečnostných udalostí predovšetkým zo sieťových sond, honeypotov a služieb tretích strán, ktoré nahlasujú zraniteľnosti, útoky, spam, vírusy či zdieľanie údajov porušujúcich autorské práva. Taktiež obsahuje databázu podsietí a kontaktov na príslušných administrátorov, ktorým automaticky odosiela hlásenia o detekovaných problémoch v ich sieťach. Je vyvíjaný oddelením CESNET SOC ako open-source projekt[22]. [5]



Obr. 2.3: **Architektúra systému Mentat.** Prevzaté z [5].

Mentat je nasadený a využívaný ako SIEM (Security Information and Event Management System) siete CESNET. Využíva množstvo interných a externých zdrojov, medzi ktoré patria rôzne druhy honeypotov a detektory anomálií sieťovej prevádzky, DoS (denial of service) a DDoS (distributed denial of service) útokov (napríklad FTAS[23] a Nemea[24]). [6]

Jadro systému Mentat využíva na komunikáciu a získavanie dát systém Warden. Ide o systém na zdieľanie informácií od viacerých producentov viacerým príjemcom s podporou pre základné filtrovanie [20]. Mentat teda predstavuje jedného z konzumentov (prijímajúcich klientov). Na prenos údajov sa využíva formát IDEA (Intrusion Detection Extensible Alert)[25], ktorý bol navrhnutý práve pre zber detailov bezpečnostných udalostí z rôznorodých zdrojov. IDEA je založená na formáte JSON (JavaScript Object Notation), ktorý rozširuje o definíciu štruktúry a sémantiku jednotlivých položiek. [19]

[20] testssl.sh: https://testssl.sh/
[21] DNSViz: https://github.com/dnsviz/dnsviz
[22] https://homeproj.cesnet.cz/projects/mentat/repository
[23] FTAS: opísaný v podsekcii 2.1.9
[24] Nemea: https://www.liberouter.org/technologies/nemea/
[25] IDEA: https://idea.cesnet.cz/en/index

Pre oddelenie SOC je webové rozhranie systému Mentat miestom, kde môžu zisťovať informácie o automaticky detekovaných bezpečnostných udalostiach a rozosielaných hláseniach.

2.1.5 NERD

Systém NERD[26] (Network Entity Reputation Database) je databázou škodlivých IP adries v sieti internet. Opäť ide o projekt vyvíjaný interne v združení CESNET. Okrem IP adries NERD obsahuje aj základné informácie o BGP (Border Gateway Protocol) prefixoch, ASN (Autonomous System Number), blokoch IP adries a organizáciách. Škodlivosť IP adries sa určuje na základe reputačného skóre. Ide o desatinné číslo medzi v intervale $\langle 0, 1 \rangle$, pričom hodnota 1 reprezentuje najvyššiu mieru škodlivosti. Výpočet reputačného skóre prebieha na základe údajov z rôznych zdrojov – predovšetkým ide o hlásenia bezpečnostných udalostí (spam, DoS útoky, skenovanie) zo systému Warden, ktorý je bližšie opísaný v podsekcii 2.1.4, prítomnosť na niektorom z verejných blacklistov, IDS (Intrusion Detection System) údaje DShield[27] a prítomnosť v hláseniach platformy MISP[28]. [8]

Vysokoúrovňový pohľad na princíp a architektúru systému NERD je podobný systému ADiCT, ktorý je opísaný v podsekcii 2.1.2. Na rozdiel od systému ADiCT, NERD nevyužíva platformu DP^3 (sekcia 2.2), avšak jeho reimplementácia nad túto platformu v čase písania tejto bakalárskej práce prebieha.

2.1.6 PassiveDNS

Označenie PassiveDNS je možné chápať dvoma spôsobmi. Prvým je PassiveDNS ako samotný prístup k ukladaniu zoznamu vykonaných zmien v DNS záznamoch, ktorý podľa Maxmiliána Tomáša [30] navrhol ako prvý Florian Weimer v roku 2005 [32].

Weimer vo svojej práci navrhol vykonávanie pasívneho zachytávania dátovej prevádzky DNS (Domain Name System) a následného ukladania DNS záznamov, ktoré boli zachytené, mimo systém DNS rezolúcie. Išlo o spôsob pre detekciu ukradnutia identity internetovej služby či obnovu už neexistujúcich DNS zón.

Na tomto základe vznikla v združení CESNET myšlienka pre vývoj vlastnej implementácie PassiveDNS systému. Prvotne sa tak udialo v rámci bakalárskej práce Maxmiliána Tomáša [30], ktorý implementoval a nasadil systém PassiveDNS pre sieť CESNET. Neskôr bol PassiveDNS výrazne upravovaný a rozširovaný až do súčasnej podoby. Názov tejto podkapitoly je referenciou práve na túto službu.

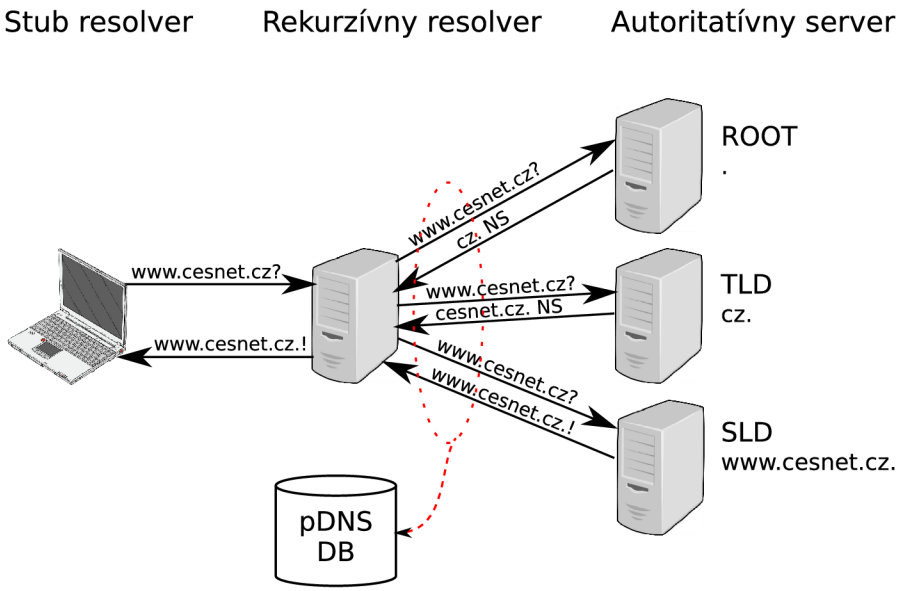
Systém PassiveDNS prijímané DNS záznamy z dôvodu náročnosti na množstvo úložiska agreguje. Formát uloženia záznamov ilustruje tabuľka 2.1. Pre prístup k uloženým údajom je možné využiť implementované webové používateľské rozhranie a webové API rozhranie. [30]

Prakticky má pre oddelenie CESNET SOC systém PassiveDNS význam napríklad pri odhaľovaní phishingových domén a dohľadávaní informácií pri riešení bezpečnostných incidentov.

2.1.7 ExaFS

ExaFS zabezpečuje odosielanie BGP (Border Gateway Protocol) správ. Tento systém umožňuje vykonávanie dynamických zmien konfigurácie siete (firewallu). Ide napríklad o nastave-

[26] https://nerd.cesnet.cz/
[27] DShield: https://www.dshield.org/
[28] MISP: https://www.misp-project.org/



Obr. 2.4: **Princíp zberu dát za DNS resolverom.** DNS tranzakcie sa replikujú do PassiveDNS (pDNS) databázy. Prevzaté z [11] podľa licencie *Creative Commons Uveďte autora 3.0* s preložením textu a úpravou fontu.

Názov atribútu	Význam atribútu
`count`	Počet výskytov (zachytených paketov) dvojice požiadavka/odpoveď
`request`	Dáta požiadavky
`response`	Dáta odpovede
`time_first`	Časový údaj o prvom zachytení
`time_last`	Časový údaj o poslednom zachytení
`type`	Typ DNS záznamu (A, AAAA alebo CNAME)

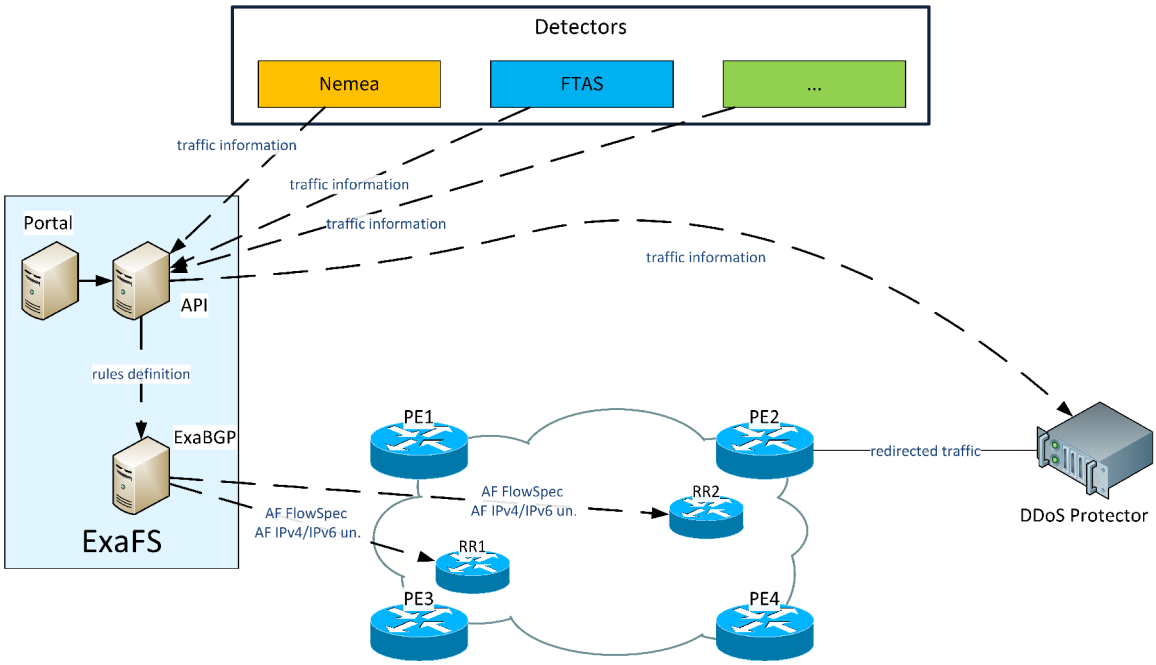
Tabuľka 2.1: **Štruktúra záznamu PassiveDNS.** Prevzaté z [30].

nie QoS (quality of service), zahadzovania či presmerovania paketov. Účelom je predovšetkým mitigácia DoS (denial of service) a DDoS (distributed denial of service) útokov alebo iných typov zneužitia služieb. Na rozdiel od priameho vytvárania BGP správ, ExaFS pridáva vrstvu pre autorizáciu používateľov, validáciu jednotlivých príkazov, ukladanie histórie a zotavenie po chybe alebo výpadku. [31]

Architektúra systému ExaFS je ilustrovaná na obrázku 2.5. Cez webové API rozhranie do ExaFS prichádzajú údaje o sieťovej prevádzke zo sieťových detektorov (Nemea[29], FTAS[30] a ďalšie). ExaFS požiadavky validuje a spracúva pomocou knižnice ExaBGP, ktorá vytvára BGP správy a odosiela ich routerom. Údaje o sieťovej prevádzke sú taktiež odosielané DDoS čističke, ktorá poskytuje štatistiky o mitigovanom útoku. ExaFS implementuje aj webové používateľské rozhranie. [21]

Pre oddelenie CESNET SOC je dôležitá možnosť zadávať nové pravidla a zobraziť existujúce pravidlá aplikované na určitú IP adresu (vrátane ich histórie).

[29]Nemea: https://www.liberouter.org/technologies/nemea/
[30]FTAS: opísaný v podsekcii 2.1.9



Obr. 2.5: **Architektúra systému ExaFS.** Prevzaté z [31].

2.1.8 OTRS

OTRS (Open source Ticket Request System) je systém pre riadenie procesu riešenia bezpečnostných incidentov (incident ticket system). Principiálne ide o nadstavbu nad emailovým klientom, ktorá umožňuje kolaboráciu viacerých osôb, pridávanie dodatočných informácií a sledovanie stavu štítkov. Štítok je vlákno emailov a každá správa v komunikácii pri riešení incidentov je reprezentovaná emailovou správou.

Združenie CESNET využíva tento systém pri riešení nahlasovaných incidentov z externého prostredia. Dôvody pre výber OTRS opisuje vo svojej technickej správe Pavel Kácha [18]. Medzi požiadavky patrila podpora pre emailovú komunikáciu so spoľahlivým udržiavaním kontextu (identifikátora štítku) a možnosť deliť a spájať štítky. Tieto požiadavky boli splnené v základnom stave. Viacero ďalších požiadaviek bolo možné splniť vlastnými rozšíreniami systému OTRS – pridaním automatickej extrakcie metadát zo správy incidentu (IP adresy, identifikátory relevantných sietí), výpočtom štatistík či overovaním časových limitov.

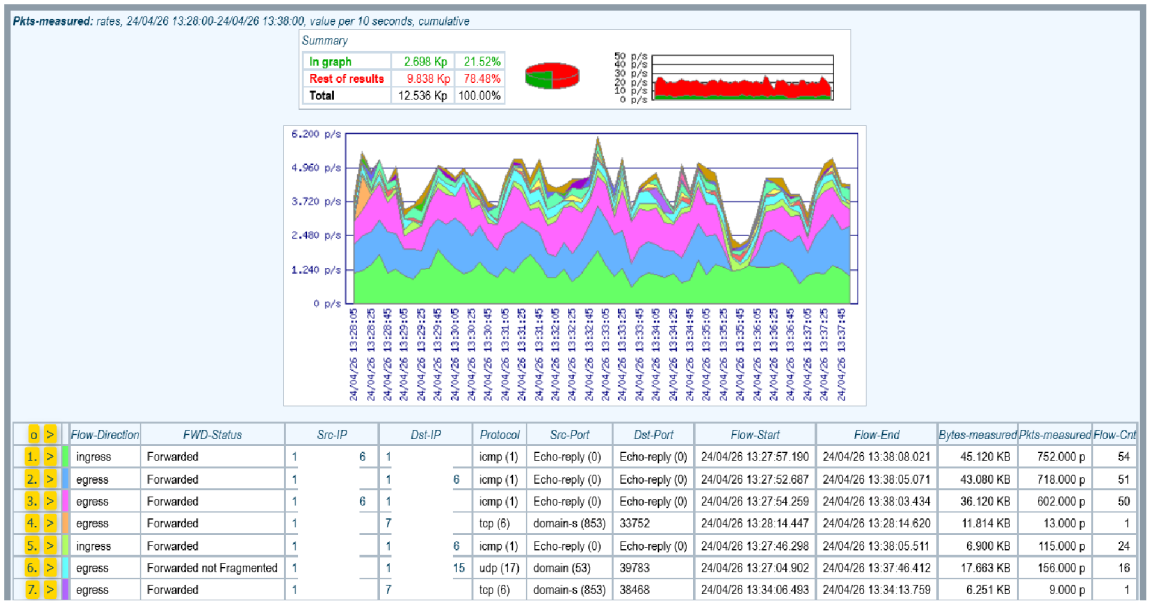
Aktuálne je OTRS výrazne prispôsobenou, vyladenou a dlhodobo fungujúcou službou. Pre oddelenie CESNET SOC ide o jeden z najdôležitejších systémov. Práve kvôli výrazne prispôsobenej inštalácii sú však aktualizácie problematické a stále sa používa verzia 3 z roku 2011. To zároveň sťažuje možnosti prepojenia s inými systémami.

2.1.9 FTAS

FTAS (Flow-Based Traffic Analysis System) je systém pre monitorovanie a analýzu sieťovej prevádzky. Jeho primárnym zdrojom sú dáta o sieťových tokoch (NetFlow). Toky následne umožňuje vzorkovať, klasifikovať, filtrovať, vyhľadávať, vizualizovať, vytvárať z nich štatistiky či na základe pravidiel nahlasovať (detekovať anomálie). Ide o komplexný systém vyvíjaný v združení CESNET už 20 rokov. Architektúra FTAS je značne flexibilná

a umožňuje rozdelenie systému na niekoľko fyzických strojov kvôli zvýšeniu priepustnosti a prispôsobenie veľkosti príslušnej siete. [16]

Význam systému FTAS pre operátorov oddelenia CESNET SOC je bezpochyby veľký, pretože umožňuje dohľadať detaily akejkoľvek sieťovej aktivity na úrovni sieťových tokov a zistiť bližšie informácie ku akémukoľvek typu bezpečnostného incidentu v sieti. Navyše umožňuje vytváranie hlásení po prekročení nastavených limitov (počet spojení, tokov, kombinácia rôznych pravidiel pre detekciu skenovania) do systému Warden, ktorý je bližšie opísaný v podsekcii 2.1.4. Obrázok 2.6 zobrazuje príklad prehľadu sieťovej aktivity v určitom časovom intervale.



Obr. 2.6: **Príklad sieťovej aktivity niekoľkých tokov v systéme FTAS.** IP adresy anonymizované.

2.1.10 Shodan

Shodan[31] je vyhľadávač zariadení v sieti internet. Umožňuje vyhľadávať zariadenia podľa služby, ktorá na nich beží, doménového mena, štátu, verzií rôznych protokolov, čísel portov, operačného systému, zraniteľností a podobne.

Shodan získava tieto informácie pomocou skenovania IP adries v sieti internet. Väčšina dát pochádza z tzv. banerov, ktoré obsahujú metadáta o bežiacej službe na danom zariadení. Tieto informácie majú využitie nielen v oblasti bezpečnosti, ale umožňujú sledovať aj podiel produktov na trhu či rozširovanie IoT (Internet of things) zariadení. [22]

Pre CESNET SOC nejde o primárny zdroj dát, ktorý využívajú operátori pri práci. Vo vlastnej sieti CESNET prebieha skenovanie vlastnými službami (SNER, Auror), preto sú údaje zo služby Shodan väčšinou menej aktuálne a redundantné. Hodia sa však pre zistenie detailov IP adresy mimo adresné rozsahy spravované združením CESNET alebo pre potvrdenie zistení spomínaných vlastných služieb.

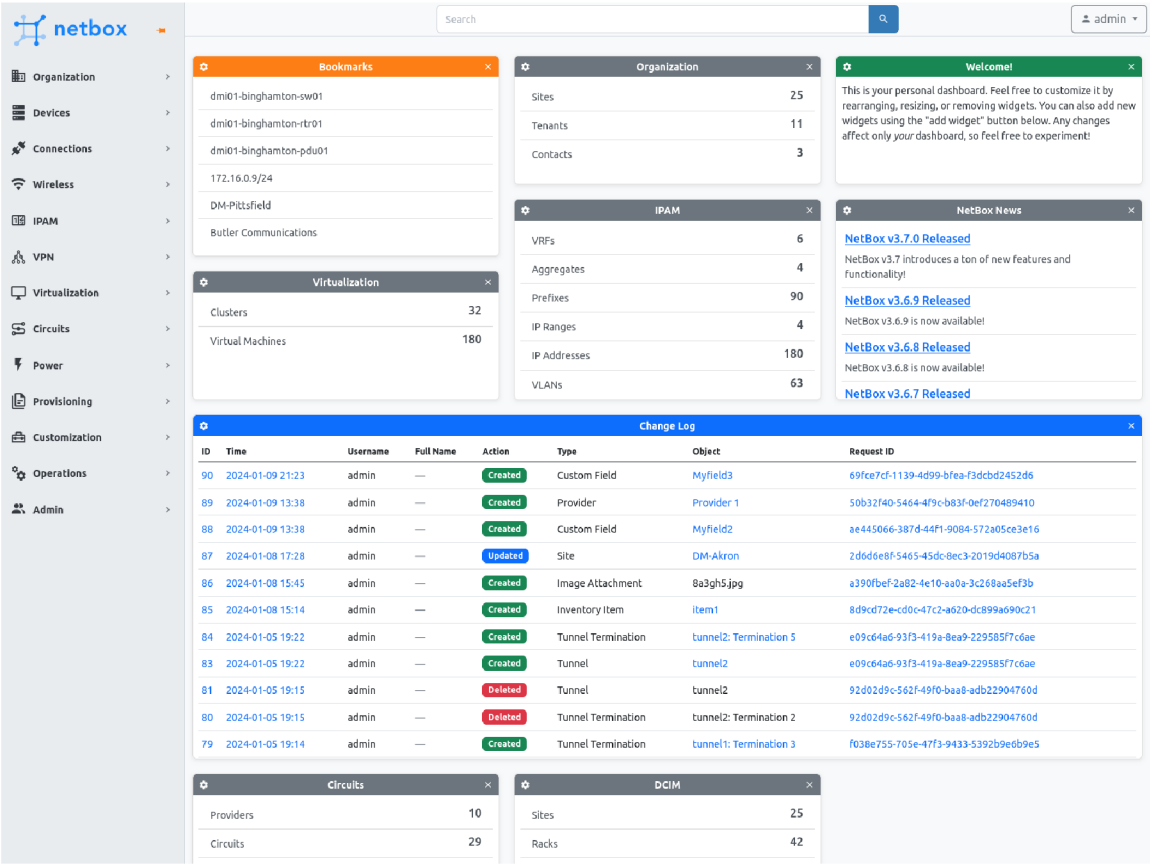
[31] https://www.shodan.io/

2.1.11 ShadowServer

Shadowserver Foundation[32] je nezisková organizácia zaoberajúca sa vyhľadávaním bezpečnostných hrozieb na internete. Denne skenuje a analyzuje miliardy IP adries a prevádzkuje množstvo honeypotov a sinkhole[33] serverov. Týmito spôsobmi získava informácie o botnetoch, DDoS, zraniteľnostiach a ďalších typoch hrozieb, ďalej ich testuje a následne vytvára a odosiela bezplatné denné správy o zistených skutočnostiach. Súčasťou správ sú však aj dáta o otvorených portoch a službách, ktoré na nich bežia. [26]

Hrozby z denných správ Shadowserver Foundation pre sieť CESNET sa využívajú v systéme Mentat (opísaný v podsekcii 2.1.4), takže operátori oddelenia CESNET SOC sa im venujú. Metadáta o skenovaných zariadeniach a službách sa vkladajú do systému ADiCT (opísaný v podsekcii 2.1.2). V súčasnosti ide o údaje o HTTP (Hypertext Transfer Protocol), SSL/TLS (Secure Sockets Layer/Transport Layer Security), SSH (Secure Shell Protocol), RDP (Remote Desktop Protocol) a SMTP (Simple Mail Transfer Protocol).

2.1.12 NetBox



Obr. 2.7: **Používateľské rozhranie NetBox.** Prevzaté z [27].

[32] https://www.shadowserver.org/

[33] Sinkholing je (v tomto kontexte) metóda pre presmerovanie komunikácie medzi malware a kontrolným serverom na vlastný server. Takýmto spôsobom je možné monitorovať a čiastočne pochopiť prebiehajúcu komunikáciu a následne bližšie identifikovať typ infekcie.

NetBox[34] je dokumentačný systém pre všetky aspekty organizácie siete. Má byť *source of truth* – definovať zamýšľaný stav. Na jeho základe umožňuje automatizovať procesy, monitorovať zariadenia, detekovať nesúlad. V rámci dátového modelu umožňuje dokumentáciu rozdelenia IP adresných rozsahov, virtuálnych sietí, serverovní, kabeláže, VPN (virtual private network), autonómnych systémov (AS) a ďalšie, ako ilustruje obrázok 2.7. [27]

Implementácia systému NetBox do siete CESNET v čase písania tejto bakalárskej práce prebieha. Lokálna inštancia má byť využitá ako *source of truth* v sieti CESNET a má byť jednotným miestom pre dokumentáciu celej e-infraštruktúry. Okrem iného má obsahovať aj očakávaný stav pre všetky spravované zariadenia v sieti CESNET — teda hlavne servery a časť klientských staníc — vrátane informácií o operačnom systéme, type zariadenia a poskytovaných službách na danom zariadení, ktoré môžu slúžiť na detekciu anomálií. Pôvodne bolo napojenie systému NetBox špecifikované medzi požiadavkami na implementáciu systému Amfora, avšak keďže inštancia NetBox zatiaľ neobsahuje potrebné dáta, je ďalej iba počítané s jej budúcim napojením a táto podkapitola je uvedená pre úplnosť a zdôvodnenie niektorých rozhodnutí.

2.2 Platforma DP^3

DP^3 je generická platforma pre uloženie dynamických dát o entitách. Principiálne ide o databázu umožňujúcu pripájanie vlastných modulov pre príjem, spracovanie a odvodzovanie dát. Platforma DP^3 bola vyvinutá v rámci projektu ADiCT (opísaný v sekcii 2.1.2) pre uloženie dát o sieťových entitách (IP adresách a ďalších), avšak od začiatku bola snaha generalizovať dátový model, preto nie je charakter uložených dát nijako obmedzený. Aktuálne je DP^3 open-source projekt[35], počíta sa s jeho využitím aj v projekte NERD (opísaný v sekcii 2.1.5) a ide taktiež o jednu z požiadaviek aj pre základ systému Amfora.

2.2.1 Dátový model platformy DP^3

Základom dátového modelu sú entity a atribúty. Entita je identifikovateľný objekt. Na identifikáciu sa využíva primárny kľúč, v tomto prípade označený `eid` (entity ID). K entite sa viaže množina atribútov, ktoré majú označenie, typ, hodnotu a ďalšie metadáta. Každý atribút má unikátne označenie a môže mať ľubovoľný dátový typ. Entity a ich atribúty však podliehajú konfigurácii, ktorá definuje dátový model pre danú aplikáciu. Entity je taktiež možné vzájomne prepájať. Konfigurácia je v jazyku YAML[36]. Jej príklad ilustruje výpis 2.1.

```
entity:
  name: Bus
attribs:
  label:
    description: Custom label for the bus.
    type: plain
    data_type: string
  speed:
    description: Speed of the bus in a~particular time. In km/h.
    type: observations
```

[34] https://github.com/netbox-community/netbox
[35] https://github.com/CESNET/dp3
[36] YAML: https://yaml.org/

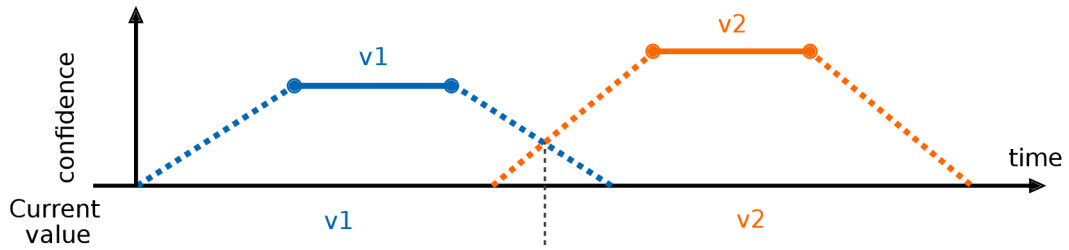
```
    data_type: float
    history_params:
      pre_validity: 1m
      post_validity: 1m
  ...
```

Výpis 2.1: **Príklad konfigurácie entity v platforme DP^3.** Konfigurácia je zámerne skrátená a ilustruje všeobecnosť platformy. Definuje typ entity `Bus`, ktorý má 2 atribúty: prostý atribút štítok (`label`) s dátovým typom reťazec a atribút rýchlosť (`speed`) typu pozorovanie s dátovým typom desatinné číslo.

Platforma DP^3 rozlišuje tri typy atribútov: prosté (plain), pozorovania (observations) a časové rady (timeseries).

Prostý atribút je najjednoduchším typom atribútu. Má priradenú iba aktuálnu hodnotu a čas poslednej zmeny. Neukladá sa história jeho predchádzajúcich hodnôt. Je teda podobný pojmu atribút z objektovo orientovaného programovania či bežných databázových systémov.

Atribút typu **pozorovanie** je najčastejšie využívaným typom atribútu na platforme DP^3. Jeho hodnota je funkciou času, pretože každá hodnota, ktorá mu je priradená, má definovaný začiatok (označený `t1`) a koniec platnosti (označený `t2`). Takto je možné uložiť nie len aktuálnu hodnotu, ale aj kompletnú históriu tých predchádzajúcich. Na rozdiel od prostého atribútu je ďalej možné ukladať pri každej hodnote mieru dôveryhodnosti (confidence) vyjadrenú desatinným číslom v intervale $\langle 0, 1 \rangle$. Takéto rozšírenie umožňuje aplikáciu aj na dáta, ktoré nie sú spoľahlivé. A nakoniec je možné označiť pozorovanie za viachodnotový (multi value) atribút – vyjadruje viacero súbežných aktuálnych hodnôt.



Obr. 2.8: **Hodnota atribútu typu pozorovanie pri prekrývajúcich sa platnostiach dátových bodov v platforme DP^3.** Atribút nie je viachodnotový (multi value). Prevzaté z [25].

Typ atribútu pozorovanie je pomerne komplexným modelovacím nástrojom. Jeho konfigurácia (ukázané v atribúte `speed` na výpise 2.1) však umožňuje definovať aj parametre pre interpoláciu hodnôt (a dôveryhodností) pred časom `t1` a po čase `t2`. Význam ilustruje obrázok 2.8 v komplikovanejšom prípade, kedy dôjde k čiastočnému prekryvu dvoch dátových bodov. Bližší popis vrátane riešenia situácií s viachodnotovými atribútmi či agregácie je možné nájsť v dokumentácii[37].

Posledným typom atribútu je **časová rada**. Ide o najnovšie rozšírenie platformy DP^3. Typ časová rada umožňuje ukladanie priebehu jednej alebo viacerých numerických hodnôt v čase. Rozdiel oproti pozorovaniu s numerickým dátovým typom spočíva v sémantike a spôsobe spracovania. Pri atribútoch s typom pozorovanie sa odvodzovanie aktuálnej

[37] https://cesnet.github.io/dp3/history_management/

hodnoty riadi hodnotami dôveryhodnosti, ako je popísané vyššie. Pri časových radách sa predpokladá, že jednotlivé merania budú mať všeobecne rôzne hodnoty a na ich spracovanie sa využijú agregačné funkcie (minimum, maximum, suma a podobne). Tento typ sa hodí predovšetkým na logovacie účely alebo na účely analýzy historických hodnôt. Časová rada môže byť: pravidelná (regular), nepravidelná (irregular) alebo nepravidelná s intervalmi (irregular intervals). Pravidelná časová rada má definovaný interval príchodu dátových bodov (napríklad 10 minút). Nepravidelné časové rady takýto interval definovaný nemajú a namiesto toho využívajú časové značky (podobne ako typ atribútu pozorovanie). V prípade nepravidelnej časovej rady bez intervalov ide iba o čas `t1` (sémantika je typicky, že v čase `t1` nastala nejaká udalosť), v prípade nepravidelnej časovej rady s intervalmi ide o čas `t1` aj `t2` (sémantika je typicky, že v čase `t1` nastala udalosť, ktorá spustila meranie prebiehajúce do času `t2` a dátový bod obsahuje jeho výsledky).

2.2.2 Architektúra platformy DP^3

Vstup dát do platformy DP^3 prebieha prostredníctvom **dátových bodov** (data points). Dátový bod je jednotka modifikácie stavu – zmena hodnoty prostého atribútu alebo pridanie nového merania v atribúte typu pozorovanie alebo časová rada. Ide o štruktúru obsahujúcu identifikáciu entity, atribútu, hodnotu atribútu a prípadné ďalšie informácie (časové značky `t1` a `t2`, dôveryhodnosť, identifikáciu zdroja). Dátové body vkladajú do platformy primárne moduly alebo externé systémy prostredníctvom **API** rozhrania. V rámci API rozhrania sa z dátových bodov stáva úloha (task), ktorá je vložená do fronty na spracovanie. Príklad dátového bodu je na výpise 2.2. Webové API rozhranie ďalej umožňuje získavanie a mazanie uložených dát a kontrolu behu platformy.

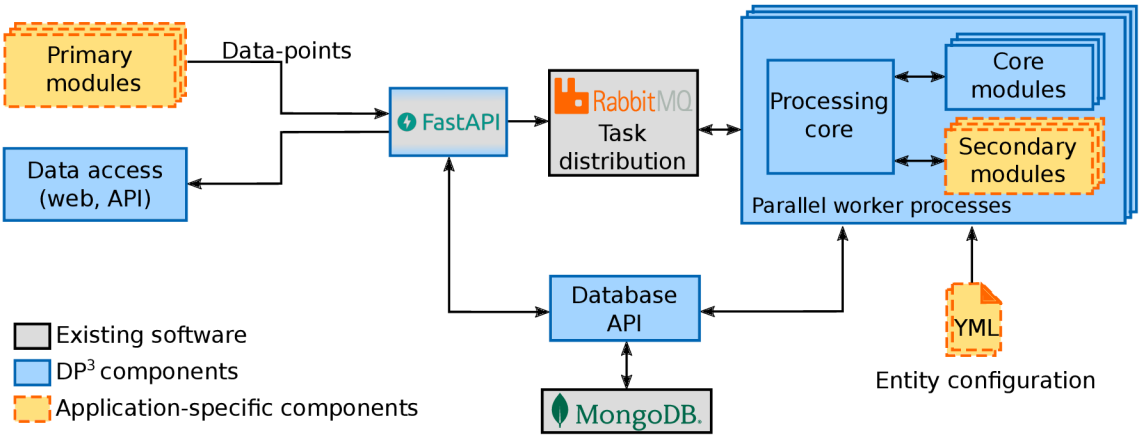
```
{
  "type": "bus",
  "id": "N99",
  "attr": "speed",
  "v": 38.4,
  "t1": "2024-01-01T23:14:00",
  "t2": "2024-01-01T23:15:00",
  "src": "GPS@bus"
}
```

Výpis 2.2: **Príklad dátového bodu v platforme DP^3.** Zodpovedá konfigurácii z výpisu 2.1.

Procesným jadrom platformy sú **worker procesy**. Tie spracúvajú prichádzajúce požiadavky vo forme úloh (tasks) z API. Úlohy môžu byť vytvárané aj interne (pri tvorbe snímok). Starajú sa o aktualizáciu databázy a komunikujú so sekundárnymi modulmi. Ako databáza sa od výraznej refaktorizácie v roku 2023 používa MongoDB[38]. Pre zjednodušenie prístupu k dátam — predovšetkým počas vývoja vlastnej aplikácie — bol v roku 2023 vytvorený taktiež plugin do vizualizačného nástroja Grafana[39]. Aplikačne špecifické komponenty pozostávajú z primárnych a sekundárnych modulov a konfigurácie. **Primárne moduly** slúžia na vkladanie dát do platformy. **Sekundárne moduly** reagujú na udalosti. Typicky ide o zmenu dát v systéme, avšak rovnako je možné zaregistrovať volanie sekundárneho modulu aj pri spracovávaní úlohy, pri vytváraní snímky alebo periodicky. Nákres architektúry sa

[38] MongoDB: https://www.mongodb.com/
[39] Grafana plugin platformy DP^3: https://github.com/CESNET/dp3-grafana/

nachádza na obrázku 2.9. Pre implementáciu je použitý jazyk Python verzie 3 a validitu dát zaisťuje knižnica Pydantic[40].



Obr. 2.9: **Architektúra platformy DP³.** Platforma pozostáva z primárnych modulov, API rozhrania, databázy, front na distribúciu úloh, worker procesov, sekundárnych modulov a konfigurácie. Prevzaté z [25].

2.2.3 Spracovanie dát v platforme DP³

Platforma DP³ využíva pre každý typ entity 3 kolekcie[41]: kolekcia pre surové dáta (raw), master kolekcia, kolekcia snímok (snapshots). **Kolekcia pre surové dáta** je záznamom všetkých prijatých validných dátových bodov. **Master kolekcia** obsahuje profily entít, ktoré sa priebežne aktualizujú – v prípade prostých atribútov obsahuje aktuálnu hodnotu, v prípade atribútov typu pozorovanie a časových rád kompletnú históriu. Tieto dáta slúžia ako podklad pre tvorbu snímok, ktoré sú súčasťou **kolekcie snímok**. Snímka reprezentuje profil entity v určitom časovom okamihu. Obsahuje aktuálnu hodnotu prostých atribútov a pozorovaní. Hodnoty atribútov typu časová rada sa do snímok nekopírujú. Sekundárne moduly môžu rozšíriť snímku o ďalšie odvodené dáta počas dátovej korelácie. Taktiež v tomto procese dochádza ku spájaniu viazaných entít.

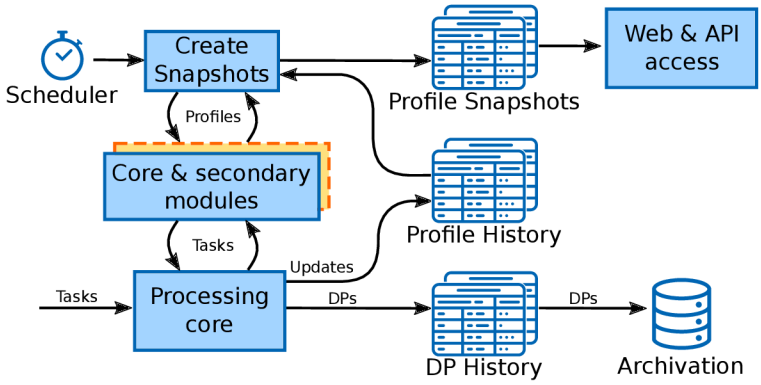
Prijatý dátový bod sa počas spracúvania worker procesom ukladá iba do prvých dvoch spomínaných kolekcií – kolekcie pre surové dáta a master kolekcie. Vytváranie snímok sa spúšťa z dôvodu výpočtovej náročnosti periodicky s periódou nastaviteľnou v konfigurácii. Tento proces ilustruje obrázok 2.10.

2.3 Metódy implementácie moderných webových rozhraní

Rozmach webu bezpochyby zmenil svet. Len málokto si dnes dokáže predstaviť život bez neho. Internetové nakupovanie zažíva nevídaný rozvoj, komunikácia sa presúva z fyzického sveta do prostredia sociálnych sietí a komunikačných aplikácií a informácie prijímame z elektronických denníkov, nie tlačených novín. Web prináša čoraz viac možností a tento trend bude iba pokračovať.

[40] Pydantic: https://docs.pydantic.dev/2.7/

[41] Kolekcia je označenie súboru dokumentov v MongoDB. Značne zjednodušene je kolekcia analogická tabuľke v relačnej databáze a dokument je analogický riadku tabuľky. MongoDB však nemá definíciu schémy.



Obr. 2.10: **Spracovanie prijatých dát vo worker procese platformy DP^3.** *DP History* je kolekcia pre surové dáta, *Profile History* master kolekcia a *Profile Snapshots* kolekcia snímok. Tvorba snímok sa spúšťa periodicky z plánovača. Prevzaté z [25].

S rozvojom webu však prichádzajú zvyšujúce sa nároky na samotné webové aplikácie. Súčasný web už nie je statický, ale dynamicky sa prispôsobuje všetkým zmenám informácií. A na to reagujú aj technológie, ktoré sa na implementáciu webových aplikácií a používateľských rozhraní používajú.

Implementácia webových aplikácií vychádza prevažne z dvoch prístupov (prípadne ich kombinácie): vykresľovanie na strane serveru (server-side rendering, SSR) alebo vykresľovanie na klientskej strane (client-side rendering, CSR). Kým v minulosti výrazne prevažoval prvý prístup, v súčasnosti je hlavný druhý prístup. Ide o dôsledok zvyšujúcich sa požiadaviek na dynamickosť webových portálov, používateľskú skúsenosť (UX), zvyšujúce sa množstvo dát a požiadaviek na modularitu. [15]

Práve z týchto dôvodov moderné webové aplikácie typicky využívajú trojvrstvovú architektúru. Ide rozdelenie software do troch úrovní (tiers): prezentačnej, aplikačnej a dátovej. Prezentačnú úroveň tvorí používateľské rozhranie. Ide o klientskú časť aplikácie označovanú tiež ako **frontend**. Aplikačnú úroveň tvorí aplikačná logika. Táto úroveň prijíma a vykonáva požiadavky z prezentačnej úrovne a poskytuje jej dáta z dátovej úrovne. Označuje sa ako **backend**. Dátová úroveň ukladá a spravuje stav aplikácie – teda všetky jej údaje. Túto úroveň implementuje databáza alebo obdobný systém. [7]

V projekte Amfora je požiadavkami špecifikované použitie platformy DP^3. Tá predstavuje dátovú úroveň aplikácie. Nasledujúce podsekcie preto obsahujú popis a porovnanie implementačných technológií na zvyšných dvoch úrovniach.

2.3.1 Frontend moderných webových rozhraní

Na implementáciu frontendovej časti webovej aplikácie sa využíva značkovací jazyk HTML (HyperText Markup Language), kaskádové štýly CSS (Cascading Style Sheets) a dynamické správanie určuje kód v jazyku JavaScript (prípadne TypeScript preložený do jazyku JavaScript). Keďže hlavnou úlohou používateľského rozhrania je interakcia s používateľom (človekom), dôraz sa kladie na používateľskú skúsenosť (UX).

Frontendová časť webovej aplikácie je najčastejšie založená na frameworku. JS/TS frameworky umožňujú delenie webu na komponenty a zjednodušujú implementáciu interak-

tívnych častí. Podľa prieskumu *State of JavaScript* [13] z roku 2022[42] majú dominantné postavenie 3 frameworky: Angular, React a Vue.

Angular

Vznik frameworku Angular[43] sa datuje do roku 2009 a spája sa hlavne s menom Miško Hevery pracujúcim v spoločnosti Google. Angular pôvodne vznikol ako bočný projekt, ktorý mal zjednodušovať pridávanie interaktívnych prvkov do statických webových stránok. V spoločnosti Google sa Angular využil na viacerých interných projektoch a jeho popularita postupne rástla. Neskôr došlo k premenovaniu Angular verzie 1 na AngularJS a kompletnej refaktorizácii a vytvoreniu Angular verzie 2, ktorý sa označil ako Angular a implementačne prechádzal na jazyk TypeScript. Angular verzie 1 v tom čas výrazne naberal na popularite a používal sa aj na projektoch, pre ktoré pôvodne vôbec nebol určený. Kompletný prepis kódu však zaznamenal negatívnu odozvu od používateľov a často dochádzalo prechodom na framework React. [12]

V súčasnosti framework Angular umožňuje dosiahnuť vysokú mieru škálovateľnosti aj pri veľkých objemoch dát. Aj naďalej ide o projekt aktívne vyvíjaný a spravovaný spoločnosťou Google. [23]

React

React[44] vytvoril v roku 2011 Jordan Walke zo spoločnosti Facebook (dnes Meta). O dva roky neskôr sa z neho stal open-source projekt. Súčasný vývoj Reactu naďalej zabezpečuje spoločnosť Meta a využíva ho vo svojich službách Facebook a Instagram. [23]

Základom frameworku React je rozdelenie používateľského rozhrania do menších reaktívnych komponentov. Na implementáciu komponentov sa využíva jazyk JavaScript/TypeScript, avšak HTML kód je možné zapisovať s využitím JSX[45].

Vue

Vue[46] je projekt Evana You. You počas práce v spoločnosti Google používal framework Angular, ktorý sa mu však zdal na jednoduché aplikácie príliš ťažkopádny. Vue vytvoril ako výrazne jednoduchší a menší framework vychádzajúci z aspektov Angularu, ktoré sa mu páčili. Základ Vue aktuálne tvorí koncept komponentov a data binding, ktorý je viac podobný frameworku React než Angular. O open-source projekt ide od roku 2014 – Vue je najmladším zo spomínaných frameworkov. [23]

Snaha Evana You spočívala vo využití konceptov značkovacieho jazyka HTML a kaskádových štýlov CSS a ich rozšírení. Takto bolo možné dosiahnuť intuitívnosť rozhrania frameworku. Vue využíva koncept jednosúborových komponentov. Takýto komponent je syntakticky HTML súbor, avšak sémanticky ide o spojenie 3 častí (HTML, CSS a JavaScript) do jedného súboru, ktoré kompletne definujú štruktúru, vzhľad a správanie jedného komponentu.

[42] Prieskum *State of JavaScript* z roku 2023 v čase písania tejto bakalárskej práce nebol zverejnený.
[43] Angular: `https://angular.io/`
[44] React: `https://react.dev/`
[45] JSX vo frameworku React: `https://react.dev/learn/writing-markup-with-jsx`
[46] Vue: `https://vuejs.org/`

Porovnanie

Menované frameworky porovnáva vo svojej bakalárskej práci Elar Saks [23]. Podľa jeho zistení je Vue najvýkonnejším frameworkom (dvojnásobne oproti React a päťnásobne oproti Angular). Veľkosť produkčnej zostavy frameworku má najmenšiu React (587 kB), tesne nasledovaný Vue (624 kB) a zďaleka najväčšiu zostavu mal framework Angular (vyše 15 MB). Najjednoduchším frameworkom na naučenie je podľa Saksa Vue kvôli využitiu syntaxe čistého HTML a CSS. Nasleduje React s mierne komplikovanejšou syntaxou, avšak podobnými konceptmi a najzložitejší je framework Angular kvôli jeho obsiahlosti. Ako však autor spomína, ide o jeho subjektívny názor, keďže učiacu krivku nie je možné kvantifikovať. Z hľadiska popularity jednoznačne vyhráva framework React nasledovaný Vue a Angular a tento trend je možné pozorovať aj v čase písania tejto bakalárskej práce[47].

Okrem spomínaných frameworkov samozrejme existuje a neustále vzniká množstvo ďalších. Výhodou najpoužívanejších je však okrem reputácie aj dobrá podpora, rozšíriteľnosť, spoľahlivosť a existencia množstva dokumentácie a návodov. Tieto aspekty sú dôležité aj pre túto bakalársku prácu, pretože systém Amfora má predpoklady na ďalšie budúce rozširovanie. Z tohto dôvodu sa táto podsekcia venovala iba trom najvýznamnejším.

2.3.2 Backend moderných webových rozhraní

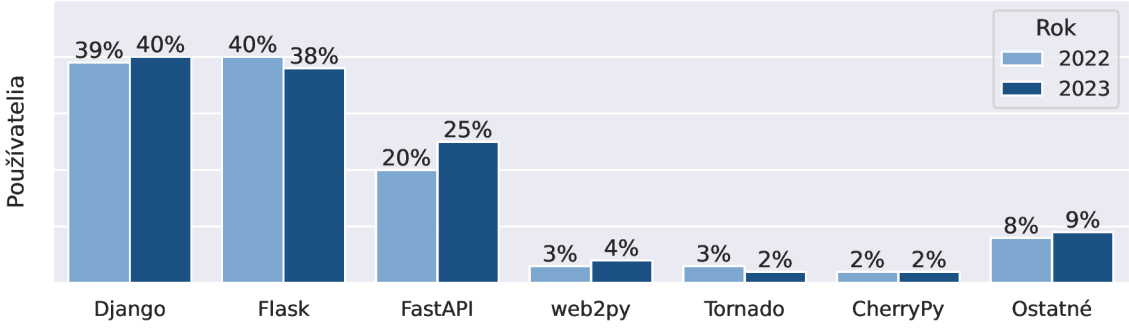
Backendová časť je oproti frontendovej výrazne flexibilnejšia v možnostiach implementácie. Požiadavkou je iba vytvorenie rozhrania pre serializovanú komunikáciu s frontendom, ktorá bude efektívne reprezentovať dáta uložené na dátovej úrovni. K dispozícii je viacero prístupov ku komunikácii (REST[48] API založené na JSON[49], XML[50], GraphQL[51], SOAP[52] a ďalšie) a na výbere programovacieho jazyka či frameworku z hľadiska systémového návrhu prakticky nezáleží. Napriek tomu aj na aplikačnej úrovni existujú viac a menej typické spôsoby implementácie. Pre systém Amfora som zvolil implementačný jazyk Python, pretože takmer všetky systémy vyvíjané v združení CESNET sú implementované práve s jeho použitím, mám s ním skúsenosti a ide o jeden z najpopulárnejších jazykov. Táto podsekcia je preto prehľadom Python frameworkov.

Z hľadiska komunikácie budem uvažovať iba o REST API založenom na formáte JSON. REST je štýl rozhrania využívajúci metódy protokolu HTTP na zakódovanie prístupu (`GET`), vytvárania (`POST`), modifikácie (`PUT`) alebo mazania (`DELETE`) objektu. Využitie GraphQL nemá z pohľadu systému Amfora zmysel, pretože dátovú vrstvu pre backend budú tvoriť API rozhrania ostatných systémov spomínaných v tejto kapitole, predovšetkým DP^3, ktoré GraphQL nepodporujú. Rozhranie založené na formáte XML (vrátane SOAP) by bolo možné, avšak zvýšila by sa réžia komunikácie a znížila podpora bežných frameworkov. REST API s využitím formátu JSON je najbežnejšie a univerzálne používané riešenie pre dátovú komunikáciu.

Pri výbere frameworku je dôležitá aj jeho popularita. Najpopulárnejšie frameworky majú vďaka výrazne širšej používateľskej základni lepšiu podporu, dokumentáciu a rozšíriteľnosť. Nedostatočne populárne frameworky, naopak, zvyšujú náročnosť riešenia problémov, ktoré pri ich využití môžu vznikať. Na základe správy *The State of Developer Ecosystem 2023*

[47] https://npmtrends.com/angular-vs-react-vs-vue
[48] REST: Representational State Transfer
[49] JSON: JavaScript Object Notation
[50] XML: Extensible Markup Language
[51] GraphQL: https://graphql.org/
[52] SOAP: Simple Object Access Protocol

od spoločnosti JetBrains [14] boli v roku 2023 v jazyku Python najpopulárnejšie webové frameworky Django, Flask a FastAPI. Výsledky zobrazuje obrázok 2.11.



Obr. 2.11: **Najpopulárnejšie webové frameworky jazyka Python v roku 2023.** Vlastná tvorba, dáta prevzaté z [14].

Django

Framework Django[53] je najznámejší webový framework jazyka Python, ktorého cieľom je uľahčiť vývoj dynamických webových aplikácií. Umožňuje kompletnú správu databázy v backende vďaka zabudovanému systém pre objektovo-relačné mapovanie (ORM). Navyše automaticky vytvára používateľské rozhranie pre administráciu databázy a v základnom stave podporuje aj autentifikačný systém podporujúci používateľské účty, oprávnenia a sedenia (sessions). Keďže primárnym zameraním frameworku Django sú webstránky s vykresľovaním na strane servera, odpovede na HTTP požiadavky sa očakávajú vo forme HTML dokumentov. Pre zjednodušenie práce Django poskytuje nástroje pre tvorbu a spracovanie HTML šablón s vykresľovaním výsledných dokumentov na základe kontextu a podporu pre internacionalizáciu webovej aplikácie. Framework Django je založený na vzore MTV (model – template – view), ktorý je podobný návrhovému vzoru MVC (model – view – controller). Model je kompletný popis štruktúry dát entity. Jeden model sa mapuje na jednu tabuľku relačnej databázy. Šablóny (templates) umožňujú dynamicky vytvárať HTML kód, ktorý sa odosiela pri prístupe na stránku používateľovi, teda vytvárať personalizované odpovede. Pohľad (view) definuje, aké dáta budú obsahom odpovede. Implementačne ide o funkciu vykonávajúcu spracovanie HTTP požiadavky. [1, 29]

Flask

Flask[54] je považovaný za microframework. V základe obsahuje podporu iba pre smerovanie požiadaviek, vykresľovanie HTML dokumentov na základe šablón a poskytovanie statických súborov. V prípade potreby je možné ďalšie funkcie pridať prostredníctvom rozšírení. Flask nevyžaduje ani dodržanie žiadneho konkrétneho návrhového vzoru a prenecháva toto rozhodnutie na používateľa. Z týchto dôvodov ide o populárnu možnosť pre začiatočníkov alebo veľmi jednoduché aplikácie s vykresľovaním na strane serveru. [3]

[53] Django: https://www.djangoproject.com/
[54] Flask: https://flask.palletsprojects.com/

FastAPI

FastAPI[55] je najnovším z tejto trojice frameworkov. Ide o microframework určený pre tvorbu API rozhraní s formátom JSON. Vďaka využitiu asynchrónnych volaní umožňuje dosahovať vysoký výkon, ktorý je porovnateľný aj s kompilovanými jazykmi. Na základe prieskumu [14] jeho popularita neustále rastie a od roku 2021 sa zvýšila takmer dvojnásobne. [2]

Hlavnou výhodou frameworku FastAPI je využitie type hints[56] jazyka Python na validáciu parametrov HTTP požiadavky a odpovede na ňu. V prípade nevalidnej požiadavky sa automaticky vytvorí podrobné hlásenie o chybe, ktoré sa vráti odosielateľovi. Framework taktiež automaticky generuje dokumentáciu k API rozhraniu na základe štandardu OpenAPI[57] a JSON Schema[58].

Validáciu zabezpečuje knižnica Pydantic[59] – najpoužívanejšia knižnica pre validáciu dát v jazyku Python. Podobne ako FastAPI, aj Pydantic využíva type hints jazyka Python na získanie typovej informácie. Jej základ je napísaný v kompilovanom jazyku Rust, preto dosahuje vysokú výkonnosť aj v prípade validácie komplikovaných štruktúr. Ide o overenú knižnicu s miliónmi stiahnutí denne. [4]

Príklad validačného modelu s využitím knižnice Pydantic je uvedený na výpise 2.3.

```
from datetime import datetime
from pydantic import BaseModel, NonNegativeFloat

class Bus(BaseModel):
  label: str
  speed: NonNegativeFloat = 0.0
  serviced: datetime
```

Výpis 2.3: **Príklad definície triedy využívajúcej knižnicu Pydantic.** Definuje triedu reprezentujúcu autobus, ktorý má označenie (`label`) typu reťazec, rýchlosť (`speed`), ktorý musí byť nezáporným desatinným číslom a dátum poslednej doby servisu (`serviced`).

Porovnanie

Napriek podobnosti základných princípov webových frameworkov jazyka Python sú medzi tromi opisovanými možnosťami značné rozdiely. Framework Django je pomerne obsiahly a snaží sa riešiť množstvo problémov navyše. Spoločne s frameworkom Flask však ide o riešenia skôr určené pre webové aplikácie s vykresľovaním na strane servera, kým framework FastAPI je určený pre tvorbu API rozhraní. Flask je rozhodne možné prispôsobiť aj pre API server, avšak jednoduchšie je prosté využitie frameworku FastAPI, čo zároveň poskytuje aj typovú bezpečnosť a vygenerovanú dokumentáciu. FastAPI na rozdiel od zvyšných možností využíva asynchrónne volania, čím je výrazne výkonnejší.

[55] FastAPI: https://fastapi.tiangolo.com/
[56] Type hint: Voliteľné pridanie typovej informácie k premennej, argumentu funkcie alebo návratovej hodnote v jazyku Python.
[57] OpenAPI: https://github.com/OAI/OpenAPI-Specification
[58] JSON Schema: https://json-schema.org/
[59] Pydantic: https://docs.pydantic.dev/2.7/

Kapitola 3

Návrh systému Amfora

Podstata tejto bakalárskej práce spočíva vo vývoji používateľského rozhrania pre systém pre správu zariadení v sieti nazývaný Amfora. Amfora agreguje dáta o IP adresách, službách a zraniteľnostiach a slúži predovšetkým potrebám oddelenia SOC v združení CESNET. Nasledujúca kapitola je zhodnotením existujúceho stavu a návrhom ďalšieho vývoja. Obsahuje však návrh systému ako celku, nie iba používateľského rozhrania. Tieto časti nie je možné oddeliť, keďže sú úzko prepojené, ide o nový systém a ich vývoj prebieha súbežne.

Sekcia 3.1 opisuje prácu, ktorá bola vykonaná pred touto bakalárskou prácou v rámci tvorby základu celého systému založeného na platforme DP^3. Následne je adresovaná tvorba demo verzie používateľského rozhrania s využitím existujúceho základu systému (sekcia 3.2). Zhromažďovanie podkladov pokračuje realizáciou používateľského prieskumu v sekcii 3.3. Všetky zistené požiadavky pre prehľadnosť zhŕňa sekcia 3.4. Nasleduje vytvorenie mockupu časti používateľského rozhrania a jeho zhodnotenie budúcimi používateľmi v sekcii 3.5. Záver kapitoly je venovaný samotnému návrhu architektúry celého systému z implementačného hľadiska a výber použitých technológií (sekcia 3.6).

3.1 Základ systému Amfora

Úvahy o potrebe nového systému pre správu zariadení v sieti sa začali v oddelení CESNET SOC objavovať už dlho pred vypísaním zadania tejto bakalárskej práce. Primárnym problémom, ktorý pozorovali, bolo veľké množstvo oddelených systémov využívaných pri riešení incidentov a s tým súvisiaca zložitosť vyhľadávania a spracovania informácií. To časom vyústilo do diskusií s oddelením *Nástrojov pre administráciu a bezpečnosť*, ktorého súčasťou je vedúci aj konzultant tejto práce, a ktoré ponúklo pomoc pri vytvorení tohto systému.

Samotný vývoj systému Amfora začal v júli 2023 a od začiatku išlo o projekt založený na platforme DP^3, ktorá je opísaná v sekcii 2.2. Vývoj zastrešovali Václav Bartoš a Ondřej Sedláček zo združenia CESNET. Platforma DP^3 bola zvolená ako vhodná pre tento typ systému. Dáta spočiatku pochádzali zo systému SNER (podsekcia 2.1.1), ADiCT (podsekcia 2.1.2), Auror (podsekcia 2.1.3), služby Shodan (podsekcia 2.1.10) a z projektu ShadowServer (podsekcia 2.1.11), a všetky sa ukladali do platformy DP^3. O mesiac neskôr som začal pracovať na tvorbe demo verzie používateľského rozhrania, ktorého bližší popis nasleduje v sekcii 3.2. To už je súčasťou bakalárskej práce.

Spočiatku obsahovala platforma DP^3 systému Amfora dva typy entít – IP adresa (*ip*) a služba (*service*). V rámci typu entity IP adresa sa zbierali nasledujúce informácie:

- **SNER**: hostname, komentár, štítky, operačný systém

- **ADiCT**: operačný systém, typ zariadenia
- **Shodan**: operačný systém
- **ShadowServer**: identifikátor zariadenia

V rámci typu entity služba sa zbierali nasledujúce informácie:

- **SNER**: použitý protokol transportnej vrstvy, stav protokolu TCP[1], softwarový produkt a jeho verzia, typ zariadenia, hostname, operačný systém, JARM[2] odtlačok, CPE[3], titulok HTTP odpovede, reverzný DNS záznam, zraniteľnosti nájdené pomocou skriptu TestSSL
- **Auror**: otvorené porty, baner HTTP servera, známka konfigurácie protokolu TLS
- **Shodan**: otvorené porty, operačný systém, software na jednotlivých portoch
- **ShadowServer**: hlavičky odpovedí HTTP[4] serverov, hlavičky protokolu TLS[5], identifikátor SSH serverov, amplifikácia odpovede oproti požiadavke na rôznych protokolových vrstvách a hlavičky protokolov RDP[6] a SMTP[7]

V tomto čase ešte neboli jasné požiadavky oddelenia CESNET SOC, preto neboli definované jednoznačné ciele systému a zbierané údaje boli iba odhadom, čo pravdepodobne bude potrebné. Zároveň išlo o experimentálnu fázu, v ktorej prebiehalo overovanie rozsahu informácií, ktoré je možné získať alebo odvodiť zo spomínaných zdrojov.

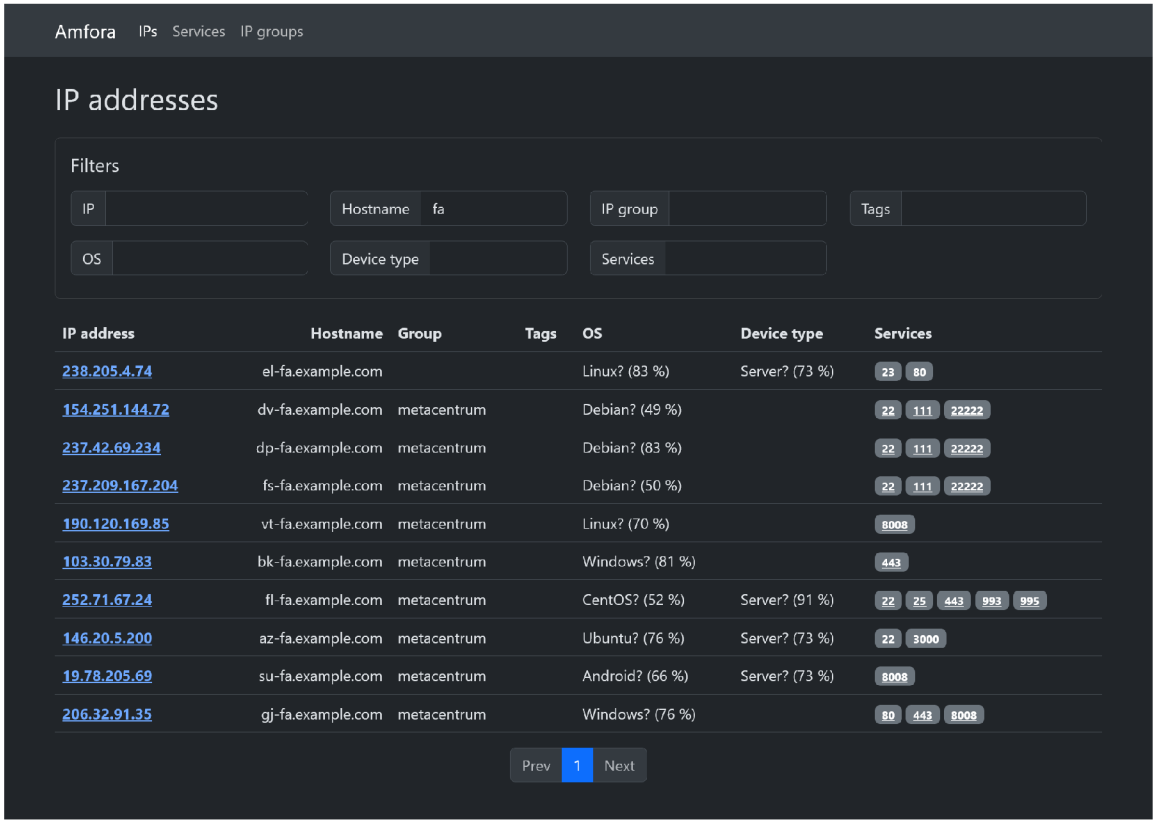
3.2 Demo verzia používateľského rozhrania

Demo verzia používateľského rozhrania bola vyvíjaná ako prvé rozšírenie základu systému Amfora v rámci tejto bakalárskej práce. Išlo o webové rozhranie pre zobrazenie údajov uložených v platforme DP^3. Cieľom bola interaktívna demonštrácia základu systému pre oddelenie CESNET SOC a uľahčenie práce pri ďalšom vývoji.

Keďže išlo o jednoduché webové rozhranie bez autentifikácie, ktoré iba zobrazovalo obsah platformy DP^3, z hľadiska architektúry bol zvolený prístup tvorby frontendovej časti s využitím API rozhrania DP^3 bez tvorby ďalšej vrstvy spájajúcej tieto dva celky. Na implementáciu bol zvolený framework Vue vo verzii 3, ktorého bližší popis obsahuje podsekcia 2.3.1.

Používateľské rozhranie podporovalo zobrazenie prehľadu IP adries a služieb v systéme Amfora. Prehľad obsahoval filtrovanie podľa viacerých atribútov, tabuľku s výsledkami a stránkovanie. Zadané filtrovacie reťazce sa ukladali do URL adresy prehliadača kvôli možnosti odkazovania a po kliknutí na identifikátor entity bol používateľ presmerovaný na detail danej entity. Detail entity zobrazoval zoznam atribútov a ich hodnôt. Atribúty boli rozdelené do viacerých sekcií podľa systému, z ktorého pochádzajú. Ich označenie a popis vychádzal z konfigurácie platformy DP^3. V detaile IP adresy sa navyše nachádzali odkazy

[1] TCP: Transmission Control Protocol
[2] JARM: https://github.com/salesforce/jarm
[3] CPE: Common Platform Enumeration
[4] HTTP: Hypertext Transfer Protocol
[5] TLS: Transport Layer Security
[6] RDP: Remote Desktop Protocol
[7] SMTP: Simple Mail Transfer Protocol



Obr. 3.1: **Prehľad IP adries v demo verzii používateľského rozhrania systému Amfora.** Prehľad obsahuje filtrovanie podľa IP adresy, hostname, skupiny IP adries, štítkov zo systému SNER, operačného systému, typu zariadenia a služby. Filter podľa hostname obsahujúce *fa* je aktívny. Tabuľka obsahuje výsledky (jednotlivé IP adresy – entity). Operačný systém a typ zariadenia vychádzajú z odvodzovacích pravidiel systému ADiCT, ktoré zároveň priradzujú hodnotám mieru dôveryhodnosti. Služby sú vyjadrené číslom portu. Anonymizované náhodnými IP adresami a hodnotami hostname.

Amfora IPs Services IP groups

Service: 97.61.26.125:443 IP

Auror

Auror open port	
Auror TLS grade	
HTTP Server banner	

Shodan

Shodan open port	true
Shodan - server software	[{ "port": 443, "sw": "Apache httpd" }]

SNER

Service tcp state	open:syn-ack
Service tcp proto	tcp
Service info	{ "product": "Apache httpd", "version": "2.4.37", "extrainfo": "(rocky) OpenSSL/1.1.1k", "devicetype": null, "ostype": null, "hostname": null, "workgroup": null, "time": null }
Service info text	product: Apache httpd version: 2.4.37 extrainfo: (rocky) OpenSSL/1.1.1k
JARM Fingerprint	5f47eb4d016eb234c70c672cbbc61e1159095e357786e6b0125a0f5ac3d046
Hostnames	
CPE	cpe:/a:apache:http_server:2.4.37
nmap_http_title	{ "title": "Roundcube Webmail :: Welcome to Roundcube Webmail", "redirect_url": null }
SNER additional Nmap data	["nmap.http-headers", "nmap.tls-alpn", "nmap.ssl-cert", "nmap.ssl-date", "nmap.ssl-enum-ciphers"]
TestSSL rDNS	
TestSSL service	
TestSSL finding	

ShadowServer

Shadowserver - HTTP	
Shadowserver - SSL	
Shadowserver - SSH	
Shadowserver - RDP	
Shadowserver - SMTP	
Shadowserver - Portmapper	

Banner OS

Banner OS	rocky

Obr. 3.2: **Detail služby v demo verzii používateľského rozhrania systému Amfora.** Ide o webový server Apache verzie 2.4.37 komunikujúci protokolom HTTPS na porte 443. Detail obsahuje sekcie pre dáta zo systému Auror, Shodan, SNER a ShadowServer.

na služby, ktoré bežia na danej IP adrese a v detaile služby, naopak, odkaz na príslušnú IP adresu.

Neskôr bol základ systému Amfora rozšírený o typ entity skupina IP adries (`ip_group`). Toto rozšírenie bolo zapracované aj do demo verzie používateľského rozhrania.

Demo verzia webového rozhrania však zobrazovala iba aktuálne dáta a nepodporovala výber a zobrazenie histórie ktoréhokoľvek atribútu či entity, hoci sa v platforme DP^3 tieto údaje nachádzali. Keďže cieľom demo verzie bola iba ukážka možností a uľahčenie vývoja, a nemalo byť nasadené do prevádzky, nešlo o výrazné obmedzenie.

Obrázky 3.1 a 3.2 obsahujú snímky obrazovky vytvoreného webového rozhrania.

3.3 Analýza potrieb bezpečnostného operačného strediska CESNET

V rámci získavania požiadaviek na výsledný systém som sa zúčastnil niekoľkých diskusií so zástupcami oddelenia CESNET SOC a nakoniec som vykonal pozorovanie práce operátorov SOC metódou *shadowing* na ich pracovisku. Celkovo som pozoroval prácu piatich operátorov. Vďaka týmto stretnutiam som si ujasnil požiadavky, ciele a získal inšpiráciu pre ich plnenie.

Oddelenie CESNET SOC využíva pri práci viacero interných a externých systémov, ktorých krátky prehľad sa nachádza na začiatku sekcie 2.1 a bližší popis v nasledujúcich podsekciách. Ide o systémy SNER (podsekcia 2.1.1), ADiCT (podsekcia 2.1.2), Auror (podsekcia 2.1.3), Mentat (podsekcia 2.1.4), NERD (podsekcia 2.1.5), PassiveDNS (podsekcia 2.1.6), ExaFS (podsekcia 2.1.7), OTRS (podsekcia 2.1.8), FTAS (podsekcia 2.1.9), Shodan (podsekcia 2.1.10), ShadowServer (podsekcia 2.1.11) a plánovaný NetBox (podsekcia 2.1.12).

Keďže hlavným riešeným problémom bolo od začiatku komplikované vyhľadávanie a spracovanie informácií uložených vo veľkom množstve oddelených systémov, podstata systému Amfora mala spočívať v **agregácii dát** z väčšiny spomínaných systémov. V používateľskom rozhraní sa mal nachádzať dostatok informácií, aby si operátor mohol utvoriť prehľad o aktuálnom a historickom správaní IP adresy či charaktere služby. K bližším informáciám v príslušných systémov mal existovať odkaz do používateľského rozhrania daného systému.

Ďalšou dôležitou požiadavkou bolo automatické porovnávanie skutočného stavu voči staticky definovanému, teda **detekcia anomálií**. Statický stav mal byť definovaný v systéme NetBox, avšak ten ani v čase písania tejto bakalárskej práce neobsahuje potrebné údaje, preto sa od jeho použitia počas implementácie upustilo.

Systém Amfora mal podporovať **zobrazenie histórie** všetkých atribútov, pri ktorých je história hodnôt k dispozícii. Používateľské rozhranie preto malo obsahovať výber dátumu a času určujúceho rozsah zobrazených časových hodnôt.

Požiadavkou bolo aj zobrazenie údajov k **externým IP adresám**. Väčšina incidentov, ktoré riešia operátori SOC, totiž pochádza z externých sietí (mimo sieť CESNET). Aj k takýmto adresám je potrebné zistiť maximum informácií a v ideálnom prípade ich aj ukladať pre možné budúce referencovanie. Od tejto požiadavky sa neskôr ustúpilo z dôvodu nedostatočného množstva dát, ktoré je možné aktuálne získať z vymenovaných systémov.

Rovnako dôležité je pre operátorov SOC aj **vyhľadávanie** entít (IP adries a služieb) na základe ich identifikátora alebo niektorého z významných atribútov. Ako dôležitý atribút bol označený názov softwarového produktu služby či jeho verzia, pretože tieto údaje sú potrebné pre vyhľadávanie entít ovplyvnených zraniteľnosťou.

3.4 Zhrnutie požiadaviek

Celkovo boli identifikované tieto požiadavky na systém Amfora a jeho používateľské rozhranie:

- **Agregácia údajov** z väčšiny používaných systémov
- **Detekcia anomálií** s využitím systému NetBox (neskôr zmenené na predprípravu pre túto funkciu)
- **Detekcia zraniteľností** s využitím údajov zo systémov SNER, Shodan a Mentat
- **Zobrazenie histórie** entít
- **Vyhľadávanie** entít s možnosťou vyhľadávania podľa identifikátora aj podľa významných atribútov (IP adresa, číslo portu, hostname, názov sotwarového produktu, verzia softwarového produktu)
- **Jediná úroveň prístupových práv** (používateľ má prístup ku všetkým dátam)
- Zobrazenie dát k **externým IP adresám** (neskôr zrušená požiadavka)

Požiadavky na rozsah zobrazovaných údajov zahŕňajú:

- reputačné skóre IP adresy zo systému NERD
- zoznam domén prislúchajúcich k IP adrese zo systému PassiveDNS (vrátane histórie)
- zoznam pravidiel na modifikáciu smerovania dátovej prevádzky zo systému ExaFS (vrátane histórie)
- základný prehľad incidentov zo systému Mentat (vrátane histórie)

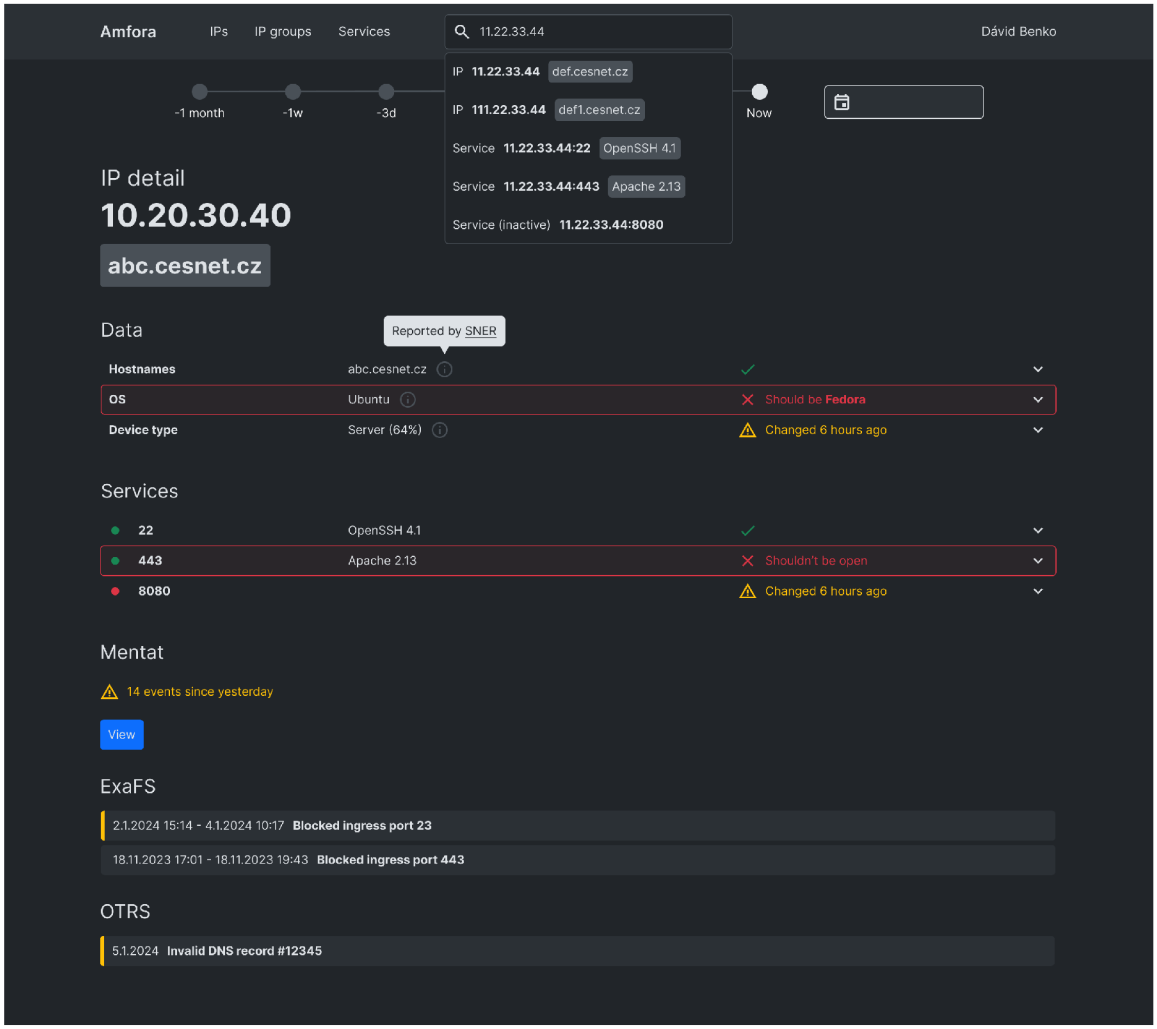
Zo systémov SNER, ADiCT, Auror, Shodan a ShadowServer je postačujúce množstvo údajov, ktoré sa zbiera v základe systému Amfora (sekcia 3.1). K systémom OTRS a FTAS je požadovaný odkaz do ich používateľského rozhrania s predvyplneným vyhľadávaním na údaje relevantné k danej entite.

Všetky požiadavky sa týkajú systému Amfora ako celku, nie len jeho používateľského rozhrania. Keďže súčasťou tejto bakalárskej práce je aj vývoj aplikačnej úrovne, požiadavky nerozdeľujem na jednotlivé úrovne.

3.5 Mockup používateľského rozhrania

Ďalším krokom v návrhu funkcií systému a rozloženia používateľského rozhrania bolo vytvorenie mockupu používateľského rozhrania – modelu jeho budúcej podoby. Tento mockup mal priblížiť a zjednotiť predstavy o výslednom webovom rozhraní medzi mnou a oddelením CESNET SOC, teda primárnym používateľom systému. Išlo o grafický návrh vytvorený v programe Figma[8]. Mockup vychádzal z požiadaviek predstavených v sekcii 3.4, ktoré mi boli dovtedy známe. Graficky boli využité prvky CSS[9] frameworku Bootstrap verzie 5.3[10],

[8] Figma: https://www.figma.com/
[9] CSS: Cascading Style Sheets
[10] Bootstrap: https://getbootstrap.com/docs/5.3/



Obr. 3.3: **Mockup používateľského rozhrania systému Amfora.** Zobrazuje detail IP adresy `10.20.30.40` s hostname `abc.cesnet.cz`, ktorého hodnota pochádza zo systému SNER a je rovnaká ako staticky definovaná. Operačný systém vykazuje anomáliu oproti staticky definovanému stavu – reálny operačný systém je Ubuntu, hoci má byť Fedora. Typ zariadenia sa nedávno zmenil. K IP adrese sa viažu tri služby – na portoch 22, 443 a 8080. Prvé dve služby sú aktívne, tretia neaktívna. Systém Mentat poukázal na 14 bezpečnostných udalostí k tejto IP adrese, v systéme ExaFS sa nachádzajú blokovania a v systéme OTRS je aktívny štítok týkajúci sa nevalidného DNS záznamu. V hornej časti detailu sa nachádza pole na výber dátumu a času, navbar a vyhľadávacie pole s niekoľkými výsledkami.

pretože ide o jeden z najbežnejších frameworkov, mal som s ním skúsenosti a využíva sa vo väčšine projektov v združení CESNET (je teda intuitívny a familiárny aj pre používateľa).

Keďže grafický dizajn vytváraného používateľského rozhrania nie je v tejto bakalárskej práci dôležitý, vytvoril som mockup iba najdôležitejšej časti s najväčším množstvom prvkov – detailu IP adresy (obrázok 3.3).

Základom detailu IP adresy je trojica nadpisov identifikujúcich stránku, dvojica tabuliek s informáciami a niekoľko dodatočných informačných prvkov s inými formátmi. Identifikácia stránky je pomocou názvu stránky, IP adresy a hostname. Nasleduje tabuľka s údajmi zo systému DP^3 – každý riadok obsahuje názov atribútu, hodnotu alebo hodnoty s označením zdroja, overenie danej hodnoty voči staticky definovanému stavu a možnosť otvorenia grafu s históriou hodnôt daného atribútu. Riadky s nevalidnou hodnotou sú zvýraznené červeným pozadím. Druhá tabuľka je podobná prvej, avšak obsahuje zoznam služieb s označením aktivity (aktívny/neaktívny) a názvom softwarového produktu namiesto hodnoty atribútu. Ďalšie tri sekcie tvoria dáta zo systémov Mentat (počet bezpečnostných udalostí), ExaFS (zoznam a popis smerovacích pravidiel) a OTRS (zoznam štítkov k príslušnej IP adrese). V hornej časti detailu sa nachádza časová os pre výber relatívneho času a pole pre zadanie absolútneho času. Navbar tvorí názov systému *Amfora*, odkazy na stránky s prehľadom IP adries, skupín IP adries a služieb, vyhľadávacie pole a meno používateľa. Vyhľadávanie prebieha medzi IP adresami a službami – pri IP adresách sa zobrazuje aj hostname a pri službách názov softwarového produktu, ak je k dispozícii.

Takéto delenie informácií do sekcií som zvolil kvôli charakteru dát v rôznych systémoch. Dátový model platformy DP^3 je založený na formáte atribút – hodnota, hodí sa teda na zobrazenie v tabuľke. V prípade ostatných systémov som v čase tvorby mockupu nemal dostatok informácií o formáte a množstve dát, preto som ich spôsob zobrazenia odhadoval.

Tento mockup bol následne prezentovaný oddeleniu CESNET SOC s pozitívnou spätnou väzbou. Jediná získaná výhrada sa týkala zobrazenia pravidiel zo systému ExaFS, ktoré mali nesprávny formát. Išlo o dôsledok chýbajúceho prístupu do systému ExaFS, kvôli ktorému som formát dát iba predpokladal.

3.6 Návrh architektúry systému Amfora

Po ujasnení požiadaviek nasledoval návrh architektúry celého systému s využitím jeho existujúceho základu. Počas návrhu architektúry taktiež prebehol výber použitých technológií (predovšetkým frameworkov).

V používateľskom rozhraní systému Amfora je vhodné niektoré dáta z dôvodu výpočtovej náročnosti načítavať iba v prípade potreby. Aktuálne ide predovšetkým o údaje zo systému Mentat, avšak v prípade možného budúceho rozšírenia môžu pribudnúť ďalšie. Preto som sa rozhodol použiť metódu vykresľovania na strane klienta (client-side rendering, CSR). Tento prístup zároveň umožňuje väčšiu flexibilitu pri nasadení. Implementácia sa preto bude skladať z dvoch väčších celkov – klientskej časti (frontend) a serverovej časti (backend), medzi ktorými bude prebiehať asynchrónna komunikácia.

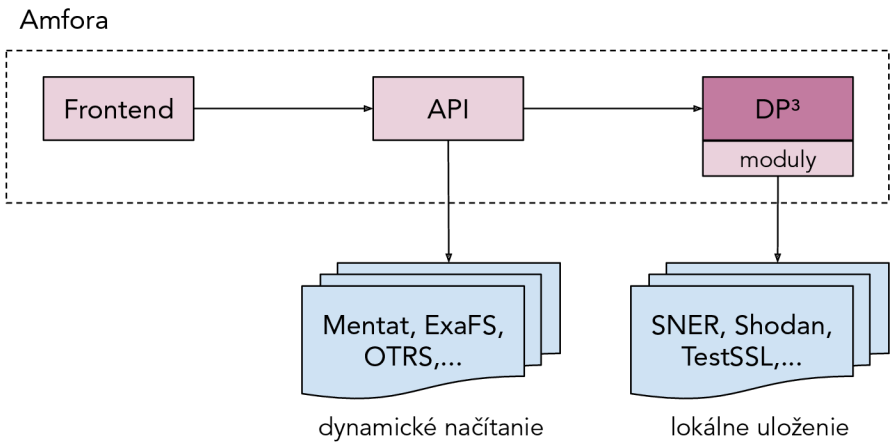
Na klientskej časti (frontende) som sa rozhodoval medzi troma frameworkami – Angular, React a Vue (porovnanie týchto frameworkov sa nachádza v podsekcii 2.3.1). Nakoniec som sa rozhodol využiť framework **Vue** vo verzii 3, pretože je zo spomínaných troch najjednoduchší, najvýkonnejší a má malú veľkosť zostavy. Navyše s ním už mám skúsenosti z predchádzajúcich projektov (vrátane tvorby demo verzie používateľského rozhrania) a po-

užíva sa aj v mnohých iných projektoch združenia CESNET, čím sa rozšíria aj možnosti udržiavania.

Na serverovej časti (backende) som sa rozhodoval medzi frameworkmi Django, Flask a FastAPI (ich porovnanie sa nachádza v podsekcii 2.3.2). V tomto prípade výber jednoznačne padol na framework **FastAPI**, pretože umožňuje pohodlné vytváranie API rozhraní s formátom JSON, automatické generovanie ich dokumentácie a typovú kontrolu parametrov požiadaviek a odpovedí. Tá v prípade systému Amfora nie je nevyhnutná, keďže ide z veľkej časti iba o proxy do iných systémov, avšak v prípade budúceho rozšírenia sa môže hodiť. Framework FastAPI má oporu v software vyvíjanom v združení CESNET a mám s ním aj osobné skúsenosti, keďže som autorom API rozhrania platformy DP^3, ktoré využíva práve tento framework. Podobné rozhodnutie na využitie FastAPI urobil z podobných dôvodov vo svojej práci aj Jakub Man [21] pri rozšírení používateľského rozhrania systému ExaFS.

Z požiadaviek uvedených v sekcii 3.4 vyplýva, že používatelia nevytvárajú ani nemodifikujú stav systému Amfora. Amfora iba využíva ostatné systémy, získava a transformuje dáta v nich uložené podľa požiadaviek používateľov. Z tohto dôvodu sa v architektúre REST[11] využijú iba požiadavky typu `GET`.

Výsledný celkový návrh architektúry reprezentuje obrázok 3.4. Frontend a API sú nové časti, ktoré budú predmetom implementácie v tejto bakalárskej práci.



Obr. 3.4: **Architektúra systému Amfora.** Skladá sa z troch základných častí – frontend, API a DP^3 s aplikačnými modulmi. DP^3 s aplikačnými modulmi reprezentuje existujúci základ systému Amfora. Získavanie údajov z ostatných systémov prebieha dvoma spôsobmi – dynamickým načítaním alebo lokálnym uložením. Dynamické načítanie počas spracovávania požiadavky sa deje v API (backende, serverovej časti) a týka sa interných systémov, ktoré obsahujú históriu údajov a dáta z nich nie sú potrebné pre ďalšie odvodzovanie v platforme DP^3. Lokálne uložené dáta sa sťahujú periodicky prostredníctvom modulov platformy DP^3. To sa týka všetkých ostatných systémov.

[11] REST: Representational State Transfer

Kapitola 4

Implementácia serverovej časti webového rozhrania

Implementácia serverovej časti webového rozhrania (API) je podľa návrhu v kapitole 3 založená na frameworku FastAPI. Jej základnou úlohou je sprostredkovanie komunikácie medzi klientskou časťou webového rozhrania (frontendom) a ostatnými systémami (DP^3, PassiveDNS, ExaFS a ďalšie). Vysokoúrovňovým pohľadom možno považovať toto API za proxy do spomínaných systémov, avšak prijaté dáta transformuje a v niektorých prípadoch navyše overuje ich validitu. Komunikácia v systéme Amfora prebieha jednosmerne – frontend požaduje a zobrazuje prijaté dáta, avšak žiadne údaje do systému nevkladá a ani nemodifikuje. Dáta sa serializujú s využitím formátu JSON.

Zdrojový kód API rozhrania je z dôvodu prehľadnosti a udržovateľnosti rozdelený do niekoľkých častí. Nasledujúci výpis je zjednodušenou reprezentáciou jeho štruktúry:

```
API
├── amfora_api
│   ├── connectors
│   │   ├── dp3.py
│   │   └── ...
│   ├── internal
│   │   └── ...
│   ├── models
│   │   ├── dp3.py
│   │   └── ...
│   └── routers
│       ├── dp3_router.py
│       └── ...
└── main.py
```

Najdôležitejšími časťami sú modul `main.py`, konektory do ostatných systémov (priečinok `connectors`), routery spracúvajúce prichádzajúce požiadavky (priečinok `routers`) a modely odpovedí (priečinok `models`).

`main.py`

Súbor `main.py` je tzv. entrypoint API rozhrania. Vytvára inštanciu frameworku FastAPI, nastavuje CORS[1] politiku podľa konfigurácie, registruje funkcie pre spracovanie výnimiek a pripája routre obsluhujúce jednotlivé využívané systémy. Ide o hlavný súbor celej aplikácie, ktorý sa využíva pri jej spúšťaní pomocou niektorého z ASGI[2] serverov.

Konektory do využívaných systémov

Systémové konektory implementujú všetky aspekty získavania dát z využívaných systémov. Riešia autentifikáciu, smerovanie požiadaviek, spracovanie chýb a vytvárajú vysokoúrovňové rozhranie, ktoré sa ďalej používa v ostatných častiach API, predovšetkým v routroch. Ich hlavný prínos spočíva v oddelení spomínanej funkcionality do samostatných jednotiek. Každý konektor je reprezentovaný triedou. Implementácia obsahuje päť konektorov: do platformy DP^3, a do systémov ExaFS, Mentat, NERD a PassiveDNS, ktoré umožňujú získať:

- **DP^3**:
  - `list_entities`: výpis entít podľa typu entity (so stránkovaním a filtrovaním)
  - `get_entity_snapshots`: výpis snímok entity podľa jej identifikátora a časového rozmedzia
- **ExaFS**:
  - `list_rules`: výpis pravidiel v časovom intervale
- **Mentat**:
  - `get_event_count_for_ip`: počet udalostí pre IP adresu
  - `ui_url_for_event_search_of_ip`: URL adresa pre vyhľadanie detailov udalostí v používateľskom rozhraní Mentat
- **NERD**:
  - `get_ip_detail`: detail IP adresy
  - `ui_url_for_ip_detail`: URL adresa pre zobrazenie detailu IP adresy vo webovom rozhraní
- **PassiveDNS**:
  - `list_domains_for_ip`: zoznam domén pre IP adresu v časovom rozmedzí
  - `ui_url_for_domain`: URL adresa pre vyhľadanie zoznamu domén v používateľskom rozhraní PassiveDNS

[1] CORS: Cross-origin resource sharing
[2] ASGI: Asynchronous Server Gateway Interface

Routery požiadaviek

Spracovanie požiadaviek prichádzajúcich na API rozhranie je opäť rozdelené podľa príslušného systému do Python modulov. Router moduly využívajú triedu `APIRouter` vo frameworku FastAPI, ktorá umožňuje vytvárať samostatné časti API rozhraní. Tie sa následne spájajú v koreňovom module `main.py`.

Okrem systémových routrov implementácia obsahuje aj modul pre koreňový router s cestou `/` vracajúci jednoduchú OK správu (`root_router.py`).

Funkcie pre spracovanie požiadaviek využívajú systémové konektory pre získanie dát. Inštanciácia konektorov prebieha na úrovni príslušného modulu. Pred vrátením dát vo väčšine funkcií prebieha ich transformácia alebo validácia. Významným príkladom je v tomto ohľade platforma DP[3], pri ktorej je spracovanie údajov natoľko komplexné, že sa nachádza v samostatnom module (`internal/dp3_data_processing.py`) a bude opísané v samostatnej podsekcii.

Typová kontrola (modely)

Implementácia serverovej časti webového rozhrania naplno využíva možnosti typovej kontroly frameworku FastAPI, resp. knižnice Pydantic[3]. S využitím tzv. type hints jazyka Python je zaistená kontrola typov pri parametroch požiadaviek. Pre každý formát odpovede je taktiež vytvorený a vynucovaný jej typovaný model. Tieto modely sú špecifické pre každú funkciu pre spracovanie požiadaviek. Kvôli deduplikácii kódu je tam, kde to je možné, využívaná dedičnosť typov a využité sú aj šablóny (`TypeVar` a `Generic` v jazyku Python), čoho príkladom sú generické odpovede pre úspešnú operáciu (`SuccessResponse`), chybu (`ErrorResponse`) či stránkovaný zoznam položiek (`PaginatedItemsResponse`).

Typovanie požiadaviek a odpovedí zaisťuje spoľahlivosť a bezpečnosť poskytovaných informácií – pre API rozhranie aj pre jeho používateľa (frontend). Zároveň umožňuje automatickú generáciu dokumentácie API rozhrania, ktorej ukážka sa nachádza na obrázku 4.1.

Spracovanie chýb

Spracovanie chýb je dvojakého charakteru. V rámci validácie modelov vo frameworku FastAPI sú implicitne vytvárané chybové hlásenia vo formáte JSON v prípade nevalidnej požiadavky (nesprávny typ alebo chýbajúci parameter).

V systéme Amfora je tento mechanizmus rozšírený o spracovanie chýb pochádzajúcich z konektorov. Modul `routers/exception_handlers.py` implementuje vytváranie chybových hlásení pri výskyte výnimky v systémovom konektore počas spracúvania prichádzajúcej požiadavky. Ide o chyby pripojenia či nesprávne prihlasovacie údaje – dôsledok nesprávnej konfigurácie alebo iného problému na strane servera. Ako návratový kód je použitý `502 Bad Gateway`.

Takto je zaistený unifikovaný prístup k spracovaniu výnimiek v celej aplikácii a odstránená duplikácia kódu na spracovanie chybových stavov.

Defined state source (detekcia anomálií)

Ako je uvedené v kapitole 3, pôvodne mala byť súčasťou implementácie napojenie systému Amfora na systém NetBox prevádzkovaný združením CESNET. NetBox mal poskytovať údaje o staticky definovanom stave jednotlivých zariadení a služieb a mal slúžiť ako

[3]Pydantic: https://docs.pydantic.dev/2.7/

DP3

GET /data/dp3/ips List Ips

GET /data/dp3/ip/{eid} Get Ip Detail

Gets IP detail (snapshots)

Parameters:

- eid (IPvAnyAddress): IP address (entity ID).
- ts (datetime, optional): Timestamp of time interval end. Defaults to current time.

Parameters | Try it out

Name	Description
eid * required string($ipvanyaddress) (path)	eid
ts string($date-time) (query)	Default value : 2024-04-25T11:46:28.626258 2024-04-25T11:46:28.626258

Responses

Code	Description	Links
200	Successful Response Media type application/json Controls Accept header. Example Value | Schema	No links
422	Validation Error Media type	No links


```
[
  {
    "ts": "2024-04-28T19:30:54.957Z",
    "s": {
      "ip": "string",
      "ip_groups": {
        "actual": [
          {
            "v": "string",
            "c": 1,
            "link": "string",
            "src": "string",
            "src_link": "string"
          }
        ],
        "defined": "string",
        "match": true
      },
      "services": {
        "actual": [
          {
            "v": 0,
            "c": 1,
            "link": "string",
            "src": "string",
            "src_link": "string"
          }
        ],
        "defined": 0,
```


Obr. 4.1: **Ukážka dokumentácie API rozhrania systému Amfora.** Vygenerované automaticky frameworkom FastAPI. Výsek zobrazuje endpoint `/data/dp3/ip/{eid}`, ktorý slúži na získavanie snímok špecifikovanej IP adresy v časovom intervale. Uvedený je krátky popis, textové polia na zadanie parametrov, formát a ukážka odpovede.

tzv. source of truth. Systém Amfora mal statický stav porovnávať s reálnym a hlásiť nezrovnalosti. Keďže systém NetBox ešte neobsahuje potrebné údaje, od tohto plánu sa upustilo a implementácia API rozhrania počíta iba s jeho budúcim napojením.

Za týmto účelom bol vytvorený Python balíček označený `defined_state_source` (DSS). Defined state source bude pravdepodobne potrebný aj pre moduly platformy DP[3], preto je oddelený od zdrojového kódu systému Amfora do samostatného balíčka. Aktuálne slúži ako placeholder.

Významnejším prispôsobením na budúce napojenie je využitie generických štruktúr dát, ktoré umožňujú vkladanie aktuálneho aj definovaného stavu k jednej hodnote a posudzovanie ich validity na základe zadaných pravidiel. Základom sú štruktúry `Data` a `DSSResult` v `models/generic.py`. Implementačne ide o modely knižnice Pydantic.

Štruktúra `Data` reprezentuje generickú jednotku dát. Obsahuje samotnú hodnotu, dôveryhodnosť (confidence), odkaz, identifikáciu zdroja a odkaz na zdroj. Výhoda tejto štruktúry spočíva v jednoduchšom spracovaní a zobrazení komplexných údajov na frontende. Použitie ilustruje výpis 4.1.

```
{
  "v": "OpenWrt",
  "c": 0.782,
  "link": "https://openwrt.org/",
  "src": "ADiCT",
  "src_link": "https://adict.example.com/dashboard/10.0.0.1"
}
```

Výpis 4.1: **Príklad inštancie modelu `Data`.** Hodnota `OpenWrt` má dôveryhodnosť 78,2 % a odkazuje na domovskú stránku projektu OpenWrt. Hodnota pochádza zo systému ADiCT s uvedeným odkazom na zdroj. Serializované do formátu JSON.

Štruktúra `DSSResult` pozostáva z aktuálnej a definovanej hodnoty, a označenia validity. Opäť ide o generický model. Tvorí základ pre validáciu v celom API rozhraní systému Amfora. Jej príklad sa nachádza na výpise 4.2.

```
{
  "actual": "Debian",
  "defined": "Fedora",
  "match": false
}
```

Výpis 4.2: **Príklad inštancie modelu `DSSResult`.** Reálna hodnota je `Debian`, hoci definovaná je `Fedora`, preto zhoda (`match`) je `false`. Serializované do formátu JSON.

K obom spomínaným štruktúram sú vytvorené pomocné konštruktory `_data` a `_dss_result`, ktoré zjednodušujú vytváranie a serializáciu inštancií týchto štruktúr. Konštruktor `_dss_result` navyše implementuje sadu jednoduchých validačných funkcií, ktoré nastavujú indikátor `match` podľa reálnej a definovanej hodnoty.

Aj napriek absencii napojenia na systém NetBox, ktorý by poskytoval údaje o definovanom stave, zdrojový kód obsahuje niekoľko pravidiel pre overovanie validity, ktoré budú opísané v nasledujúcej podsekcii.

Spracovanie dát z DP^3

Spracovanie dát z platformy DP^3 tvorí najkomplexnejšiu časť celého API rozhrania. Jeho implementácia sa nachádza v module `interal/dp3_data_processing.py`.

Každá z typov entít — IP adresa (`ip`), skupina IP adries (`ip_group`) a služba (`service`) — má samostatnú funkciu `process_{ip,ip_group,service}_detail`, ktorá slúži ako vstupný bod pre spracovanie príslušných dát z platformy DP^3. Vstupom týchto funkcií je vždy jedna snímka entity. Obsah tejto snímky je postupne spracovaný. Hodnoty atribútov sa transformujú tak, aby využívali štruktúry `Data` a `DSSResult` opísané v predchádzajúcej podsekcii a sémanticky podobné atribúty sa spájajú do jedného (napríklad viaceré atribúty s operačným systémom sa spájajú do jediného atribútu `os`). V niektorých prípadoch sa údaje navyše zoraďujú alebo menia formát.

Spracovanie snímky skupiny IP adries využíva aj snímky jednotlivých IP adries, ktoré sú súčasťou skupiny. V tomto prípade transformácia dát využíva funkciu `process_ip_detail_overview`, ktorá zhodnocuje počet validných a nevalidných atribútov príslušnej IP adresy.

Implementácia spracovanie dát z DP^3 obsahuje niekoľko pravidiel pre overovanie ich validity. Na tento účel sa využíva rozhranie pôvodne určené na validáciu dátami zo systému NetBox. Všetky spomínané atribúty sú súčasťou typu entity služba (`service`).

- `vuln` a `cve`: zoznam hrozieb musí byť prázdny
- `tls_grade`: známky zabezpečenia protokolu TLS musia byť `A+`, `A`, `A-` alebo `B` (detaily známkovania systému Auror sú uvedené v sekcii 2.1.3)
- `tls_poodle`[4]: zraniteľnosť na POODLE útok nesmie existovať
- `tls_freak_vulnerable`[5]: zraniteľnosť na FREAK útok nesmie existovať
- `tls_cert_valid`: certifikáty musia byť validné
- `tls_cert_expired`: certifikáty nesmú byť vyexspirované

Vyhľadávanie v entitách

Vyhľadávanie v entitách prijíma textový vstup, ktorého príkladom sú:

- `1.2. port:22`: vyhľadanie služieb s číslom portu 22 na IP adrese obsahujúcej `1.2.`
- `ip:1.2.3.4 product:Apache`: vyhľadanie služieb na IP adrese obsahujúcej `1.2.3.4` so softwarovým produktom `Apache`

Podporované filtrovacie atribúty sú IP adresa (`ip`), číslo portu (`port`), hostname podľa reverzného záznamu (`hostname`), softwarový produkt (`product`) a verzia softwarového produktu (`version`). Časť textu pred selektorom sa chápe ako filter pre identifikátor entity.

Spracovanie textového vyhľadávania IP adries a služieb je implementované v routri platformy DP^3. Textový reťazec je tokenizovaný a spracúvaný po častiach. Neexistujúce filtrovacie atribúty sú ignorované. Je vytvorená štruktúra so všetkými filtrami. Ak filtre neobmedzujú typ entity (napríklad `product` implikuje vyhľadávanie iba medzi službami),

[4]POODLE: Padding Oracle On Downgraded Legacy Encryption
[5]FREAK: Factoring RSA Export Keys

odosiela sa požiadavka na vyhľadanie IP adries a služieb do platformy DP[3]. Z výsledkov sa vyberú maximálne prvých `n` entít každého typu (nastaviteľné v konfigurácii), pretransformujú sa ich atribúty a vrátia sa v odpovedi na požiadavku.

Smerovanie

Smerovanie požiadaviek prebieha na základe zhody cesty a využíva možnosti frameworku FastAPI. Cesty všetkých endpointov majú prefix `/data` nasledovaný identifikátorom služby, o ktorú ide, na ďalšej úrovni. Tento princíp hierarchického členenia menného priestoru je prirodzený a integrovaný aj do triedy `APIRouter` frameworku FastAPI. Výnimku zo spomínaného formátu cesty tvorí koreňový endpoint (`/`), ktorý slúži na overenie spustenia API rozhrania.

Konfigurácia

Konfigurácia API rozhrania pozostáva z jediného súboru vo formáte YAML[6]. Jeho načítanie a spracovanie je implementované v module `internal/config.py`. Aj konfigurácia má definovaný model a jej validita sa overuje s využitím knižnice Pydantic.

Súčasťou konfigurácie sú predovšetkým URL adresy jednotlivých využívaných systémov a ich prihlasovacie údaje, nastavenia CORS politiky a trvanie pohyblivého okna v zobrazení histórie.

[6]YAML: https://yaml.org/

Kapitola 5

Implementácia klientskej časti webového rozhrania

Klientská časť webového rozhrania (frontend) vychádza z návrhu v kapitole 3 a je založená na frameworku Vue verzie 3 s využitím jazyka JavaScript. Používa serverovú časť webového rozhrania opísanú v kapitole 4, ktorá sprostredkúva získavanie dát zo všetkých využívaných systémov vrátane DP[3]. Získané dáta prehľadne zobrazuje v používateľskom rozhraní.

5.1 Popis zdrojového kódu klientskej časti webového rozhrania

Súborová štruktúra klientskej časti vychádza zo šablóny frameworku Vue, ktorá bola vytvorená príkazom `npm create vue@latest`. Táto šablóna bola následne rozširovaná podľa požiadaviek na systém. Niektoré časti boli čiastočne prebrané z demo verzie používateľského rozhrania. V zjednodušenej podobe je výsledkom táto súborová štruktúra:

```
frontend
└── src
    ├── components
    │   ├── DataWithSrc.vue
    │   ├── NavigationBar.vue
    │   ├── PaginatedListing.vue
    │   └── ...
    ├── router
    │   └── index.js
    ├── stores
    │   └── requestsState.js
    ├── views
    │   ├── HomeView.vue
    │   ├── IPView.vue
    │   └── ...
    ├── App.vue
    └── main.js
```

Zdrojový kód implementácie pozostáva prevažne z modulov jazyka JavaScript a komponentov frameworku Vue. Tieto komponenty predstavujú jednotky znovupoužiteľných prvkov používateľského rozhrania. Každý komponent pozostáva z troch častí – HTML šablóny

(prvok `<template>`), kaskádových štýlov definujúcich vzhľad (prvok `<style>`) a definície správania v jazyku JavaScript (prvok `<script>`).

V nasledujúcich podsekciách bližšie opíšem obsah a význam jednotlivých položiek súborovej štruktúry.

`main.js`

Súbor `main.js` je základným modulom aplikácie. Importuje väčšinu využívaných knižníc, kaskádové štýly, súčasti frameworku Vue, spracúva konfiguráciu poskytovanú zostavovacím (build) systémom a nastavuje využitie frameworku Vue v klientskej verzii systému Amfora. Obsahuje taktiež import základného Vue komponentu aplikácie `App.vue`. Ide o východzí bod zostavy.

`App.vue`

Súbor `App.vue` reprezentuje koreňový Vue komponent. Integruje komponenty základného rozloženia aplikácie – navbar, zobrazenie aktuálnej stránky z routra a ukazovateľ načítavania.

Komponenty

Frontend systému Amfora je rozdelený do množstva komponentov. Tento prístup zjednodušuje vývoj a umožňuje znovupoužiteľnosť v rôznych častiach webového rozhrania. Implementované komponenty frameworku Vue sa nachádzajú v priečinku `components`. Patrí sem navbar (`NavigationBar.vue`), pole pre vyhľadávanie entít (`EntitySearch.vue`), všeobecná tabuľka výsledkov so stránkovaním (`PaginatedListing.vue`) či množstvo komponentov na zobrazenie dát z využívaných systémov.

Za zmienku stojí komponent `DataWithSrc.vue`, ktorý všeobecným spôsobom zobrazuje reprezentáciu štruktúry `Data` zo serverovej časti webového rozhrania (opísané v kapitole 4). Ukážka jeho použitia sa nachádza na obrázku 5.1.

Obr. 5.1: **Ukážka komponentu `DataWithSrc`.** Zobrazené dáta pochádzajú z príkladu inštancie modelu `Data` vo výpise 4.1.

Pohľady

Pohľady (views) sú striktne taktiež komponenty frameworku Vue, avšak ich významom nie je vytváranie znovupoužiteľných celkov. Každý pohľad totiž reprezentuje jednu konkrétnu stránku používateľského rozhrania. Zobrazenie správneho pohľadu zaisťuje Vue Router. Vue Router[1] je knižnica umožňujúca vývoj SPA[2] vo frameworku Vue. Zobrazuje vybranú stránku na základe URL adresy v prehliadači a umožňuje navigáciu medzi stránkami bez obnovenia.

[1] Vue Router: https://router.vuejs.org/
[2] SPA: single-page application

Implementácia obsahuje 6 pohľadov: pre domovskú stránku (`HomeView.vue`), zoznamy entít (`IPsView.vue` pre IP adresy, `IPGroupsView.vue` pre skupiny IP adries, `ServicesView.vue` pre služby) a detaily entít (`IPView.vue` pre IP adresy a `ServiceView.vue` pre služby). Domovská stránka je jednoduchým rozcestníkom. Zoznamy entít obsahujú tabuľku so stránkovaním a vlastný filter pre vyhľadávanie podľa rôznych atribútov. Detaily entít obsahujú označenie entity, dáta z platformy DP[3] a dynamicky načítané dáta z ostatných systémov. Detail skupiny IP adries nemá samostatnú stránku, pretože k skupinám IP adries nie je k dispozícii dostatočné množstvo dát.

Úložiská stavu (`stores`)

Stav, ktorý je potrebné propagovať do celej aplikácie frameworku Vue a reagovať dynamicky na jeho zmeny, sa často implementuje s využitím úložísk (stores). Oficiálnym riešením je v tomto ohľade knižnica Pinia[3]. Takéto úložisko umožňuje jednoduchý a unifikovaný prístup k premenným a implementáciu akcií na nich naviazaných.

Implementácia frontendu systému Amfora obsahuje jedno úložisko stavu – pre ukazovateľ načítavania.

Ukazovateľ načítavania

Ukazovateľ načítavania využíva úložisko stavu (store) na sledovanie priebehu načítavania asynchrónnych volaní na získanie dát z API rozhrania. Počas prebiehajúcej komunikácie medzi frontendom a API rozhraním sa zobrazuje v aplikácii spinner. Zároveň sa zjednocuje spracúvanie chýb a vytváranie chybových hlásení. V prípade chybového návratového kódu je používateľ informovaný prostredníctvom tzv. toastu (malého vyskakovacieho okna). Takto je možné sledovať všetky (aj paralelné) požiadavky a informovať o ich priebehu používateľa. Implementácia úložiska sa nachádza v súbore `stores/requestsState.js`) a prvky používateľského rozhrania vo Vue komponente `components/RequestsState.vue`.

Konfigurácia

Snahou pri implementácii klientskej časti webového rozhrania bolo minimalizovať množstvo konfigurácie a využiť serverovú časť ako jediný zdroj dát. Takýto prístup minimalizuje riziko nekompatibilných nastavení, zjednodušuje nasadenie a postup pri zmenách. Z API rozhrania sa preto získavajú aj statické URL adresy do používateľských rozhraní využívaných systémov.

Celá konfigurácia frontendu pozostáva z dvoch premenných prostredia:

- `VITE_API_URL`: URL adresa na API rozhranie systému Amfora
- `VITE_LOGOUT_URL`: URL adresa na odhlásenie z poskytovateľa identity, ktorú odkazuje odhlasovacie tlačidlo v navbare (voliteľné)

Knižnice

Pri implementácii frontendu som využil viaceré open-source knižnice. Medzi najdôležitejšie patria:

- **axios**: knižnica na vykonávanie HTTP požiadaviek

[3]Pinia: https://pinia.vuejs.org/

- **Bootstrap**: CSS framework, ktorý je využitý v celom používateľskom rozhraní
- **ChartJS**: knižnica pre tvorbu interaktívnych grafov využitá pri zobrazovaní histórie hodnôt atribútov (komponent `components/TableAttribute.vue`)
- **DayJS**: malá knižnica zjednodušujúca prácu s dátumami

Zároveň som v komponentoch frameworku Vue využíval pri vytváraní kaskádových štýlov jazyk SCSS[4].

5.2 Výsledné používateľské rozhranie

Výsledné používateľské rozhranie — až na niekoľko grafických odchýlok — zodpovedá mockupu, ktorý bol vytvorený vo fáze návrhu (sekcia 3.5). Pozostáva zo šiestich stránok: hlavná stránka, prehľady entít (IP adries, skupín IP adries a služieb) a detaily entít (IP adresy a služby). Všetky stránky obsahujú navbar s navigáciou, poľom pre vyhľadanie entít a odhlasovacím tlačidlom, a ukazovateľ načítavania. Pole pre vyhľadanie entít zobrazuje medzi výsledkami vždy maximálne `n` prvých IP adries a služieb. Pri IP adresách sa zobrazuje aj hostname a pri službách názov a verzia software (ak sú známe). Odhlasovacie tlačidlo je prítomné, ak je nakonfigurovaný odkaz na odhlásenie z poskytovateľa identity.

Hlavná stránka je jednoduchým rozcestníkom na prehľady entít.

Základný prvok **prehľadov entít** je tabuľka so zoznamom entít. Jej obsah a rozloženie závisí na vybranom type entity. Pod tabuľkou sa nachádza stránkovanie. Nad tabuľkou sa nachádza filter pre hodnoty atribútov, ktorý je opäť závislý na zvolenom type entity. Identifikátory vždy odkazujú na detail danej entity. Výnimku tvorí prehľad skupín IP adries, pri ktorých nie je dostatok dát, preto samostatný detail neexistuje.

Detaily pozostávajú v základe zo selektoru na výber dátumu a času, identifikácie entity v nadpise a sekcie s dátami z platformy DP^3. V prípade detailu IP adresy nasleduje ešte sekcia s odkazmi na služby patriace danej IP adrese a sekcie s dátami zo systémov PassiveDNS, ExaFS, NERD, Mentat a odkaz do systému OTRS. Odkaz do systému FTAS v implementácii chýba, pretože nie je k dispozícii dostatok informácií na jeho skonštruovanie. Dáta z platformy DP^3 využívajú štruktúru `Data`, ktorá bola opísaná v kapitole 4 a zobrazujú sa pomocou komponentu `DataWithSrc.vue`, ktorý bol opísaný v sekcii 5.1. Každý atribút obsahuje riadok s názvom atribútu, hodnotou, ikonou validity, popisom chyby pri validácii (ak nastala) a šípkou na rozbalenie grafu s históriou hodnôt. Ikona validity má tri stavy – zelená „fajka“ pre validnú hodnotu, červený „krížik“ pre nevalidnú hodnotu a sivý otáznik pre hodnotu s neznámou validitou. Graf zachytáva vývoj za konfiguračne nastaviteľné časové obdobie. Selektor na výber dátumu a času umožňuje vybrať relatívny (pomocou časovej osi) aj absolútny čas (priamym zadaním do poľa).

V detaile aj v prehľade entít sa červenou farbou podfarbujú riadky s nevalidnou hodnotou atribútu alebo nevalidnou entitou (nevalidná je aspoň jedna hodnota atribútu).

Nasledujúca časť obsahuje ukážky jednotlivých stránok používateľského rozhrania na obrázkoch 5.2, 5.3, 5.4, 5.5 a 5.6. Vo všetkých prípadoch obsahujú anonymizované dáta (náhodné IP adresy, hostname a podobne).

[4]SCSS: https://sass-lang.com/

Amfora IPs IP groups Services Search Logout

IP addresses

Filters

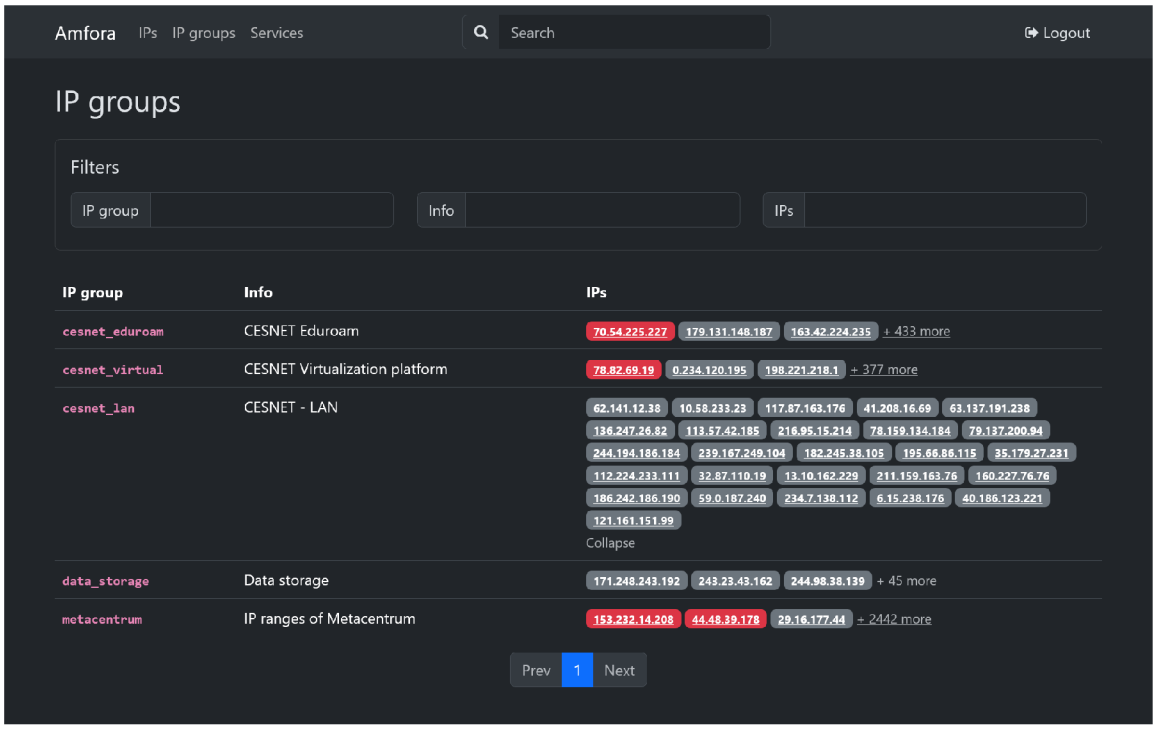
IP | Hostname fa | IP group

OS (ADiCT) | Device type (ADiCT) | Services

IP address	Hostname	Group	OS	Device type	Services
120.202.216.69	wx.example.com				
234.137.155.95	ef.example.com		Linux (83 %)		
93.239.4.187	sg.example.com				
219.30.238.168	rm.example.com				
163.162.48.253	xa.example.com		Windows (81 %)	Server (73 %)	23 80
85.153.196.45	cu.example.com				443 993 995
126.114.22.34	no.example.com				
226.229.131.208	bj.example.com				
117.132.21.246	xq.example.com				
27.148.240.221	mw.example.com	metacentrum	Linux (83 %)		22 111 22222
3.25.178.161	oe.example.com	metacentrum	Linux (83 %)		22 111 22222
162.26.155.72	ac.example.com	metacentrum	Linux (83 %)		22 111 22222
84.165.10.24	zw.example.com	metacentrum	Linux (78 %)		8008
92.163.151.138	qz.example.com	metacentrum	Windows (82 %)		443
72.76.54.2	rk.example.com	metacentrum	Windows (81 %)	Server (91 %)	22 25 443 587 993 + 1 more
20.170.30.246	cx.example.com	metacentrum	Linux Ubuntu (76 %)	Server (73 %)	22 3000
75.129.25.227	ot.example.com	metacentrum	Android (76 %)	Server (73 %)	8008
18.131.78.105	xj.example.com	metacentrum	Windows (71 %)		80 443 8008

Prev 1 Next

Obr. 5.2: **Ukážka prehľadu IP adries v používateľskom rozhraní systému Amfora.** Prehľad obsahuje filtre pre IP adresu, hostname, názov skupiny IP adries, operačný systém podľa systému ADiCT, typ zariadenia podľa systému ADiCT a identifikátor služby. Tabuľka obsahuje vymenované položky ako stĺpce. V ukážke sa nachádzajú IP adresy s určeným hostname. Druhá polovica patrí do skupiny `metacentrum`. K väčšine IP adries bol zistený operačný systém s dôveryhodnosťou na 71 % až 83 %. Štyri IP adresy boli identifikované ako servery. Stĺpec `Services` obsahuje zoznam portov s bežiacimi službami.



Obr. 5.3: **Ukážka prehľadu skupín IP adries v používateľskom rozhraní systému Amfora.** Tabuľka a podporované filtre opäť obsahujú rovnaké položky. Skupina IP adries má identifikátor, názov a zoznam IP adries, ktoré do nej patria. Pri obsiahnutých IP adresách prebieha validácia ich stavu. Nevalidné adresy sa zobrazujú ako prvé a sú podfarbené červenou farbou. Zoznam je možné expandovať.

Amfora IPs IP groups Services Search Logout

Services

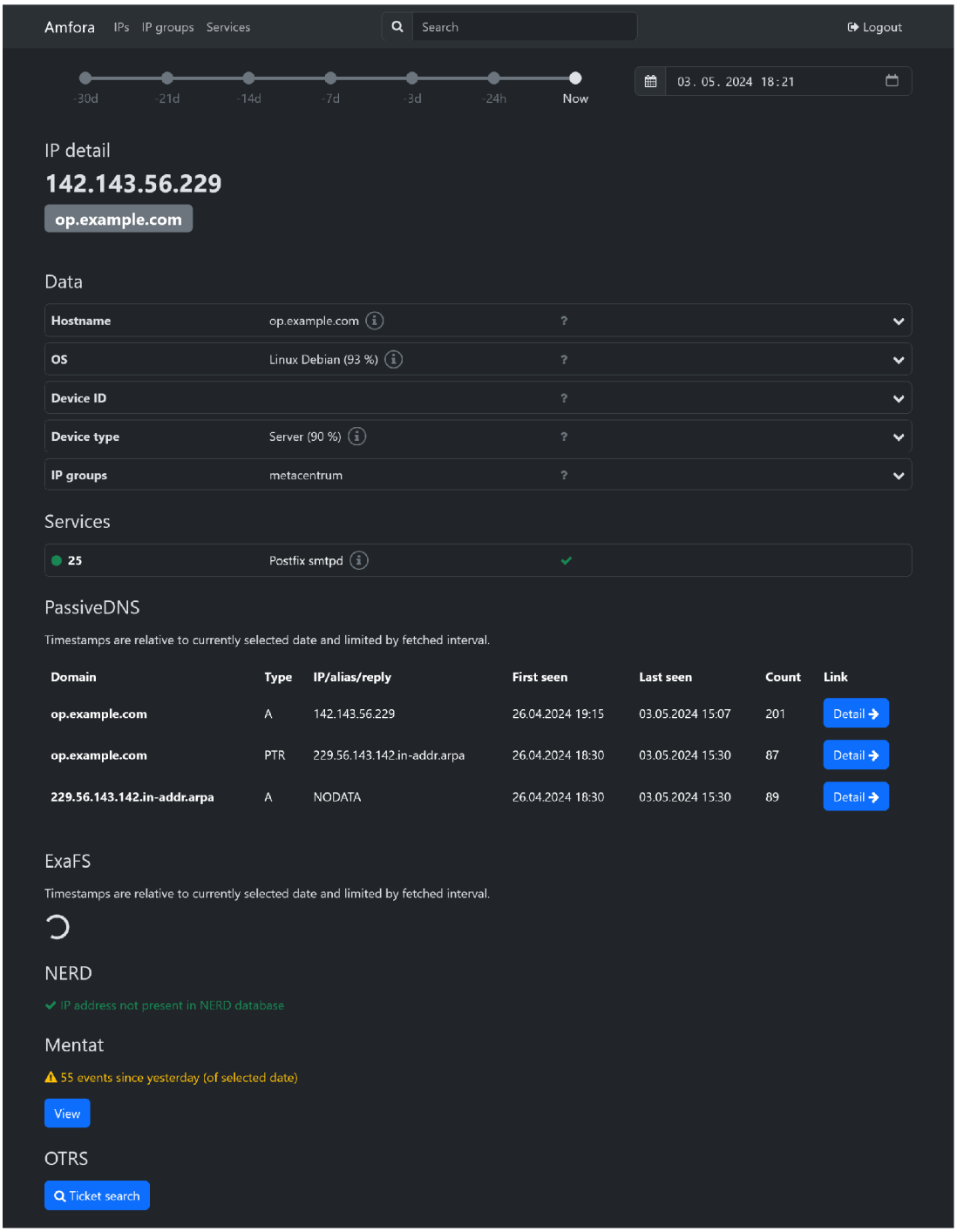
Filters

IP:port | SW (SNER) | IP banner OS

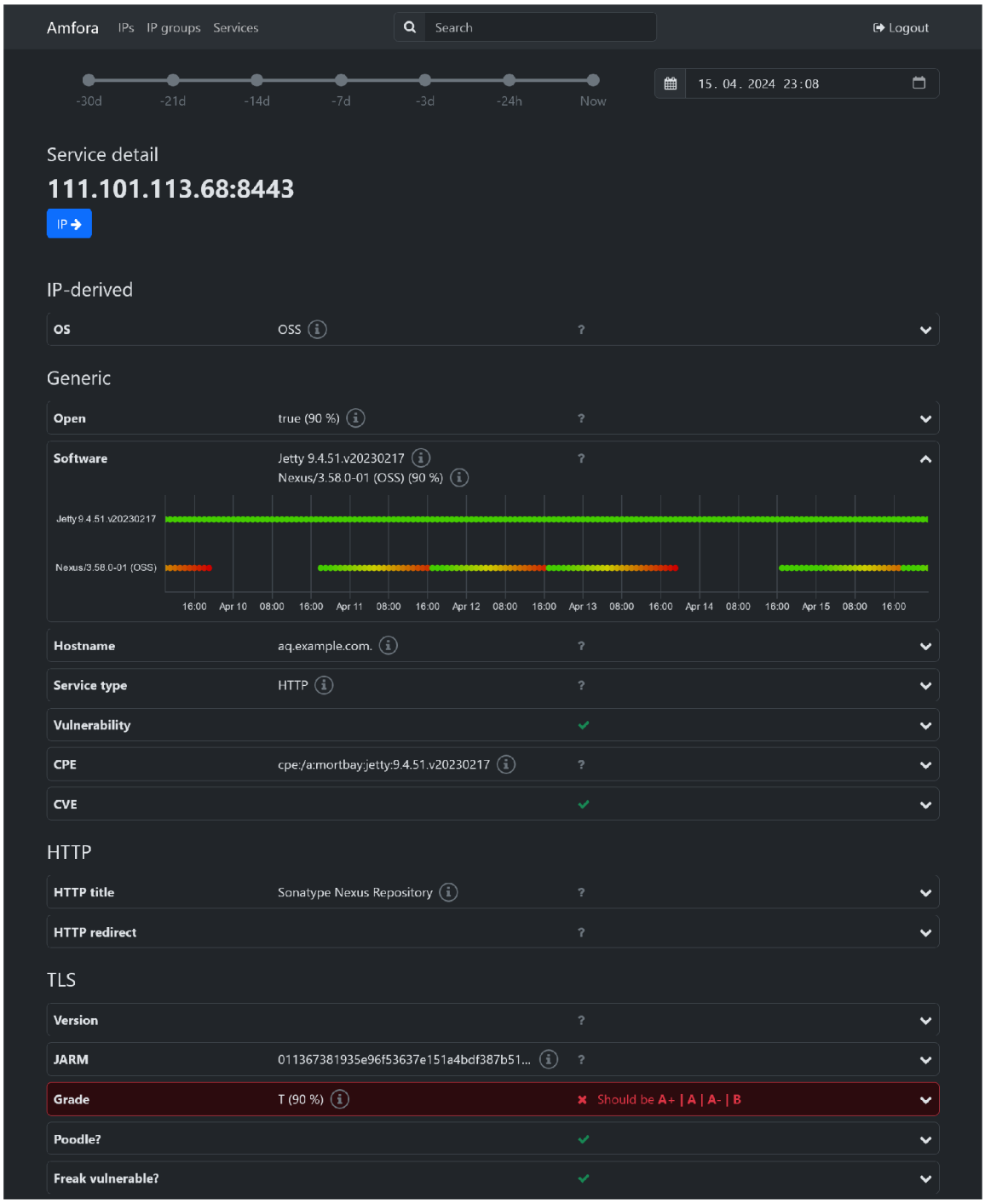
IP:port	Software	Hostname	IP OS
253.140.99.28 22	OpenSSH (96 %)		
60.207.29.91 22	OpenSSH (96 %)		
153.132.91.183 22	OpenSSH (96 %)		
211.205.204.91 443	Apache httpd (90 %) Apache httpd 2.4.38	aa.example.com aa.example.com.	
239.172.247.43 22	OpenSSH (96 %)		
241.86.124.73 25	Postfix smtpd	iu.example.com	
107.250.20.32 22	OpenSSH (96 %)		
105.51.68.153 22	OpenSSH (96 %)		
255.92.226.184 636		kz.example.com	
140.105.248.12 21	Synology DiskStation NAS ftpd (90 %)		
207.56.88.218 22	OpenSSH (96 %)		
56.223.14.178 2222	OpenSSH (98 %)		
125.186.39.69 3306	Dionaea Honeypot (93 %)		
213.45.128.29 123			
57.205.84.195 443	nginx (97 %)		
96.78.117.176 4000	NoMachine NX Server remote desktop (80 %) NoMachine NX Server remote desktop 7.7.4		
76.182.21.251 161	ciscoSystems (84 %)		
57.238.66.62 179			
76.93.64.99 443	Apache httpd (89 %) Apache httpd 2.4.57	pp.example.com.	Debian Debian (81 %)
54.51.180.2 80	Apache httpd (90 %) Apache httpd 2.4.57	og.example.com	

Prev 1 ... 14 15 16 ... 305 Next

Obr. 5.4: **Ukážka prehľadu služieb v používateľskom rozhraní systému Amfora.** Prehľad služieb obsahuje tabuľku so stĺpcami: identifikátor služby (IP adresa a port), softwarový produkt, hostname podľa dát z príslušnej služby a operačný systém získaný z IP adresy. Filtre využívajú podobné atribúty. Ukážka obsahuje 20 služieb. Na väčšine z nich beží OpenSSH alebo Apache server. Operačný systém je známy iba pri jednej z IP adries a ide o Debian na základe dvoch zdrojov. Dve služby majú nevalidné hodnoty atribútov, preto sú podfarbené červenou farbou.



Obr. 5.5: **Ukážka detailu IP adresy v používateľskom rozhraní systému Amfora.** Ukážka zobrazuje detail IP adresy `142.143.56.229` s hostname `op.example.com`. Na stroji s touto IP adresou je pravdepodobne používaný operačný systém Debian a ide o server. IP adresa patrí do skupiny `metacentrum`. Beží na nej služba na porte 25 a na základe údajov zo skenovania ide o Postfix server s validnými hodnotami atribútov. Podľa systému PassiveDNS sa k IP adrese viazalo za posledných 7 dní iba doménové meno `op.example.com`. K dátam zo systému ExaFS prebieha načítavanie. V databáze systému NERD sa IP adresa nenachádza. Systém Mentat vykazuje 55 bezpečnostných udalostí za posledných 24 hodín. Časové údaje sú relatívne voči aktuálnemu času.

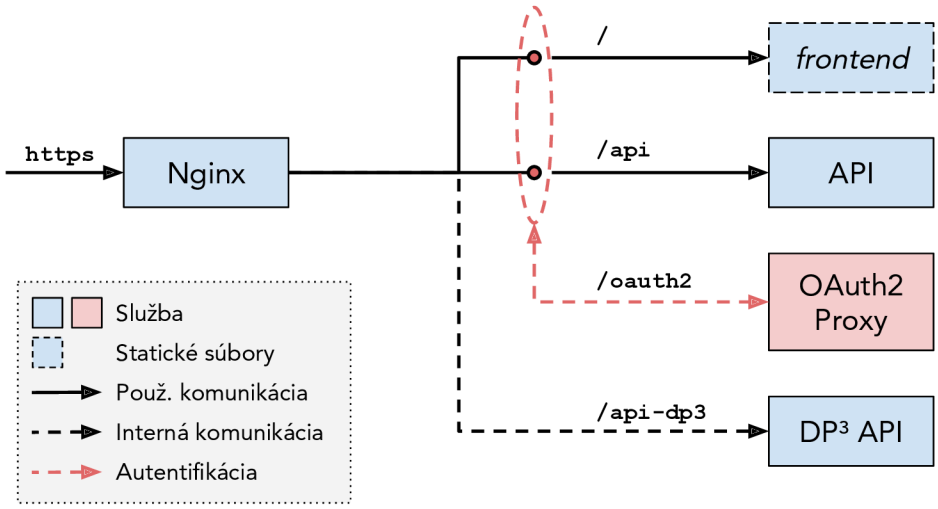


Obr. 5.6: **Ukážka detailu služby v používateľskom rozhraní systému Amfora (skrátená).** Ide o detail služby na IP adrese `111.101.113.68` a porte 8443. Na tejto IP adrese sa používa operačný systém OSS. Služba využíva produkt Jetty a/alebo Nexus – údaje z dvoch skenovacích systémov sa rôznia. Graf zobrazuje časový vývoj týchto hodnôt. Zelené body reprezentujú hodnoty so 100 % dôveryhodnosťou a červené s 0 %. Služba má nastavený hostname `aq.example.com` a ide o HTTP server. Zoznam zraniteľností a CVE je prázdny, čo je vyhodnotené ako validný stav. Bol zachytený aj titulok HTTP a JARM odtlačok protokolu TLS. Známka TLS je na 90 % `T` (nedôveryhodný certifikát), čo nie je validný stav. Implementácia protokolu TLS nie je zraniteľná na POODLE a FREAK útoky.

Kapitola 6

Nasadenie

Nasadenie systému Amfora prebehlo v spolupráci s Václavom Bartošom z Liberouter tímu združenia CESNET na server `amfora.liberouter.org`. Jeho schému ilustruje obrázok 6.1. Nasadená verzia bola odovzdaná oddeleniu CESNET SOC, ktoré systém Amfora už využíva pri svojej práci.



Obr. 6.1: **Schéma nasadeného systému Amfora.** Nasadený systém pozostáva z webového serveru Nginx, frontendu a API rozhrania systému Amfora, platformy DP^3 a služby OAuth2 Proxy. Server Nginx plní úlohu reverse proxy pre API rozhranie systému Amfora, OAuth2 Proxy a pre API rozhranie platformy DP^3 (prístupné iba z interných adresných rozsahov). Taktiež je serverom pre statické súbory frontendu. Frontend a API rozhranie systému Amfora vyžadujú pre prístup OIDC autorizáciu, ktorú zabezpečuje OAuth2 Proxy.

Nginx

Server Nginx[1] som zvolil, pretože ide o jednu z najpopulárnejších možností, je výkonný, podporuje potrebné funkcie a mám s ním skúsenosti. V tomto prípade plní dve úlohy – je serverom pre statické súbory frontendu a zároveň proxy pre ostatné nasadené služby. Statické súbory frontendu sú výsledkom procesu produkčného zostavenia a kvôli využitiu Vue Router

[1] Nginx: https://nginx.org/en/

vyžadujú smerovanie požiadaviek na všetky stránky na koreňový súbor `index.html`. To je zaistené direktívou `index` v konfigurácii Nginx. Nginx rovnako zaisťuje termináciu HTTPS s protokolom HTTP/2. HTTPS využíva certifikát od certifikačnej autority Let's Encrypt. Na jeho získanie a automatické obnovovanie som na server nasadil nástroj Certbot[2]. Použitie HTTPS namiesto nezabezpečeného HTTP je vynútené automatickým presmerovaním stavovým kódom `301 Moved Permanently`.

Autorizácia

Autorizačný mechanizmus je nevyhnutný, keďže systém Amfora obsahuje množstvo citlivých údajov. Pri nasadení bol požadovaný protokol OIDC[3] (založený na OAuth 2.0) s využitím služby Perun[4]. V požiadavkách od oddelenia CESNET SOC bola špecifikovaná iba jedna úroveň prístupových práv, preto som sa rozhodol namiesto vývoja interného autorizačného mechanizmu využiť existujúcu a overenú službu, ktorá mechanizmus zaistí v kooperácii s webovým serverom. Vybral som službu OAuth2 Proxy[5], ktorá umožňuje prepojenie s využitím `auth_request` direktívy serveru Nginx. Pri návšteve webového rozhrania je používateľ automaticky presmerovaný na prihlasovaciu stránku služby Perun.

Platforma DP^3

Nasadenie platformy DP^3 bolo realizované už v procese vývoja základu systému Amfora. Súčasti platformy — konkrétne worker procesy, API rozhranie a vstupný modul zo systému Shodan — bežia pod správcom procesov Supervisor[6]. Konfigurácia a postupy zodpovedajú dokumentácii platformy[7].

Frontend

Nasadenie frontendu webového rozhrania spočívalo v inštalácii potrebných balíčkov a vytvorení produkčnej zostavy (statických súborov pre webový server). Nainštalovať potrebné balíčky je možné príkazom `npm install`. Na vytvorenie produkčnej zostavy slúži príkaz `npm run build`. Oba príkazy vyžadujú systémovú inštaláciu behového prostredia Node.js[8] so správcom balíčkov npm[9]. Vytvorené súbory sa nachádzajú v priečinku `dist`.

API rozhranie

API rozhranie systému Amfora vyžaduje pre beh ASGI server. Na tento účel som zvolil server uvicorn[10]. Keďže súčasti platformy DP^3 využívajú správcu procesov Supervisor, umiestnil som pod tohto správcu aj konfiguráciu pre spúšťanie ASGI servera API rozhrania.

Zdrojový kód API rozhrania vyžaduje jazyk Python verzie 3.9 alebo novší. Na inštaláciu závislostí sa využíva príkaz `pip install -r requirements.txt`. Spustenie API rozhrania so serverom uvicorn je možné príkazom: `uvicorn main:app`.

[2] Certbot: https://certbot.eff.org/
[3] OIDC: OpenID Connect
[4] Perun: https://aai.cesnet.cz/en/index
[5] OAuth2 Proxy: https://oauth2-proxy.github.io/oauth2-proxy/
[6] Supervisor: http://supervisord.org/
[7] Nasadenie DP^3: https://cesnet.github.io/dp3/install/#application-deployment
[8] Node.js: https://nodejs.org/en
[9] npm: https://www.npmjs.com/
[10] uvicorn: https://www.uvicorn.org/

Kapitola 7

Testovanie

Táto kapitola opisuje priebeh a výsledky testovania vyvinutých častí systému Amfora. Prvá časť kapitoly (sekcia 7.1) sa venuje testovaniu API rozhrania pomocou automatických testov. Ďalšie sekcie sa venujú testovaniu používateľského rozhrania – v sekcii 7.2 ide o testovanie na používateľoch systému, v sekcii 7.3 o testovanie responzívneho dizajnu webového používateľského rozhrania a v sekcii 7.4 o testovanie zabezpečeného prístupu pomocou protokolu OpenID Connect. Nakoniec popisuje sekcia 7.5 možné budúce rozšírenia systému Amfora.

7.1 Testovanie pomocou automatických testov

Keďže systém Amfora principiálne iba agreguje dáta z množstva iných systémov, jeho úplná izolácia počas testovania by bola náročná. Pri testovaní API rozhrania som preto zvolil kompromisné riešenie a rozdelil testy na dve časti. Vytvoril som jednotkové testy pre najkomplikovanejšie časti spracovania dát a testy endpointov API rozhrania pre všeobecné testovanie funkčnosti. Jednotkové testy sú izolované a využívajú iba testované časti implementácie bez pripájania na externé služby. Testy endpointov API rozhrania izolované nie sú, využívajú nastavenú konfiguráciu, pripájajú sa na všetky využívané systémy a slúžia na overenie správneho behu všetkých komponentov systému Amfora.

Zdrojové kódy vytvorených testov sú súčasťou zdrojových kódov API rozhrania v priečinku `tests`. Tento priečinok obsahuje krátky popis v súbore `README.md` a podpriečinky `api` a `unit` pre spomínané dva druhy testov. Ich implementácia využíva framework pytest[1].

Jednotkové testy využívajú tento framework priamo. Testovaná funkcionalita zahŕňa spracovanie dát zo systému DP[3]. Ide o transformáciu a v prípade niektorých atribútov aj overovanie validity dát ku entitám (IP adresám, skupinám IP adries a službám). Spracovanie dát z ostatných systémov je pomerne priamočiare, preto považujem v týchto prípadoch za postačujúce testovanie príslušných endpointov API rozhrania.

Testy endpointov API rozhrania využívajú triedu `TestClient` frameworku FastAPI. Testovací klient umožňuje vytváranie HTTP požiadaviek na server, čím je možné overiť systémovú funkčnosť. V skutočnosti žiaden HTTP server nebeží (nie je ho potrebné spúšťať ani manuálne), pretože smerovanie požiadaviek zaistí framework FastAPI interne. Testujú sa rôzne typy vstupov a testy sa zameriavajú aj na okrajové podmienky. Kvôli závislosti na využívaných systémoch nie je vo väčšine prípadov možné overovať obsah odpovede na požiadavku. Jej validitu však zabezpečuje dátový model opísaný v kapitole 4. Vo všetkých

[1] pytest: https://docs.pytest.org/en/8.2.x/

prípadoch prebieha overovanie návratového kódu. Spoločné časti implementácie sa nachádzajú v module `common.py` a na parametrizáciu testov sa používa parametrizačný dekorátor `pytest.mark.parametrize`.

7.2 Používateľské testovanie

Vytvorené používateľské rozhranie bolo po nasadení ukázané a odovzdané operátorom oddelenia CESNET SOC, teda primárnym používateľom systému. Zo získanej spätnej väzby vyplýva, že vytvorený systém zodpovedá ich požiadavkám a bude všeobecne užitočný pri reakcii na incidenty (incident response). Pri testovaní však objavili niekoľko problémov. Išlo napríklad o lexikálne radenie IP adries namiesto numerického. Tento problém vyžaduje úpravu dátového modelu platformy DP^3 a pridanie podpory pre radenie dát podľa atribútov, ktorá je naplánovaná v strednodobom horizonte. Taktiež bol identifikovaný problém v spracovaní klasifikácií IP adries zo systému ADiCT, kedy sa v systéme Amfora objavovali aj klasifikácie s dôveryhodnosťou 0 %. Tento problém už bol vyriešený. Posledným problémom bolo zaradenie IP adries do nesprávnej skupiny, čo bolo rovnako vyriešené.

7.3 Testovanie responzivity na mobilných zariadeniach

Hoci operátori oddelenia CESNET SOC budú k používateľskému rozhraniu systému Amfora pristupovať primárne zo zariadení s veľkou obrazovkou (pracovná stanica, notebook), z rozličných dôvodov môže byť potrebné zobrazenie rozhrania aj z mobilných zariadení. Navyše, prispôsobenie webových rozhraní pre mobilné telefóny je v dnešnej dobe považovaná za samozrejmosť. Z tohto dôvodu je potrebné otestovať responzivitu používateľského rozhrania systému Amfora.

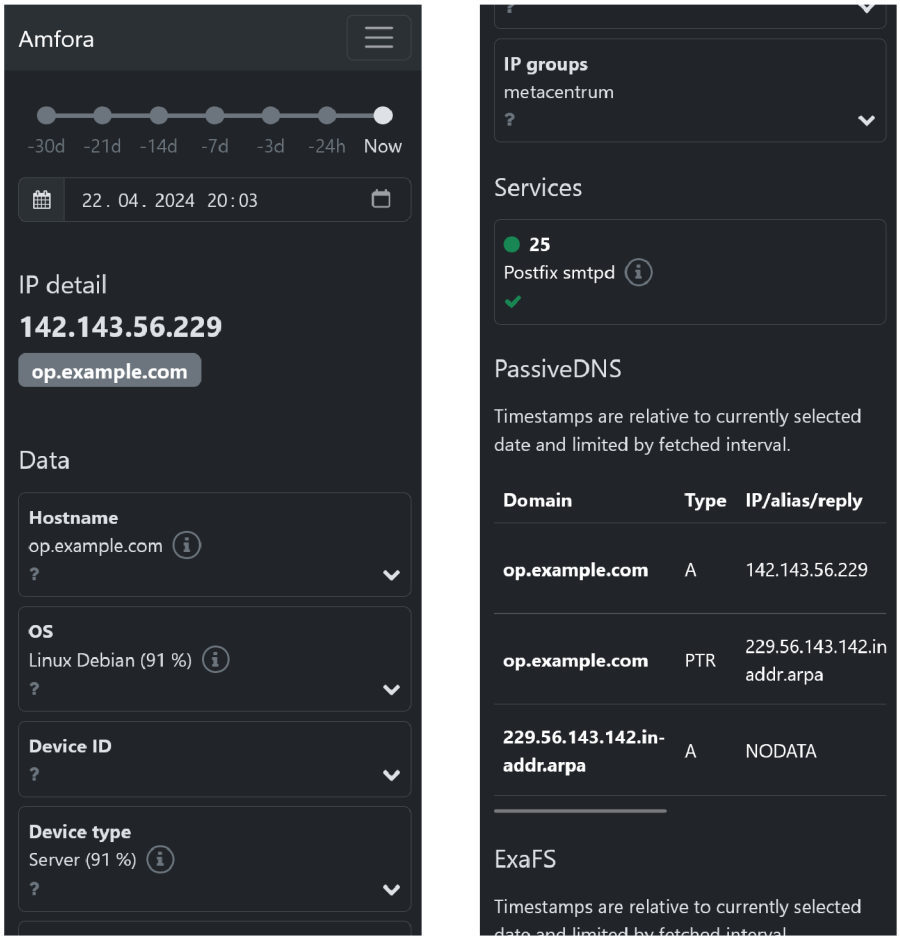
V rámci testovania responzivity bolo overené zobrazenie na mobilných telefónoch s operačným systémom Android a taktiež pomocou režimu *Responzívny dizajn* vo vývojových nástrojoch prehliadača Firefox.

Na mobilných zariadeniach s šírkou displeja pod 992 pixelov je navbar zložený. Rozloženie stránok sa prispôsobuje šírke zariadenia. V prehľadoch entít sa na mobilných telefónoch zobrazuje filtrovací formulár v jednom stĺpci. Tabuľka je horizontálne rolovateľná, keďže obsahuje príliš veľké množstvo dát. V detailoch entít sa atribúty z platformy DP^3 zobrazujú v zhustenom stave. Žiadna časť stránky nepreteká. Vlastnosti mobilného zobrazenia dokladá obrázok 7.1. Podobný stav je možné pozorovať aj na rozlíšení obrazoviek typickom pre tablety. Mobilná verzia používateľského rozhrania neskrýva žiadne informácie – zobrazené dáta sú identické pri zobrazení na pracovnej stanici. Z týchto dôvodov konštatujem, že používateľské rozhranie je responzívne.

7.4 Testovanie autentifikácie a autorizácie

Veľmi dôležitým testom je v prípade systému Amfora test autorizácie. Vzhľadom na citlivosť údajov je overenie správnej funkcie autorizačného mechanizmu priam kľúčové. Tento test sa viaže na nasadenú verziu systému.

Prístup do systému Amfora majú iba operátori oddelenia CESNET SOC a niekoľko ďalších ľudí v združení CESNET. Overenie správnej funkčnosti autorizácie spočíva v umožnení prístupu pre jedného z povolených účtov a neumožnení prístupu pre iný účet. Realizácia je pomerne priamočiara – keďže mám vytvorený účet v združení CESNET s prístupovými



Obr. 7.1: **Zobrazenie detailu IP adresy na mobilnom telefóne.** Zobrazuje identickú entitu ako obrázok 5.5. Horná časť stránky je na ľavom obrázku, nasledujúca časť na pravom obrázku. Prvky sú zúžené a v jednom stĺpci. Tabuľka s údajmi zo systému PassiveDNS je rolovateľná. IP adresy a doménové mená sú anonymizované.

právami pre systém Amfora a taktiež mám účet na Vysokom učení technickom, po autentifikácii prvým účtom by som mal byť autorizovaný, avšak po autentifikácii druhým účtom nie. S využitím dvoch samostatných privátnych okien prehliadača som tento stav overil. V prípade VUT účtu bol prístup odmietnutý.

Ďalším testom je pokus o získanie prístupu bez akejkoľvek autentifikácie. Realizácia spočívala vo vytváraní požiadaviek na existujúce stránky webového a API rozhrania. Očakávaný stav bolo automatické presmerovanie na službu Perun s výzvou na autentifikáciu. To vo všetkých prípadoch nastalo a bez autentifikácie nebolo možné pokračovať.

7.5 Možné budúce rozšírenia

Systém Amfora bol v rámci tejto bakalárskej práce vyvinutý a nasadený do produkčného používania pre operátorov oddelenia CESNET SOC. Už teraz však existujú plány na jeho možné ďalšie vylepšenie.

Prirodzeným plánom je podpora pre systém NetBox, ktorý bol z implementácie vynechaný pre jeho aktuálnu nepripravenosť. Po pripojení získa systém Amfora dôležitý zdroj dát, čo výrazne zvýši použiteľnosť. Vo fáze návrhu bola taktiež požiadavka pre zobrazovanie informácií o externých IP adresách a službách. Aj to bolo neskôr vynechané z dôvodu chýbajúcich dát. V budúcnosti sa môžu dátové modely jednotlivých systémov rozšíriť alebo môžu vzniknúť nové nástroje, ktoré budú považované za užitočné pri práci operátorov oddelenia CESNET SOC. V takom prípade je možná úprava alebo pridanie podpory aj do agregačného systému Amfora.

V strednodobom horizonte je plánované rozšírenie možností platformy DP^3 o zoradenie dát podľa atribútov či možnosť numericky radiť IP adresy, čo bude využité aj v systéme Amfora.

Medzi ďalšie možné rozšírenia patria notifikácie o nových detekovaných anomáliách. Vzhľadom na veľké množstvo entít môže byť sledovanie ich stavu náročné. Odosielanie emailových alebo iných notifikácií by uľahčilo dohľad. Niektoré využívané systémy už takýto mechanizmus implementujú, preto je potrebné zvážiť prípady, kedy by boli notifikácie prínosné. Táto funkcia však vyžaduje pripojenie systému NetBox.

Kapitola 8

Záver

Cieľom tejto práce bolo vytvorenie webového používateľského rozhrania pre systém Amfora. Keďže existujúci základ systému založený na platforme DP^3 nebol postačujúci, došlo pri práci k jeho rozšíreniu. Implementovaná bola frontendová a backendová (API) časť systému Amfora a v návrhovej fáze aj demo verzia používateľského rozhrania. Frontend bol vytvorený s využitím frameworku Vue a API rozhranie s využitím frameworku FastAPI.

Systém Amfora agreguje dáta o IP adresách, službách a zraniteľnostiach zo 7 existujúcich interných systémov (SNER, ADiCT, Mentat, PassiveDNS a ďalšie), 2 externých systémov (Shodan, ShadowServer), odkazuje sa na systém OTRS a umožňuje ďalšiu rozšíriteľnosť. Medzi entitami umožňuje vyhľadávať a filtrovať podľa rôznych kritérií. Zobrazuje stránkovaný súhrn všetkých entít daného typu (IP adresy, skupiny IP adries, služby) vrátane ich validácie. Používateľské rozhranie obsahuje aj podrobný detail IP adries a služieb s dátami o ich aktuálnom a historickom správaní.

Implementácii predchádzala analýza požiadaviek operátorov oddelenia CESNET SOC, ktorí sú primárnymi používateľmi systému Amfora. Na jej základe bol vytvorený návrh systému, mockup používateľského rozhrania a zvolené technológie na vývoj.

Súčasťou implementácie malo byť pripojenie na systém NetBox, ktorý bude *source of truth* pre sieť CESNET. Prepojenie by umožňovalo detekciu anomálií porovnávaním staticky definovaného stavu oproti reálnemu. Z dôvodu nepripravenosti systému NetBox k prepojeniu v práci nedošlo. Implementácia obsahuje prípravu na toto prepojenie a zatiaľ ho čiastočne nahrádza vlastnými validačnými pravidlami. Pripojenie systému NetBox zostáva najvýznamnejším plánom na budúce rozšírenie.

Systém Amfora bol produkčne nasadený a odovzdaný do užívania operátorom oddelenia CESNET SOC, ktorým už pomáha pri riešení bezpečnostných incidentov. Nasadená verzia bola zabezpečená autentifikačným protokolom OpenID Connect s využitím služby Perun. Implementácia systému bola otestovaná automatickými testami, používateľsky na operátoroch oddelenia SOC a prebehli testy responzivity používateľského rozhrania. Na nasadenej verzii boli vykonané testy autentifikácie a autorizácie.

Literatúra

[1] *Django* [online]. [cit. 2024-04-13]. Dostupné z: https://www.djangoproject.com/.

[2] *FastAPI* [online]. [cit. 2024-04-13]. Dostupné z: https://fastapi.tiangolo.com/.

[3] *Flask Documentation (3.0.x)* [online]. [cit. 2024-04-13]. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/.

[4] *Pydantic* [online]. [cit. 2024-04-13]. Dostupné z: https://docs.pydantic.dev/2.7/.

[5] *Mentat* [online]. CESNET, september 2018 [cit. 2024-04-07]. Dostupné z: https://mentat.cesnet.cz/cs/index.

[6] *Systém Mentat: SIEM sítě CESNET2* [online]. CESNET, august 2022 [cit. 2024-04-07]. Dostupné z: https://csirt.cesnet.cz/cs/services/mentat.

[7] Abdelrahman, M. M., Zhan, S. a Chong, A. A Three-Tier Architecture Visual-Programming Platform for Building-Lifecycle Data Management. *SimAUD*. Máj 2020. DOI: 10.13140/RG.2.2.26013.74724. Dostupné z: https://doi.org/10.13140/RG.2.2.26013.74724.

[8] Bartoš, V. NERD: Network Entity Reputation Database. In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*. Association for Computing Machinery, 2019. ARES '19. DOI: 10.1145/3339252.3340512. ISBN 9781450371643. Dostupné z: https://doi.org/10.1145/3339252.3340512.

[9] Bodó, R. *SNER - Slow Network Recon Service* [online]. CESNET [cit. 2024-04-05]. Dostupné z: https://sner.flab.cesnet.cz/.

[10] Bodó, R. *Sner – slow network recon* [online]. 2023 [cit. 2024-04-05]. Dostupné z: https://github.com/bodik/sner4/tree/60f1f13bb0bf88db518c59d6f45db71e32ba6098.

[11] Caletka, O. *Passive DNS*. 2014 [cit. 2024-04-07]. Prezentované na konferencii Internet a Technologie 14, Praha. Dostupné z: https://www.nic.cz/public_media/IT14/prezentace/Ondrej_Caletka.pdf.

[12] Gavigan, D. *The History of Angular* [online]. Apríl 2018. Dostupné z: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7.

[13] Greif, S., Burel, E. et al. *State of JavaScript 2022: Front-end frameworks* [online]. [cit. 2024-04-14]. Dostupné z: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/.

[14] JetBrains. *Python Programming - The State of Developer Ecosystem in 2023 Infographic* [online]. [cit. 2024-04-13]. Dostupné z: https://www.jetbrains.com/lp/devecosystem-2023/python/.

[15] Kishore, P. a Mahendra, B. M. Evolution of Client-Side Rendering over Server-Side Rendering. *Recent Trends in Information Technology and its Application.* 2020, zv. 3, č. 2, s. 4–13. ISSN 2584-0991.

[16] Košňar, T. *Flow-Based Traffic Analysis System – Architecture Overview.* Technical report 15/2004. CESNET, december 2004. Dostupné z: https://archiv.cesnet.cz/doc/techzpravy/2004/ftas-arch/.

[17] Kropáčová, A. *Pracovní náplň oddělení SOC (Security Operation Centre).* CESNET, 2024 [cit. 2024-04-05]. Interný dokument CESNET.

[18] Kácha, P. *OTRS: Tool for Security Incident Reports Management.* Technical report 12/2007. CESNET, november 2007. Dostupné z: https://archiv.cesnet.cz/doc/techzpravy/2007/otrs/.

[19] Kácha, P. IDEA: Classification of Security Events, their Participants and Detection Probes. *WSEAS Transactions on Computers.* 2015, zv. 14, s. 213–223. ISSN 1109-2750.

[20] Kácha, P., Kostěnec, M. a Kropáčová, A. Warden 3: Security event exchange redesign. In: *Proceedings of the 19th International Conference on Computers: Recent Advances in Computer Science.* 2015.

[21] Man, J. *User Interface for DDoS Mitigation Configuration.* Brno, CZ, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/24819/.

[22] Matherly, J. et al. *What is Shodan?* [online]. [cit. 2024-04-09]. Dostupné z: https://help.shodan.io/the-basics/what-is-shodan.

[23] Saks, E. *JavaScript Frameworks: Angular vs React vs Vue.* Helsinki, 2019. Bachelor's Thesis. Haaga-Helia University of Applied Sciences. Dostupné z: https://urn.fi/URN:NBN:fi:amk-2019110720797.

[24] Sedláček, O. *Fúze dat pro klasifikaci síťových zařízení.* Brno, CZ, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/25057/.

[25] Sedláček, O., Benko, D. a Bartoš, V. *Dynamic Profile Processing Platform (DP^3)* [online]. 2024 [cit. 2024-04-10]. Dostupné z: https://cesnet.github.io/dp3/.

[26] Shadowserver Foundation. *Shadowserver* [online]. [cit. 2024-04-09]. Dostupné z: https://www.shadowserver.org/.

[27] Stretch, J. et al. *NetBox* [online]. 2024 [cit. 2024-04-09]. Dostupné z: https://github.com/netbox-community/netbox/tree/b7668fbfc3a81abf2c6a8ace98047647fd9244c9.

[28] Svoboda, L. Věda bez hranic. *AB / Akademický bulletin*. Středisko společných činností AV ČR. Júl 2021, 6-7, s. 14–15. Dostupné z: https://www.avcr.cz/export/sites/avcr.cz/.content/galerie-souboru/AB/2021/AB-2021-0607.pdf.

[29] Syrjäkoski, J. a Yrjänäinen, J. *Lovelace messaging microservice*. Oulu, 2023. Bachelor's Thesis. University of Oulu, Faculty of Information Technology and Electrical Engineering. Dostupné z: https://urn.fi/URN:NBN:fi:oulu-202305171823.

[30] Tomáš, M. *Rozšíření reputační databáze o informace z Passive DNS*. Praha, 2018. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií. Dostupné z: http://hdl.handle.net/10467/76827.

[31] Vraný, J. a Man, J. *ExaFS* [online]. 2024 [cit. 2024-04-07]. Dostupné z: https://github.com/CESNET/exafs/tree/dd2f5927a60d7fa2212269c816cf096433f5936d.

[32] Weimer, F. Passive DNS Replication. In: *FIRST conference on computer security incident*. 2005, sv. 98, s. 1–14.

Príloha A

Obsah priloženého pamäťového média

Priložené pamäťové médium obsahuje nasledujúce adresáre:

- **`api`**: zdrojové súbory API rozhrania
- **`config`**: ukážka konfigurácie API rozhrania v súbore `api.yml.example` a konfigurácia platformy DP^3 v podadresári `dp3`
- **`defined_state_source`**: zdrojové súbory pre budúce prepojenie so systémom NetBox
- **`frontend`**: zdrojové súbory frontendu používateľského rozhrania
- **`frontend_demo`**: zdrojové súbory demo verzie používateľského rozhrania
- **`thesis`**: zdrojové súbory tejto bakalárskej práce a jej finálna podoba vo formáte PDF