VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

OPTIMALIZACE REPUTAČNÍHO SYSTÉMU SÍŤOVÝCH ENTIT

OPTIMIZATION OF NETWORK ENTITY REPUTATION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAKUB MAGDA
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

155230

Ústav: Ústav počítačových systémů (UPSY)
Student: **Magda Jakub**
Program: Informační technologie
Název: **Optimalizace reputačního systému síťových entit**
Kategorie: Počítačové sítě
Akademický rok: 2023/24

Zadání:

1. Seznamte se se systémem Network Entity Reputation Database (NERD) provozovaným sdružením CESNET.
2. Nastudujte platformu Dynamic Profile Processing Platform (DP³) vyvíjenou tamtéž.
3. Navrhněte způsob, jak reimplementovat funkcionalitu systému NERD v podobě aplikace nad platformou DP³. Identifikujte případné nedostatky v současné verzi platformy DP³ a zajistěte implementaci nezbytných úprav.
4. Navrženou aplikaci implementujte, nakonfigurujte a ověřte její funkčnost.
5. Diskutujte rozdíly mezi původní a nově vytvořenou verzí systému NERD.
6. Zhodnoťte výhody nového řešení a navrhněte možná rozšíření do budoucna.

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 problematikou návrhu a reimplementácie reputačného systému sieťových entít NERD (Network Entity Reputation Database) na platformu DP^3 (Dynamic Profile Processing Platform). Práca predstavuje návrh nového systému NERD a jeho dátového modelu. Nefungujúce časti systému odstraňuje, vyberá časti, ktoré budú potrebné aj na novej platforme a navrhuje spôsob ich adaptácie na DP^3. Návrh dátového modelu vychádza z pôvodnej verzie, ktorú vylepšuje a prevádza na platformu DP^3. Výsledkom práce je reputačný systém fungujúci na všeobecnejšej platforme. Vďaka tomu je otvorený budúcim rozšíreniam.

Abstract

This thesis addresses the problem of designing and reimplementing a Network Entity Reputation Database (NERD) to the Dynamic Profile Processing Platform (DP^3). This thesis presents the design of the new NERD system and its data model. The non-functioning parts of the system are removed, and those that will also be needed on the new platform are selected. A proposal is then made for adapting them to DP^3. The data model has been designed based on the original version, which has been improved and converted to the DP^3 platform. The outcome of this endeavour is a reputation system that is operational on a more general platform. This facilitates future extensions.

Kľúčové slová

Bezpečnosť počítačových sietí, reputačná databáza, platforma na sledovanie hrozieb, CTI, NERD, DP^3

Keywords

Network security, reputation database, threat intelligence platform, CTI, NERD, DP^3

Citácia

MAGDA, Jakub. *Optimalizace reputačního systému síťových entit.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Žádník, Ph.D.

Optimalizace reputačního systému síťových entit

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Martina Žádnika, Ph.D. Ďalšie informácie mi poskytol pán Ing. Václav Bartoš zo združenia CESNET. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

. . . . . .
Jakub Magda
7. mája 2024

Poďakovanie

Chcel by som sa poďakovať môjmu vedúcemu práce Ing. Martinovi Žádnikovi, Ph.D. a taktiež konzultantovi Ing. Václavovi Bartošovi, ktorý mi pomáhal a poskytol veľa informácií. Ďakujem aj všetkým kolegom zo združenia CESNET, ktorý mi akýmkoľvek spôsobom poradili a pomohli.

Obsah

1 Úvod **4**

2 Zhrnutie doterajšieho stavu **5**
2.1 Network Entity Reputation Database (NERD) 5
2.1.1 Architektúra NERD 5
2.1.2 Dátový model 7
2.1.3 Zdroje dát 7
2.1.4 Rozdeľovanie úloh 8
2.2 Dynamic Profile Processing Platform (DP^3) 10
2.2.1 Architektúra 11
2.2.2 Dátový model 13
2.2.3 Správa histórie 14
2.2.4 API 17
2.2.5 Konfigurácia 18
2.2.6 Primárne moduly 19
2.2.7 Sekundárne moduly 19

3 Návrh riešenia **21**
3.1 Porovnanie architektúry NERD a DP^3 21
3.2 Rozbor systému NERD 22
3.3 Dátový model 22
3.4 Primárne moduly 23
3.4.1 Modul blacklists 23
3.4.2 Modul Warden receiver 24
3.4.3 Modul MISP receiver 25
3.4.4 Modul OTX receiver 26
3.4.5 Modul DShield 26
3.5 Sekundárne moduly 27
3.5.1 Modul DNS 27
3.5.2 Modul DNS blacklist resolver 28
3.5.3 Modul Passive DNS resolver 28
3.5.4 Modul Redis blacklist 28
3.5.5 Modul Geolocation 29
3.5.6 Modul Hostname class 29
3.5.7 Modul Shodan 29
3.5.8 Moduly Reserved IP a Tags 30
3.5.9 Zvyšné sekundárne moduly NERD 30
3.5.10 Potrebné rozšírenie DP^3 31

4 Implementácia **32**
4.1 Primárne moduly 32
4.1.1 Konfigurácia 32
4.1.2 Posielanie data-pointov 33
4.1.3 Moduly 33
4.2 Sekundárne moduly 34

5 Testovanie **37**

6 Záver **41**

Literatúra **42**

A Obsah priloženého pamäťového média **43**

Zoznam obrázkov

2.1 Architektúra systému NERD 5
2.2 Architektúra frontov pracovných procesov v systéme NERD 9
2.3 Zjednodušená schéma fungovania manažéra aktualizácií. 10
2.4 Architektúra DP³ 12
2.5 Dátový tok v DP³ 12
2.6 Intervaly platnosti pre jeden data-point 15
2.7 Interpolácia spoľahlivosti pre jeden data-point. 15
2.8 Vyhľadanie aktuálnej hodnoty s prekrývajúcimi sa platnosťami data-pointov 15
2.9 Správanie atribútu s parametrom *multi_value* 16
2.10 Dva po sebe idúce data-pointy s rovnakou hodnotou sa zlúčia. 16
2.11 Spoľahlivosť zlúčených data-pointov sa spriemeruje. 17
2.12 Každá hodnota viachodnotového atribútu sa zlúči samostatne. 17

3.1 Aktuálne podporované entity a ich vzťahy. Obrázok bol prevzatý z [1]. . . . 23

5.1 Graf využitia CPU novej verzii NERD. V legende id-1 až 8 označuje 8 jadier procesora. 38
5.2 Graf využitia CPU pôvodnej verzii NERD. Server má 8 jadier, preto má osa y 800%. 38
5.3 Graf využitia pamäte v novej verzii NERD. 39
5.4 Graf využitia pamäte v pôvodnej verzii NERD. 39
5.5 Počet spracovaných úloh podľa zdroja v novej verzii NERD. 39
5.6 Počet spracovaných úloh podľa zdroja v pôvodnej verzii NERD. 40

Kapitola 1

Úvod

Internet v dnešnej dobe používa celý svet. Denne ho používajú miliardy ľudí. S istotou je možné povedať, že súčasná civilizácia je od internetu závislá, lebo všetky dôležité služby, vrátane tých najdôležitejších, fungujú vďaka nemu. Rovnako ako každý deň pribúda počet pripojených zariadení, pribúda aj počet útokov na ne. Potreba bezpečnosti na internetovej sieti bude naďalej rásť. Je potrebné zistiť o útočníkoch čo najviac informácií a vďaka tomu im v tom zabrániť alebo aspoň urobiť zložitejším ďalšie útočenie.

Systém NERD (Network Entity Reputation Database), ktorý je predmetom tejto práce, získava a zhromažďuje informácie o útočníkoch. Zhromažďuje ich IP adresy, webové adresy a veľa iných informácií. Zoznam - databázu týchto škodlivých sieťových entít vytvára vďaka informáciám z viacerých zdrojov. Tieto informácie potom prehľadne poskytuje verejnosti. Službu využíva viacero firiem a správcov sietí po celom svete na zvýšenie ich bezpečnosti. NERD je vyvíjaný združením CESNET.

Cieľom práce je optimalizovať systém NERD tým, že bude prevedený na platformu Dynamic Profile Processing Platform (DP^3). V súčasnej dobe NERD používa vlastnú platformu na prácu s dátami. Združenie CESNET prevádzkuje viacero iných systémov na platforme DP^3. Zjednotiť systémy na jednu platformu má veľa výhod. Uľahčí to vývoj jednotlivých systémov, keďže nebude potrebné vyvíjať zároveň aj viacero platforiem. Platforma DP^3 je rovnako vyvíjaná združením CESNET a predstavuje lepšiu alternatívu pre budúce smerovanie systému NERD ako jeho súčasná platforma. Jej návrh vychádza práve z architektúry systému NERD, ale je navrhnutá kvalitnejšie, poskytuje pre užívateľa lepšie rozhranie a vyššiu modularitu. Ďalej poskytuje možnosť konfigurácie entít a atribútov čo znamená aj lepšiu prácu s dátami.

Aby sa podarilo ciele práce naplniť bolo potrebné urobiť viaceré kroky. Kapitola 2 sa venuje rozboru súčasných verzií systému NERD a platformy DP^3. Kapitola 3 opisuje návrh ako presunúť jednotlivé komponenty systému na novú platformu a návrhu dátového modelu, ktorý bude nová verzia systému používať. Kapitola 4 sa venuje opisu implementácie navrhnutých zmien. Kapitola 5 opisuje testovanie systému a jeho skúšobné nasadenie.

Kapitola 2

Zhrnutie doterajšieho stavu

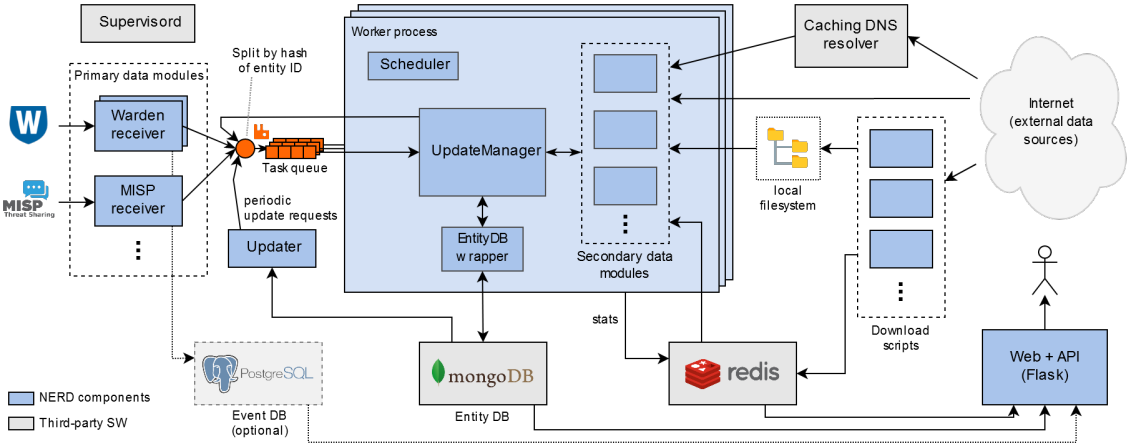
Táto kapitola zhrňuje doterajší stav problematiky, ktorý bolo potrebné naštudovať pre vypracovanie práce. Prvá sekcia 2.1 je venovaná systému NERD, jeho architektúre a jednotlivým súčastiam. Druhá sekcia 2.2 je o platforme DP³. Rovnako opisuje jej architektúru a prácu s dátami.

2.1 Network Entity Reputation Database (NERD)

Network Entity Reputation Database, skrátene NERD, je softvér a služba, ktorá zhromažďuje ukladá, agreguje a následne prezentuje rôzne informácie o známych škodlivých sieťových entitách. Väčšinu entít v systéme tvoria IP adresy. NERD je vyvíjaný a prevádzkovaný združením CESNET. Všetky informácie a obrázky sú prebrané z [1] a z [2].

2.1.1 Architektúra NERD

Systém NERD sa skladá z viacerých častí. Jeho architektúra je zobrazená na obrázku 2.1. Modrou farbou sú zobrazené časti architektúry, ktoré sú vyvíjané združením CESNET a sivou farbou softvér tretích strán.



Obr. 2.1: Architektúra systému NERD

Systém NERD by sa dal logicky rozdeliť na tieto 4 časti.

Back end

Predstavuje najväčšiu časť systému a zároveň najpodstatnejšiu z hľadiska tejto práce. Tvorí ho viacero častí a komponentov, ktoré môžu paralelne bežať vo viacerých inštanciách.

Jadrom systému sú pracovné procesy (worker processes), ktoré zabezpečujú spracovávanie všetkých záznamov a úloh v systéme. Každý takýto proces vyberá úlohy z hlavného frontu úloh (main task queue) a vykoná požadovanú aktualizáciu záznamu v databáze, prípadne volá sekundárne zásuvné moduly na úpravu dát alebo získanie ďalších údajov.

Ďalšou časťou sú moduly (primárne moduly). Úloha modulov je získavanie dát z primárnych dátových zdrojov a vkladanie získaných dát do hlavného frontu úloh (main task queue) na ďalšie spracovanie. Viac o primárnych dátových zdrojoch sa píše v podsekcii 2.1.3.

Sekundárne moduly sú modulárnou súčasťou jadra systému. Majú zaregistrované spätné volania na rôzne udalosti. Ak nastane zaregistrovaná udalosť, jadro spustí funkciu jeho obsluhy, ktorú definuje modul. V NERDe sú 2 základné typy udalostí. Prvá je udalosť, ktorú spôsobilo pridanie/zmenenie nejakej entity. Druhou je udalosť, ktorú poslal primárny modul. Táto udalosť sama o sebe nevykoná žiadne zmeny, ale spôsobí spustenie na ňu zaregistrovaných spätných volaní.

Aktualizátor (Updater) je modul, ktorého úlohou je raz za deň zabezpečiť aktualizáciu každého záznamu v databáze. Každý záznam je aktualizovaný v iný čas, takže sú aktualizácie rovnomerne rozložené po celý deň aby nevzniklo nárazové zaťaženie systému.

Hlavný front úloh pozostáva z viacerých frontov a slúži na distribúciu úloh medzi pracovné procesy. Viac o rozdeľovaní úloh sa píše v podsekcii 2.1.4.

Redis je *key-value* databáza. Ukladajú sa v nej rôzne informácie pre sekundárne moduly, napríklad čierne listiny. Rovnako uchováva aj rôzne štatistiky o spracovaní dát v systéme.

Skripty na sťahovanie údajov slúžia na pravidelné sťahovanie dát zo sekundárnych dátových zdrojov a ich ukladanie do Redis databázy alebo do lokálnych súborov. Tieto dáta potom následne môžu čítať sekundárne moduly a pridávať ich do databázy.

Front end

Aby komunita mohla k dátam jednoducho pristupovať NERD v súčasnosti poskytuje webový server napísaný v jazyku Flask a API na prístup ku všetkým údajom. Existuje už aj novšia a modernejšia verzia webu, ktorá vznikla ako diplomová práca Ing. Kristíny Oltmanovej [5]. Tá však nie je aktuálne nasadená.

Databázy

Databáza entít je hlavnou databázou a ukladá všetky záznamy o sieťových entitách v NERDe. Používa nerelačnú databázu MongoDB.

Databáza udalostí je voliteľná databáza na ukladanie kópií každého upozornenia (správy IDEA) prijatého z Wardenu. Hlavná inštancia NERD v súčasnosti namiesto toho používa databázu projektu Mentat. Databáza je relačná a používa PostgreSQL databázový systém.

Konfigurácia

Na správne fungovanie a jednoduché nastavovanie všetkých častí systému slúžia konfiguračné súbory. Obsahujú základnú konfiguráciu systému, ale aj zoznamy a prístupové údaje k dátovým zdrojom alebo pravidlá na úpravu a dát a extrakciu informácií.

2.1.2 Dátový model

Ako už bolo spomenuté v časti o architektúre systému 2.1.1, NERD používa na ukladanie všetkých sieťových entít databázu MongoDB. Pre každý typ entity sa v databáze nachádza kolekcia, ktorá obsahuje záznam s informáciami pre každú entitu. Zoznam typov entít:

- IP adresa - Najpočetnejšie zastúpenie, tvorí veľkú väčšinu entít. Podporuje iba verziu 4, pretože NERD v súčasnosti nepodporuje verziu 6,
- BGP prefix,
- ASN - Autonomous System Number (Číslo autonomného systému),
- IP blok - alokovaný blok IP adries,
- organizácia.

Záznamy entít sú uložené pomocou dátovej štruktúry typu slovník. Kľúče slovníka sa nazývajú atribúty. Každý záznam obsahuje unikátne ID, čas vytvorenia záznamu a čas poslednej aktualizácie záznamu. Pre každý typ entity existujú rôzne atribúty, podľa informácie, ktorú obsahujú. Nasleduje zoznam všeobecných formátov atribútov, ktoré NERD používa:

- jednoduché hodnoty,
- hodnoty s mierou spoľahlivosti,
- mapovanie - zoznam objektov,
- štítok (tag).

2.1.3 Zdroje dát

NERD prijíma dáta z rôznych zdrojov. Niektoré sú verejné k iné sú priamo od CESNETu a verejnosť sa k nim môže dostať iba cez NERD. Nasledujúca podsekcia bola prebratá z [5].

Warden

Warden je systém na zdieľanie upozornení, ktorý je prevádzkovaný spoločnosťou CESNET. Jeho úlohou je zbierať hlásenia z niekoľkých systémov, napríklad honeypots, analyzátorov toku dát a iných detektorov nebezpečných sieťových prevádzok, ktoré sú monitorované prevažne (ale nie výlučne) v českých akademických sieťach. Denne je zdieľaných približne milión upozornení z niekoľkých desiatok detekčných systémov. Ku kompletným dátam upozornení zo zdroja Warden má prístup len uzavretá komunita ľudí. Verejnosti prístupné sú iba základné metadáta, ako napríklad počty upozornení podľa rôznych kategórií a IP adries. Tieto informácie sú verejnosti dostupné práve cez systém NERD.

Čierne listiny

NERD využíva takmer 50 verejne dostupných čiernych listín, ktoré obsahujú plaintext zoznamy nebezpečných IP adries. Tieto listiny sú získavané z 20 rôznych zdrojov. Pre každú IP adresu, ktorá sa vyskytne na listine je vytvorený záznam v systéme NERD.

DShield

DShield je firewall korelačný systém založený na spolupráci v správe inštitútu SANS[1]. Rozsiahla komunita prispievateľov tu zdieľa informácie o neočakávaných alebo neoprávnených prichádzajúcich pripojeniach. Verejnosti sú dostupné aj štatistiky o individuálnych portoch a IP adresách. NERD využíva denný výpis všetkých IP adries nahlásených pomocou systému DShield a do svojej databázy pridá také IP adresy, ktoré boli nahlásené minimálne desaťkrát, a to aspoň tromi rôznymi zdrojmi.

AlienVault Open Threat Exchange (OTX)

OTX je open source komunitný portál, na ktorom počítačoví bezpečnostní výskumníci a profesionáli zdieľajú denne milióny indikácií hrozieb. NERD odoberá niekoľko účtov a pulzov, ktoré boli vyhodnotené ako spoľahlivé a užitočné. Konkrétne účet AlienVault a niekoľko honeypot zdrojov je použitých k dopĺňaniu informácií do systému NERD, ktorý prehľadá v nich prítomné IP adresy a vytvorí nové alebo modifikuje už existujúce záznamy vo svojej databáze.

Malware Information Sharing Platform (MISP)

NERD je pripojený na rozsiahlu inštanciu systému MISP, ktorá je prevádzkovaná centrom hlásení počítačových incidentov v Luxembursku, CIRCL (The Computer Incident Response Center Luxembourg). NERD z tohto systému získava všetky IP adresy s označením tlp:white, teda verejne dostupné informácie.

Sekundárne zdroje

Každá IP adresa získaná z vyššie uvedených zdrojov je navyše obohatená o nasledujúce informácie nadobudnuté zo sekundárnych zdrojov:

- doménové meno – získané reverzným DNS dotazom,
- ASN – autonómne systémové číslo,
- informácie z whois[2] (zdroja pre Domain Name System (DNS) informácie),
- geografická lokalizácia – získaná z databázy GeoLite2 od MaxMind[3],
- prítomnosť na sekundárnych čiernych listinách,
- pasívny DNS systém spoločnosti CESNET.

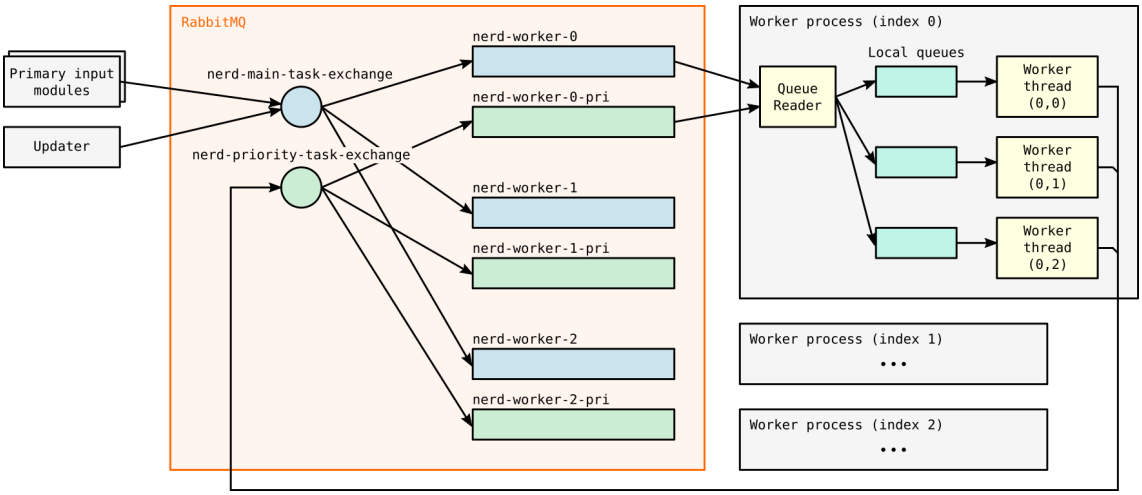
2.1.4 Rozdeľovanie úloh

Ako sa píše v podsekcii 2.1.1 jadro systému vyberá úlohy z frontov a následne ich spracováva a zmeny ukladá do databáze. V tejto podsekcii sa preberá podrobnejšie proces výberu úloh z frontov a ich následné spracovanie.

[1] https://www.sans.edu
[2] https://who.is
[3] https://www.maxmind.com/en/geoip-databases

Fronty

Každý pracovný proces v systéme má dva RabbitMQ[4] fronty. Jeden je normálny a druhý prioritný. Väčšina komponentov posiela úlohy do normálneho frontu vrátane primárnych modulov. Iba jedna metóda v pracovných procesoch posiela úlohy do prioritného frontu. Normálny front je avšak obmedzený dĺžkou koľko správ sa do neho zmestí. Potom musia komponenty systému, ktoré sa snažia poslať správu čakať, kým sa miesto uvoľní, čo však môže byť teoreticky do nekonečna. Prioritný front je neobmedzený, takže tu tento problém nenastane. Tieto neblokujúce prioritné fronty sú avšak potrebné aby nevzniklo vyhladovanie. Metóda, ktorá používa prioritný front je volaná počas spracovania inej úlohy. Ak by používala normálny front a ten by bol plný, nemohla by začať spracovávať ďalšiu úlohu ani dokončiť spracovávanie aktuálnej a tak by sa celý pracovný proces zasekol. Architektúra frontov je zobrazená na obrázku 2.2.



Obr. 2.2: Architektúra frontov pracovných procesov v systéme NERD

Ako je vidno na obrázku 2.2 jednotlivé úlohy sú rozdeľované medzi jednotlivé pracovné procesy. Toto rozdeľovanie sa uskutočňuje pomocou hašovacej funkcie MD5. To zabezpečuje dve veľmi podstatné veci. Prvá je rovnomernosť záťaže na jednotlivé pracovné procesy a druhá je, že jednu entitu (IP adresu) bude vždy obsluhovať rovnaký pracovný proces. To znamená, že odpadá potreba synchronizácie pracovných procesov alebo výlučného prístupu do databázy.

Pracovné procesy pracujú vo viacerých vláknach, pričom každé vlákno má svoj lokálny front. Preto je potrebné rozdelovať úlohy aj medzi tieto vlákna ako to ukazuje pravá časť obrázka 2.2. Medzi vlákna sa potom jednotlivé úlohy rozdeľujú podobne ako to je pri rozdeľovaní medzi jednotlivé procesy.

Formát úlohy

Každá úloha, ktorá prichádza do frontov sa skladá z trojice, ktorá špecifikuje záznam entity, s ktorým sa má pracovať a zoznam operácií, ktoré sa majú vykonať:

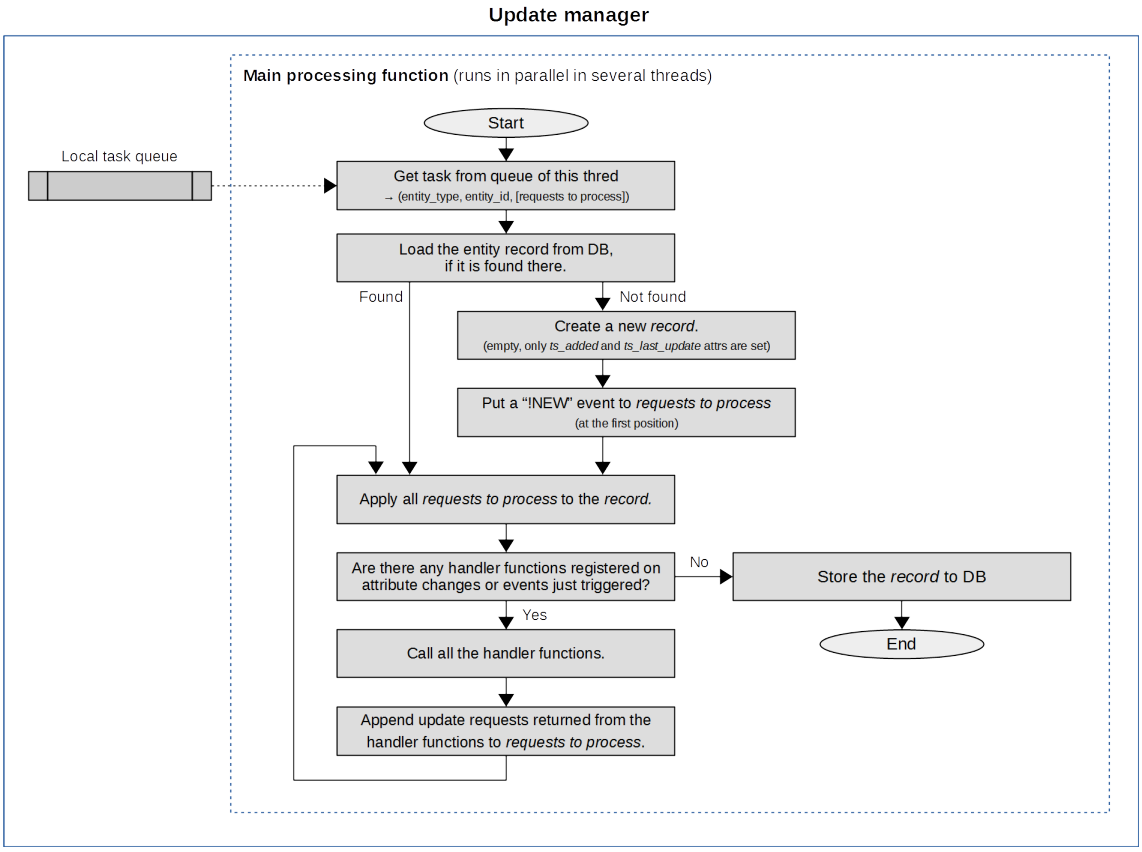
- *etype*: typ entity, napríklad IP adresa,
- *eid*: identifikátor entity,

[4] https://www.rabbitmq.com

- *requested_changes* (požadované zmeny): zoznam n-tíc, v ktorom sú špecifikované operácie, ktoré sa majú vykonať. Každá n-tica má tento formát: *(operácia, kľúč, [hodnota/parametre, ...])*.

Manažér aktualizácií

Manažér aktualizácií (Update manager) je centrálnou komponentou NERD, ktorá riadi tok dát medzi jednotlivými modulmi. Spracúva požiadavky na aktualizáciu vydané modulmi, načítava záznamy z databázy, aplikuje požadované zmeny a prípadne volá obslužné funkcie zaregistrované na tieto zmeny. Základná funkčnosť komponenty je zobrazená na obrázku 2.3.



Obr. 2.3: Zjednodušená schéma fungovania manažéra aktualizácií.

2.2 Dynamic Profile Processing Platform (DP³)

DP³ je platforma, ktorá pomáha udržiavať databázu informácií o rôznych entitách, ktoré sa menia v čase. Jadrom platformy je databáza a hlavnou úlohou DP³ je udržiavať túto databázu celistvú a aktuálnu. Okrem toho DP³ nerobí veľa operácií ako taká, ale je otvorená pre moduly, ktoré pracujú nad dátami užívateľa podľa jeho potrieb a predstáv.

DP³ vychádza z architektúry NERD, ale je v mnohých ohľadoch kvalitnejšie navrhnutá a pridáva viacero funkcií, ktoré poskytujú lepšiu prácu s dátami. Súčasná nová verzia DP³ je vyvinutá zo staršej. Obe boli vytvorené združením CESNET primárne pre Asset Discovery

Classification and Tagging (ADiCT) projekt, ale v súčasnosti sa nová verzia používa vo viacerých systémoch. ADiCT sa zameriava na klasifikáciu sieťových zariadení, ale platforma je všeobecná a môže byť použitá na rôzne typy dát. Všetky informácie a obrázky v tejto sekcii sú prebrané z projektovej dokumentácie DP³ [6].

2.2.1 Architektúra

Táto podsekcia hovorí o architektúre DP³. Návrh architektúry vychádza zo staršej verzie platformy a rovnako zo systému NERD. Platforma je všeobecná a pracuje s rôznymi dátovými typmi.

Data-point

Základnou jednotkou dát, ktoré platforma používa sa nazýva data-point. Výpis 2.1 ukazuje príklad takéhoto data-pointu.

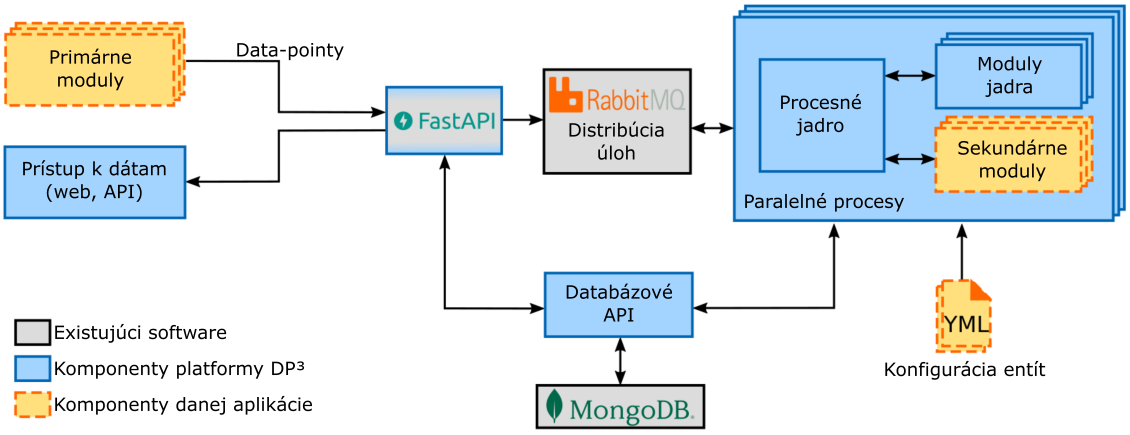
```
{
  "type": "ip",
  "id": "192.168.0.1",
  "attr": "blacklist",
  "v": "spamhaus_edrop",
  "t1": "2024-02-05T13:00:00",
  "t2": "2024-02-05T13:15:00",
  "src": "blacklists_module"
}
```

Výpis 2.1: Ukážka data-pointu s typom entity IP adresa.

Data-point obsahuje informácie v dátovom type JSON. Obsahuje typ entity, podľa konfigurácie. Viac o konfigurácii v podsekcii 2.2.5. Identifikátor data-pointu. Názov atribútu a jeho hodnotu. Ďalej obsahuje čas platnosti, *t1* a *t2* určujú časový interval, v ktorom je hodnota data-pointu platná. Hodnota *t2* nemusí byť zadaná, nie je povinná, tak sa vyjadrí platnosť hodnoty v jednom časovom bode a dĺžka platnosti hodnoty sa už odvíja od konfigurácie konkrétneho atribútu. Poslednou časťou je zdroj z ktorého hodnota pochádza, aby bolo možné jasne určiť odkiaľ aké dáta prichádzajú.

Architektúra platformy

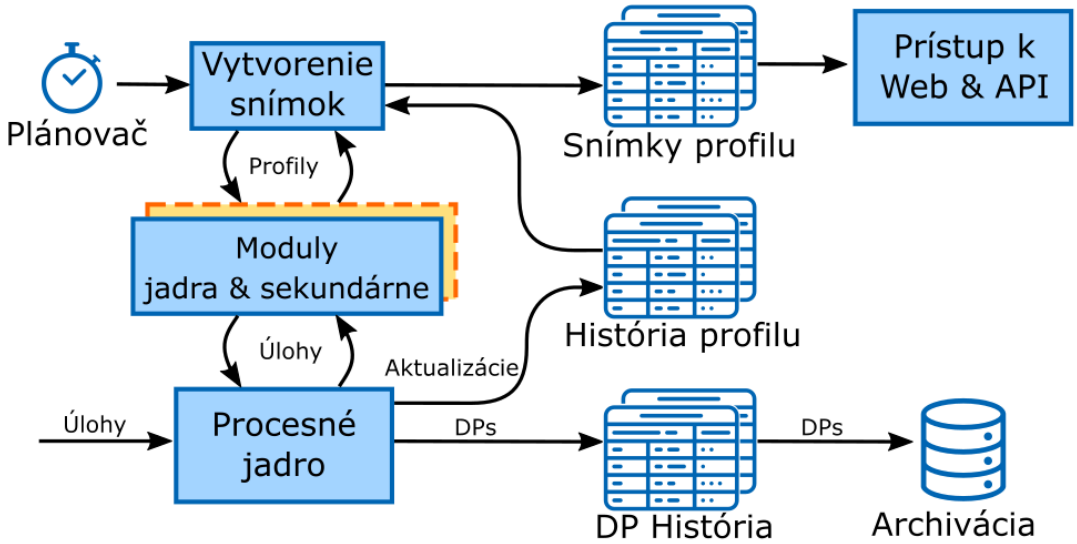
Architektúra DP³, ako je ukázané na obrázku 2.4, sa skladá z viacerých komponentov. Komponenty, ktoré poskytuje priamo DP³ sú na obrázku modrou farbou. Prvou časťou je HTTP API, ktoré je postavené na Fast API rámci. API prijíma a validuje prichádzajúce data-pointy a posiela ich ďalej na distribúciu úloh. Ďalej poskytuje prístup k databáze pre webové rozhranie a užívateľské programy. Viac o API sa píše v podsekcii 2.2.4. Distribúcia úloh je vykonávaná pomocou RabbitMQ frontov, ktoré rozdeľujú úlohy medzi jednotlivé paralelné procesy. Hlavná a najrozsiahlejšia časť platformy beží v paralelných pracovných procesoch (worker processes). V pracovnom procese je procesné jadro, ktoré vykonáva všetky aktualizácie a komunikuje s modulmi jadra a so sekundárnymi modulmi, ktoré sú aplikačne-špecifické. Viac o sekundárnych moduloch nájdete v podsekcii 2.2.7. Jadrom celej platformy je pochopiteľne databáza. DP³ v súčasnosti používa NoSQL databázu MongoDB. Stálosť databázy zabezpečujú skripty platformy aby nevznikali problémy pri spracovávaní dát. HTTP API a pracovné procesy používajú databázové API k prístupu



Obr. 2.4: Architektúra DP³

do databázy. Poslednou časťou je webové rozhranie, ktoré je aplikačne-špecifické, ale DP³ poskytuje štartovnú alternatívu všeobecného webového rozhrania na vizualizáciu údajov ako zásuvný modul pre Grafanu. Aplikačne-špecifické komponenty, ktoré sú zobrazené na obrázku žltou farbou sú sekundárne moduly. Ďalej tu patria aj primárne moduly. Rozdiel medzi primárnymi a sekundárnymi modulmi je ten, že zatiaľ čo primárne moduly posielajú data-pointy pomocou HTTP API, sekundárne moduly iba pracujú s dátami, ktoré sa už v systéme nachádzajú. Môžu pridávať nové informácie, spájať ich, prípadne ich meniť alebo mazať. Viac o primárnych moduloch je v podsekcii 2.2.6 a o sekundárnych v podsekcii 2.2.7.

Tok dát



Obr. 2.5: Dátový tok v DP³

Obrázok 2.5 ukazuje dátový tok v platforme DP³ a je priblížením pohľadu na pracovný proces z obrázka architektúry platformy 2.4. Prichádzajúce úlohy, ktoré obsahujú

data-pointy z HTTP API sa odovzdávajú spätným volaniam sekundárnych modulov nakonfigurovaným na nový data-point alebo okolo vytvorenia entity. Tieto moduly môžu dáta upraviť, vytvoriť ďalšie data-pointy alebo čokoľvek iné. Keď je všetko dokončené dáta sa zapíšu do dvoch kolekcií databázy. Kolekcia história DP (data-point), kde sú uložené surové data-pointy až dokiaľ nie sú archivované. Kolekcia história profilov obsahuje dokument pre každú entitu s relevantnou históriou. V databáze sú tieto kolekcie pomenované *{entity}#raw* a *{entity}#master*.

DP³ periodicky vytvára nove profily snímok spustené plánovačom. Snímky z histórie profilu redukujú celú históriu atribútu entity na jednu hodnotu, čím vytvárajú aktuálnu hodnotu atribútu. Frekvenciu vytvárania snímok je možné nakonfigurovať. Snímky sa vytvárajú na základe jednotlivých entít, ale všetky prepojené entity sa spracúvajú v rovnakom čase. To znamená, že pri vytváraní snímok môžu zaregistrované spätné volania snímok pristupovať ku všetkým prepojeným entitám pre svoje potreby korelácie údajov. Po dokončení všetkých spätných volaní sa snímka zapíše do kolekcie snímky profilu. Táto kolekcia je potom prístupná užívateľovi cez HTTP API. Kolekcia je v databáze prístupná pod menom *{entity}#snapshots*.

2.2.2 Dátový model

Základnými prvkami dátového modelu DP³ sú typy entít, pričom každá entita obsahuje množinu atribútov. Každý atribút má nejakú hodnotu, časovú pečiatku (môže sa uložiť história predchádzajúcich hodnôt) a voliteľne hodnotu spoľahlivosti.

Typy atribútov

Typy atribútov určujú ako sa atribút bude správať v čase, aké hodnoty môžu obsahovať a iné. V DP³ sú tieto 3 typy atribútov:

Plain

Bežné atribúty s iba jednou hodnotou určitého dátového typu. Nie je uložená história, iba časová značka poslednej zmeny. Bežné atribúty sú veľmi užitočné pre údaje z externého zdroja, keď nie je potrebné ukladať históriu, ale len aktuálnu hodnotu. Taktiež sú vhodné pre poznámky a iné ručne zadané informácie.

Observation

Atribúty s históriou hodnôt v určitom čase alebo časovom intervale. Následne môžeme z týchto hodnôt odvodiť hodnotu v ľubovoľnom čase. Každá hodnota môže mať priradenú spoľahlivosť. Tieto atribúty môžu mať jednu hodnotu (*single-value*) alebo viac hodnôt (*multi-value*), to znamená viac aktuálnych platných hodnôt v jednom časovom bode. Atributy tohto typu sú veľmi užitočné pre údaje, pri ktorých je potrebná aktuálna hodnota aj história alebo viacero platných hodnôt naraz.

Časové rady

Jedna alebo viac číselných hodnôt pre určitý čas. V tomto type atribútu: história > aktuálna hodnota. V skutočnosti sa neuvádza žiadna explicitná aktuálna hodnota. Časové rady sú veľmi užitočné pre akýkoľvek druh analýzy založenej na histórii a zaznamenávanie udalostí/zmien.

Môžu byť:

- pravidelné: odber vzoriek je pravidelný,
- nepravidelné: vzorkovanie je nepravidelné,
- nepravidelné intervaly: vzorkovanie je nepravidelné a obsahuje dve časové značky (od kedy do kedy boli poskytnuté údaje zhromaždené).

Vzťahy

Špeciálnym dátovým typom atribútu je *link*, ktorý reprezentuje vzťahy medzi entitami. Vzťahy môžu byť reprezentované s históriou alebo bez nej. V závislosti od toho, či je história dôležitá, môžu byť nakonfigurované pomocou typu *plain* alebo *observation*. Vzťahy môžu obsahovať ďalšie údaje, ak to zodpovedá potrebám modelovania. Vzťahy sú veľmi užitočné pre akýkoľvek druh vzťahu medzi entitami alebo prepojenie dynamických entít s entitami so statickými údajmi.

2.2.3 Správa histórie

V tejto podsekcii sa píše o správe histórie hodnôt v DP^3 a vysvetľuje niekoľko pojmov, ktoré sa používajú na správu údajov atribútov s históriou, nazývaných *observations*. DP^3 zakladá svoju prácu s týmito atribútmi na myšlienke aktuálnej hodnoty atribútu v danom čase. DP^3 vytvára z histórie každej entity výsek pre konkrétny čas a prezentuje len tieto aktuálne hodnoty všetkých atribútov. Pri atribútoch typu *plain* je to triviálne, pretože majú len jednu hodnotu, ale pri *observations* je to zložitejšie. Vzniká tam veľa rôznych stavov, ktoré treba vyriešiť, rôzne druhy intervalov platnosti, atribúty, ktoré majú hodnotu spoľahlivosti, atribúty s prekrývajúcimi sa data-pointmi. Spájanie susedných data-pointov s rovnakou hodnotou. Hodnoty sa interpolujú a extrapolujú z blízkych data-pointov atď. Táto podsekcia popisuje všetky tieto problémy súvisiace s časom.

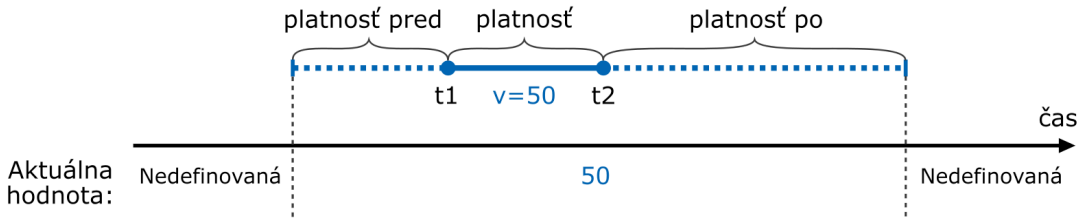
Platnosť data-pointu

Každý data-point má interval platnosti, čo je časový interval, v ktorom sa hodnota atribútu považuje za platnú. Tento interval je špecifikovaný v samotnom data-pointe pomocou počiatočnej a konečnej časovej značky, *t1* a *t2*. Tieto časové značky by mali byť nastavené na interval, v ktorom bola táto hodnota skutočne pozorovaná, a tak sa hodnota považuje za platnú v intervale *[t1, t2]* a neplatnú mimo neho. Tento základný interval je rozšírený prostredníctvom parametrov *pre_validity* (platnosť pred) a *post_validity* (platnosť po), ktoré sú uvedené v definícii atribútu.

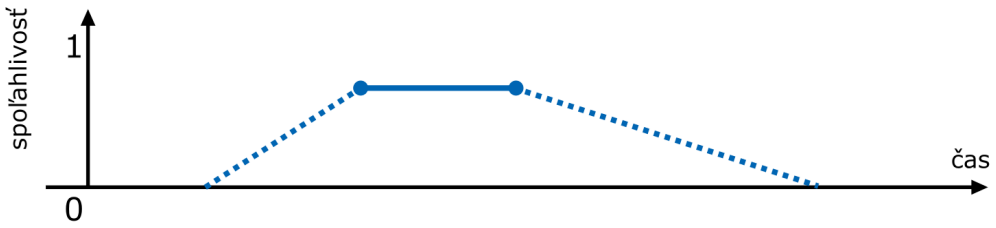
Vo všeobecnosti je interval platnosti data-pointu s časovými značkami *t1* a *t2* *[t1 - pre_validity, t2 + post_validity]* ako je ukázané na obrázku 2.6.

Spoľahlivosť

V konfigurácii atribútu, môže byť nastavený aj parameter spoľahlivosti. Tento parameter určuje, či má DP^3 sledovať hodnotu spoľahlivosti priradenú ku každému data-pointu. Hodnota spoľahlivosti je číslo medzi 0 a 1, kde 0 znamená, že hodnota je úplne nespoľahlivá (dokonca neplatná), a 1 znamená, že je úplne spoľahlivá. Ako už bolo spomenuté, všetky data-pointy by teraz mali obsahovať hodnotu spoľahlivosti c, čo je číslo medzi 0 a 1. Ak



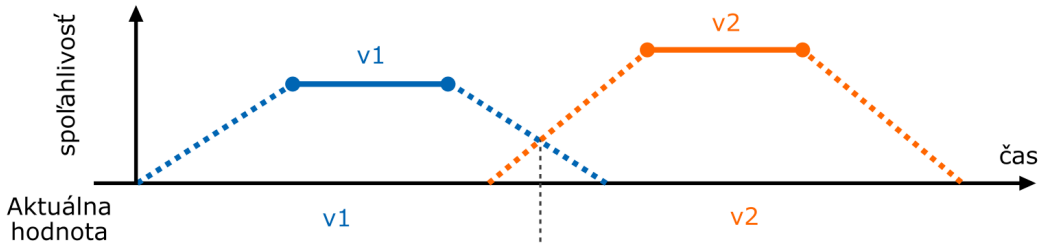
Obr. 2.6: Intervaly platnosti pre jeden data-point



Obr. 2.7: Interpolácia spoľahlivosti pre jeden data-point.

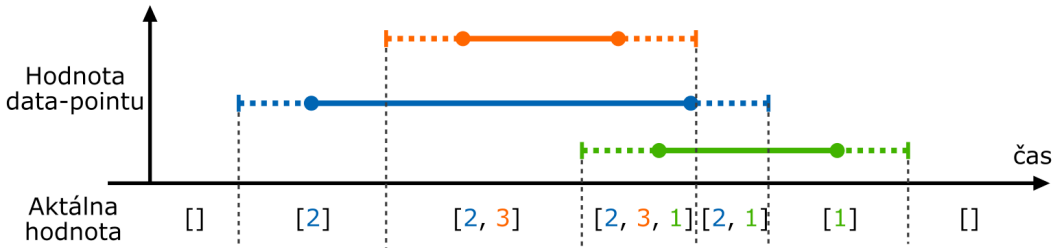
data-point neobsahuje hodnotu spoľahlivosti je implicitne nastavená na 1. V intervale medzi *t1* a *t2* daného data-pointu sa hodnota spoľahlivosti udržiava konštantná. V intervaloch platnosti pred/po (*pre/post-validity*) sa hodnota spoľahlivosti lineárne interpoluje medzi 0 a hodnotou *c* v intervale prísnej platnosti *[t1, t2]*. Toto je ukázané na obrázku 2.7.

Prekrývajúce sa data-pointy



Obr. 2.8: Vyhľadanie aktuálnej hodnoty s prekrývajúcimi sa platnosťami data-pointov

Na obrázku 2.8 sú 2 data-pointy, jeden s hodnotou *v1* a druhý s hodnotou *v2*. V tomto príklade sa intervaly *[t1, t2]* neprekrývajú, ale rozšírené intervaly *[t1 pre_validity, t2 + post_validity]* áno. Ako je vyznačené na obrázku, za aktuálnu hodnotu sa považuje tá, pri ktorej je interpolovaná hodnota spoľahlivosti vyššia. Ak by atribút nemal žiadne hodnoty spoľahlivosti, výsledok by bol rovnaký, ale základná "falošná" hodnota spoľahlivosti použitá na výpočet by bola vždy 1.



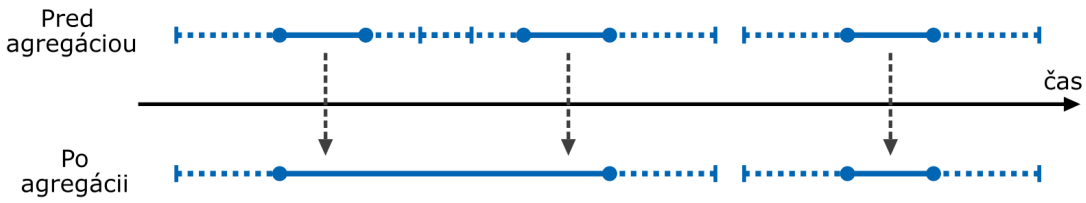
Obr. 2.9: Správanie atribútu s parametrom *multi_value*

Viachodnotové data-pointy

V konfigurácii atribútu, ktorý je typu *observation*, môže byť nastavený, ak to je z pohľadu dát vhodné, parameter *multi_value*. To znamená, že atribút bude môcť v jednom časovom bude nadobúdať viacero rôznych hodnôt. DP³ sa tomu prispôsobí a podľa toho spracuje data-pointy. Najdôležitejšie je, že napriek tomu, že jednotlivé data-pointy sú typu celé číslo (int), aktuálna hodnota bude zoznam celých čísel a aktuálna spoľahlivosť bude zoznamom desatinných čísel, ak je to povolené. Príklad takejto situácie je na obrázku 2.9. V prípade, že by v databáze boli presne tieto údaje, ale nebol nastavený parameter *multi_value*, správanie by bolo získanie aktuálnej hodnoty ako jedného celého čísla a aktuálnej dôveryhodnosti (ak je nastavená) ako jedného desatinného čísla. Aktuálna hodnota by bola tá s najvyššou hodnotou spoľahlivosti. V prípade viacerých prekrývajúcich sa data-pointov s rovnakou spoľahlivosťou sa aktuálnou implementáciou vyberie jeden na základe aktuálneho poradia, ale nie sú definované žiadne presné kritériá. Vo všeobecnosti platí, že ak nie je nastavený parameter *multi_value*, je dôležité aby vstupné moduly neposielali prekrývajúce sa data-pointy, inak nie je presné správanie definované a môže nastať vo výstupe dátová nestabilita.

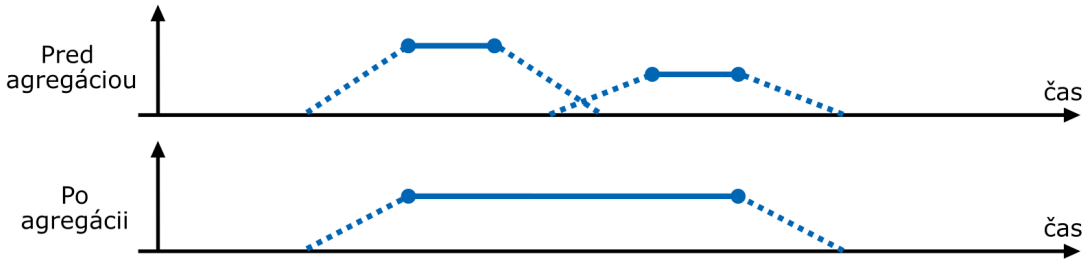
Spájanie data-pointov

Na úsporu miesta na disku DP³ spája po sebe idúce data-pointy s rovnakou hodnotou a prekrývajúcimi sa intervalmi platnosti. Tento postup sa označuje aj ako agregácia data-pointov. Hranica toho, čo sa považuje za po sebe idúce pre dva data-pointy A a B, je definovaná ako $A.t2 + post_validity >= B.t1 - pre_validity$. Inými slovami, pokiaľ sa intervaly platnosti dvoch data-pointov prekrývajú, považujú sa za po sebe idúce. Tento prípad je zobrazený na obrázku 2.10.



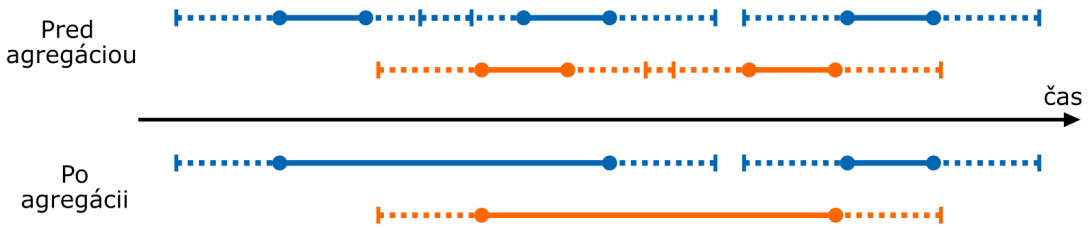
Obr. 2.10: Dva po sebe idúce data-pointy s rovnakou hodnotou sa zlúčia.

Táto agregácia sa vykonáva periodicky, takže aj keď data-pointy prídu mimo poradia, budú správne zlúčené. Ak majú hodnoty nastavenú spoľahlivosť, výsledná spoľahlivosť bude priemerom oboch hodnôt, ako to ukazuje obrázok 2.11.



Obr. 2.11: Spoľahlivosť zlúčených data-pointov sa spriemeruje.

Pri viachodnotových atribútoch sa toto spájanie uskutočňuje rovnakým spôsobom, ale pre každú hodnotu zvlášť. To znamená, že ak máte dva data-pointy s prekrývajúcimi sa intervalmi platnosti, ale s rôznymi hodnotami, nebudú zlúčené, ale budú ponechané ako samostatné data-pointy. Data-pointy s rovnakou hodnotou sa zlúčia ako predtým. Je to znázornené na obrázku 2.12.



Obr. 2.12: Každá hodnota viachodnotového atribútu sa zlúči samostatne.

2.2.4 API

DP³ má HTTP API, ktoré sa dá použiť na odosielanie data-pointov a čítanie údajov uložených v databáze DP³. Nasleduje zoznam API koncových bodov, ktoré platforma ponúka:

- **GET /** - skontroluje, či API beží,
- **POST /datapoints** - vloží data-pointy do DP³,
- **GET /entity/<entity_type>** - vráti zoznam aktuálnych snímok všetkých entít daného typu,
- **GET /entity/<entity_type>/<entity_id>** - získa údaje entity s daným ID entity,
- **GET /entity/<entity_type>/<entity_id>/get/<attr_id>** - získa hodnotu atribútu,

- **GET /entity/<entity_type>/<entity_id>/set/<attr_id>** - nastaví hodnotu atribútu,
- **GET /entity/<entity_type>/_/distinct/<attr_id>** - získa rozličné hodnoty atribútov a ich počty na základe najnovších snímok,
- **DELETE /entity/<entity_type>/<entity_id>** - odstráni údaje entity pre dané ID,
- **POST /entity/<entity_type>/<entity_id>/ttl** - predĺži TTL (čas životnosti) špecifikovanej entity,
- **GET /entities** - vráti zoznam konfigurácie entít,
- **GET /control/<action>** - pošle preddefinovanú akciu do frontu na vykonanie,
- **GET /telemetry/sources_validity** - získa informácie o platnosti dátových zdrojov.

2.2.5 Konfigurácia

Platforma DP³ je veľmi dobre konfigurovateľná. Okrem konfigurácie jednotlivých entít a ich atribútov je možné nakonfigurovať aj procesy, ktoré platforma vykonáva a iné časti platformy.

Konfigurácia platformy

DP³ po nainštalovaní ponúka základnú konfiguráciu, ktorú je potrebné upraviť pre potreby konkrétnej aplikácie. Nasleduje zoznam konfiguračných súborov aj s popisom ich funkcie:

- **db_entities** - Konfiguračný priečinok databázových entít.
- **modules** - Konfiguračný priečinok modulov.
- **api.yml** - Konfiguračný súbor API.
- **control.yml** - Konfiguračný súbor riadiaci povolené akcie koncového bodu */control*.
- **database.yml** - Pripojenie k databáze.
- **event_logging.yml** - Sledovanie operácií aplikácie pomocou Redis.
- **garbage_collector.yml** - Odstránenie entít s uplynutou dobou životnosti.
- **history_manager.yml** - Ako často sa vykonáva správa histórie.
- **processing_core.yml** - Nastavenia hlavných pracovníkov aplikácie.
- **snapshots.yml** - Ako často sa robia snímky entít.
- **updater.yml** - Pravidelné aktualizácie všetkých entít v dlhšom časovom horizonte.

Konfigurácia databázových entít

Vďaka konfigurácii databázových entít si vie užívateľ nadefinovať niečo ako schému databázy presne podľa jeho požiadavkou. Entity sa dajú predstaviť ako triedy v objektovo-orientovanom programovaní. Entita je definovaná jednoducho. Skladá sa z identifikátora a mena entity. Entitu tvoria atribúty, ktoré sú nositeľmi informácií o entite. Ako sa píše v podsekcii 2.2.2 atribúty majú viacero typov. Ich konfigurácia záleží na type atribútu, na informácii, ktorú obsahuje.

2.2.6 Primárne moduly

Primárne moduly nie sú priamou súčasťou platformy DP^3. Na rozdiel od sekundárnych modulov 2.2.7 nemajú žiadnu pevnú štruktúru. Procesné jadro s nimi nijako nekomunikuje. S platformou interagujú iba pomocou HTTP API. Používajú sa zväčša ako primárny dátový zdroj. Získané dáta posielajú pomocou API do DP^3. Primárny modul môže rovnako aj pomocou API získavať a priamo upravovať záznamy v databáze v prípade, že by to bolo pre modul potrebné. Na prácu s dátami a ich úpravy sa však používajú hlavne sekundárne moduly. Konfigurácia primárnych modulov, ich spustenie a riadenie behu sa musí riešiť mimo platformy.

2.2.7 Sekundárne moduly

DP^3 umožňuje používateľom vytvárať vlastné moduly na vykonávanie analýzy údajov špecifických pre danú aplikáciu. Moduly sa načítavajú pomocou architektúry podobnej zásuvným modulom a môžu ovplyvňovať tok dát od prvého okamihu po spracovaní požiadavky na vytvorenie data-pointu. Ako je opísané v podsekcii 2.2.1, DP^3 používa kategorizáciu modulov na primárne a sekundárne moduly. Rozdiel medzi primárnymi a sekundárnymi modulmi je taký, že primárne moduly posielajú do systému data-pointy pomocou HTTP API, zatiaľ čo sekundárne moduly reagujú na údaje prítomné v systéme. Napríklad menia tok údajov spôsobom špecifickým pre aplikáciu, odvodzujú ďalšie údaje na základe prichádzajúcich data-pointov alebo vykonávajú koreláciu údajov na snímkach entít. Všetky moduly musia byť podtriedou triedy *BaseModule*. Ak trieda nie je podtriedou triedy *BaseModule*, nebude načítaná a aktivovaná hlavným pracovným procesom DP^3. Moduly môžu byť nakonfigurované vlastným konfiguračným súborom vo formáte *YAML* ak je to potrebné. Aby modul mohol niečo robiť, musí prečítať poskytnutú konfiguráciu a zaregistrovať spätné volania na vykonanie dátovej analýzy pomocou objektu *registrar*. Je možné zaregistrovať širokú škálu objektov. Nasleduje zoznam druhov spätných volaní, ktoré DP^3 poskytuje:

- periodické spätné volania spúšťača CRON,
- spätné volania v rámci spracovania,
  - na začiatku spracovania úlohy,
  - povolenie na vytvorenie entity,
  - po vytvorení novej entity,
  - po prijatí nového atribútu,
  - pred vytvorením snímky pre časové rady - umožňujú spracovať nahromadené údaje časových radov na atribúty typu *plain* / *observation*,
- korelačné spätné volania,

  - počas vytvárania snímky - umožňujú vykonávať analýzu údajov snímky,
  - pred vytvorením snímky,
  - po dokončení vytvárania snímky,
- spätné volania periodickej aktualizácie.

Kapitola 3

Návrh riešenia

Táto kapitola rieši problematiku návrhu optimalizácie systému NERD. Analyzujem tu jednotlivé časti a hlavne moduly systému NERD a navrhujem vhodnú implementáciu pre fungovanie s platformou DP³. Systém NERD je aktuálne dobre fungujúcou službou. Problémom je, že jeho architektúra už nie je dostatočná pre smerovanie, ktorým sa chce uberať združenie CESNET pri vývoji tohto nástroja. Ako už bolo spomenuté platforma DP³ je používaná vo viacerých systémoch a službách. Preto prechod na túto novšiu platformu je výhodný z viacerých dôvodov napríklad jednoduchšia údržba a vývoj do budúcna, ale hlavne možnosť rozširovania systému NERD o nové funkcionality, ktoré súčasný návrh systému neumožňuje.

3.1 Porovnanie architektúry NERD a DP³

Aktuálna architektúra, ktorú systém NERD používa je navrhnutá presne pre jeho potreby. Preto má ale nízku znovu použiteľnosť, čo je v prípade prevádzky viacerých systémov s rôznymi platformami, ako je to v združení CESNET, veľká nevýhoda. Ako už bolo spomenuté platforma DP³ a jej architektúra vychádza práve z architektúry NERDu. Obe majú nejaké procesné jadro, ktoré pristupuje k databáze a primárne a sekundárne modely, ktoré interagujú so systémom podobným spôsobom v oboch prípadoch. Z tohto pohľadu prechod na novú platformu DP³ nie je veľký problém. Moduly, ktoré vkladali do systému dáta z primárnych dátových zdrojov sa stanú primárnymi modulmi v DP³. Sekundárne moduly z NERDu budú rovnako sekundárnymi modulmi aj v DP³. Zložitejšie to už však je s procesným jadrom a samotným spracovávaním jednotlivých informácií, ktoré do systému prídu. V systéme NERD je práca s dátami priamočiarejšia. Modul nadefinuje čo sa s dátami, ktoré posiela má urobiť. Či sa má záznam vytvoriť alebo iba aktualizovať prípadne nejaký záznam vymazať. Zároveň definuje poslednú zmenu záznamu a dokedy je záznam v systéme platný V DP³ moduly vytvárajú data-pointy, ktoré následne jadro platformy ukladá a ako je opísané v podsekcii 2.2.3 vyhodnocuje aktuálnu hodnotu podľa histórie. Zároveň platnosť jednotlivých data-pointov neurčujú moduly, ale záleží na časoch *t1* a *t2* a na konfigurácii konkrétneho atribútu, preto nie je potrebné aby data-point explicitne určoval dĺžku života ako to je v súčasnej verzii systému NERD. Platforma DP³ je celkovo všeobecnejšia a jej prístup k spracovaní dát umožňuje jednoduchší vývoj systému NERD v budúcnosti. Dôležitým rozdielom z hľadiska architektúry je aj HTTP API, ktoré používa platforma DP³. NERD používa HTTP API len pre webový server a bežné API na prístup k dátam pre užívateľov. Platforma DP³ okrem toho používa API aj na komunikáciu s primárnymi mo-

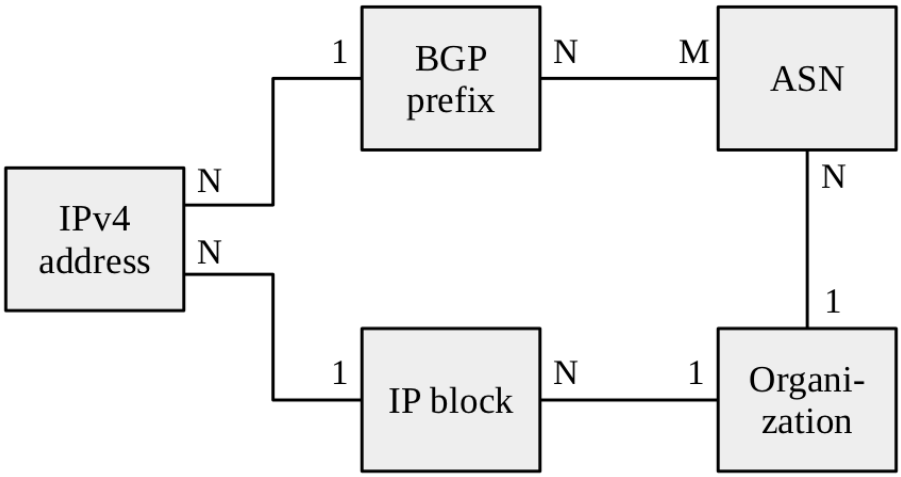
dulmi. Zatiaľ čo v NERDe moduly vkladajú úlohy priamo do hlavného frontu úloh v DP^3 sa data-pointy posielajú na API. Rovnako aj prístup k databázovým štruktúram je v NERDe priami. Modul sa musí napojiť priamo na databázové API a čítať z neho. To však môže spôsobiť niekedy dátovú nekonzistenciu. V DP^3 je prístup k záznamom v databáze cez API jadra . Vďaka tomu je platforma DP^3 lepšie zapuzdrená a priamy prístup k údajom v databáze má iba procesné jadro platformy. Použitie API na komunikáciu s primárnymi modulmi má aj ďalšiu výhodu. Tou je možnosť prevádzky modulov aj na iných strojoch. NERD je v súčasnej podobe nútený bežať len na jednom, čo nepredstavuje veľký problém, ale na druhú stranu limituje prípadné rozšírenie.

3.2 Rozbor systému NERD

V sekcii 2.1 som popisoval jednotlivé časti systému. Teraz sa zameriam konkrétne časti back endu. Back end alebo tiež nazývaný NERD démon by sa dal rozdeliť na 3 respektíve 4 časti. Prvou je procesné jadro, ktoré zabezpečuje prácu s dátami a ich ukladanie do databázy. To nebude v implementácii potrebné, pretože platforma DP^3 poskytuje všetko čo je potrebné. Druhou časťou, naväzujúcou na procesné jadro sú sekundárne moduly. Tie bude potrebné reimplementovať na DP^3. Avšak úlohy niektorých modulov preberá už samotná platforma preto tie už naďalej potrebné nebudú. Treťou časťou back endu sú primárne moduly. Tie by sa dali teoreticky rozdeliť ešte na 2 časti. Prvé sú primárne moduly, ktoré prijímajú dáta z primárnych zdrojov NERDu a posielajú ich do procesného frontu. Tieto moduly budú potrebné a reimplementované pre nový NERD používajúci DP^3. Druhou časťou primárnych modulov sú moduly, ktoré zabezpečujú beh systému ako celku alebo zabezpečujú aktualizáciu atribútov a podobne. Tieto moduly už nebudú potrebné, lebo ako to bolo s niektorými sekundárnymi modulmi ich funkcionalitu ponúka samotná platforma. Ďalej v tejto kapitole sa zameriam len na časti back endu, ktoré budú reimplementované. Stručne opíšem princíp fungovania jednotlivých modulov a spôsob ich reimplemetácie. Zameriam sa taktiež na konfiguráciu jednotlivých atribútov pre platformu DP^3 aby zabezpečili správne fungovanie systému NERD a zachovali logiku pôvodnej implementácie. Vo všeobecnosti bude potrebné zmeniť spôsob a formu získaných informácií o entitách z dôvodu iného rozhrania a dátového modelu procesného jadra platformy DP^3. Úpravy v získavaní informácií nebudú zväčša potrebné.

3.3 Dátový model

Dátový model systému NERD v súčasnosti pozostáva z viacerých entít ako sa píše v podsekcii 2.1.2. Pochopiteľne najrozšírenejšou entitou, ktorá sa v systéme nachádza je IP adresa. Entity tohto typu poskytujú aj hlavnú informačnú hodnotu, ktorú NERD ponúka. Všetky entity majú medzi sebou vzťahy ako to ukazuje obrázok 3.1. Tieto ostatné entity sa avšak ukázali ako nie príliš dôležité a užívateľmi málo využívané. Pritom však vytvárajú pomerne zložitý dátový model. Preto vedúci práce rozhodol, že v prvej verzii tieto entity nebudú podporované a nebudú sa modelovať a ich podpora sa nechá pre prípadné budúce rozšírenie. Táto úprava neovplyvní veľmi informačný aspekt systému, ale výrazne zjednoduší dátový model a vďaka tomu aj zníži výpočtovú náročnosť.



Obr. 3.1: Aktuálne podporované entity a ich vzťahy. Obrázok bol prevzatý z [1].

3.4 Primárne moduly

V tejto sekcii analyzujem primárne moduly, ktoré budú reimplementované pre novú verziu systému NERD. Všetky moduly v NERDe používajú spoločnú konfiguráciu. Je viacero týchto spoločných konfiguračných súborov. Dôležité pre reimplemnetáciu sú všeobecná konfigurácia - *nerd.yml* a konfigurácia back endu - *nerdd.yml*. V týchto súboroch má každý komponent vrátane modulov všetky potrebné informácie na jeho fungovanie. Nachádza sa tu konfigurácia hlavných pracovných procesov a tiež napríklad API kľúče, ktoré jednotlivé moduly potrebujú. V súčasnej verzii sa pomerne zložito tieto informácie z týchto súborov vyčítavajú. Moduly zároveň kontrolujú či je konfigurácia systému správna čo nepokladám za potrebnú respektíve vyžadovanú funkciu. Preto toto nebude ani v novej verzii systému potrebné. DP^3 ako sa píše v podsekcii 2.2.5 používa na konfiguráciu viacero súborov, ktorý každý zabezpečuje konfiguráciu istej oblasti platformy. Zároveň konfigurácia sekundárneho modulu v DP^3 je automaticky načítaná zo súboru s rovnakým názvom ako je názov modulu. To znamená, že každý sekundárny modul má svoj vlastný konfiguračný súbor, kde má všetko potrebné pre svoje fungovanie. Tento prístup som použil aj pre primárne moduly. Zvolil som to aby všetky moduly, primárne aj sekundárne, mali jednotný spôsob konfigurácie.

3.4.1 Modul blacklists

Úlohou modulu *Blacklists* (Čierne listiny) je, ako to už z názvu vyplýva, sťahovať rôzne čierne listiny pre uloženie v systéme. Modul si načíta zoznam čiernych listín z konfiguračného súboru a stiahne zoznam IP adries. Keďže rôzny poskytovatelia čiernych listín majú rôzne spôsoby ukladania je potrebné ich správne upraviť (parsovať) pomocou regulárnych výrazov. Takýmto spôsobom vznikne zoznam IP adries, ktoré sa nachádzajú na konkrétnej čiernej listine. Každá čierna listina má unikátny identifikátor (ID), to je spravidla meno danej čiernej listiny. Pre každú IP adresu je vytvorená úloha, ktorá je vložená do frontu úloh. Tá obsahuje IP adresu, identifikátor čiernej listiny na ktorej sa nachádza, čas stiahnutia a maximálnu dĺžku života. Modul sťahuje jednotlivé čierne listiny počas dňa v rôzne časy podľa konfiguračného súboru. Čierne listiny sa sťahujú každý deň. Takýmto spôsobom sa aktualizujú záznamy v databáze.

Prvou otázkou je, ako presne informácie o čiernych listinách v databáze DP³ uložiť. Buď môže byť pre každú čiernu listinu samostatný atribút (pomenovaný napríklad *bl-<názov>*, s pravdivostnou hodnotou), alebo to môže byť jeden atribút (*blacklists*) s viacerými hodnotami a dátovým typom reťazec. Prvý spôsob (samostatné atribúty) by mal výhodu v tom, že by bolo možné explicitne uložiť neprítomnosť na čiernej listine. Ďalšou výhodou by bolo, že by to v budúcnosti umožnilo uložiť pre niektoré čierne listiny niečo viac ako len či sa daná IP adresa nachádza alebo nenachádza na čiernej listine (väčšina čiernych listín neposkytuje viac informácií, ale niektoré áno). Okrem toho by bolo možné pre každý z nich nastaviť inú platnosť pred/po (*pre/post_validity*), čo by mohlo byť užitočné, keď sa rôzne čierne listiny sťahujú s rôznou frekvenciou. Pridávanie/odstraňovanie čiernych listín však bude oveľa zložitejšie a bude si vyžadovať oveľa dlhší konfiguračný súbor a viac miesta v databáze. Ďalšou nevýhodou bude aj nutnosť kontrolovať pomocou databáze alebo iným spôsobom ukladania dát, či sa IP adresa na čiernej listine nachádzala minulý deň a dnes už nie a následne nastaviť jej neprítomnosť. Druhý spôsob (viachodnotový) by bol jednoduchší a úspornejší, len by sa nedalo explicitne rozlíšiť, či adresa je/nebola na čiernej listine, alebo či sa jej systém z nejakého dôvodu (zlyhanie, chyba) ani nedotkol (ale to nie je možné rozlíšiť ani v súčasnej verzii NERD). Ani ďalšie výhody uvedené vyššie by sa nedali využiť. Rozlíšiť či sa adresa nachádza na čiernej listine by sa dalo iba správnym nastavením hodnôt pred/po validity. Keďže sa čierne listiny sťahujú každý deň, nastavenie po validity na 24 hodín je logické. Ak sa na ďalší deň neobnoví prítomnosti na čiernej listine, časom platnosť data-pointu vyprší a IP adresa nebude klasifikovaná ako prítomná na konkrétnej listine. Neprítomnosť síce nebude skoková, ale prejaví sa.

Z dôvodu jednoduchosti a komplikáciám s prvým spôsobom som zvolil druhý spôsob reprezentácie prítomnosti IP adresy na čiernych listinách. Pre reimplementáciu bude teda potrebné, ako pri všetkých moduloch, zmeniť spôsob odosielania dát. Modul vytvorí jeden data-point pre každú IP adresu, ktorá sa na čiernej listine nachádza. To znamená, že ak sa nachádza IP adresa na viacerých čiernych listinách pre každý výskyt bude vytvorený jeden data-point. Data-point bude obsahovať IP adresu, identifikátor čiernej listiny a čas kedy boli záznamy z čiernej listiny stiahnuté, ako to je aj v súčasnej verzii NERDu.

3.4.2 Modul Warden receiver

Jeden z hlavných zdrojov dát pre NERD sú hlásenia zo systému Warden, ktorý prevádzkuje združenie CESNET. Sťahovanie dát z Wardenu zabezpečuje samostatný program *warden filer* (nie je súčasťou systému NERD). Je to démon, ktorý pravidelne sťahuje nové správy z Warden servera a ukladá ich ako súbory do určeného priečinku (napríklad */data/warden_filer/incoming*). Odtiaľ ich potom číta warden recevier, ktorý je súčasťou systému NERD. Správy vo Wardene sú interne číslované a program *warden filer* si pamätá, ktorú správu stiahol ako poslednú. Potom si pri každom dopyte na server vyžiada všetky správy s vyšším poradovým číslom a opäť si uloží posledné číslo. Takže aj keď príjem chvíľu neprebieha, napríklad z dôvodu krátkodobého výpadku alebo dosiahnutia limitu počtu správ v priečinku, nie je to problém a keď sa spustí, stiahnu sa všetky správy, ktoré medzitým prišli do wardenu.

Samotný modul warden receiver (warden prijímač) pracuje nasledovne. Modul číta warden správy z priečinka, kde ich uložil *warden filer*. Keď dáta modul prečíta, súbor vymaže. To zabezpečí, že rovnaké dáta už znova neprečíta a nemusí si nikde ukladať informácie o tom, ktorý súbor už čítal. Zároveň zabezpečuje priebežné čistenie priečinka, takže nebude za bežných okolností dosiahnutý jeho limit, ktorý by pozastavil sťahovanie nových správ.

Po tom, čo je správa prečítaná, je overený jej formát a zároveň sú vyfiltrované len tie správy, ktoré spĺňajú definované pravidlá. Inak povedané použité sú tie správy, ktoré pre systém NERD prinášajú nejakú informačnú hodnotu. Následne modul vloží novú úlohu do frontu. Úloha obsahuje IP adresu, dátum detekcie udalosti, čas skončenia udalosti a to najpodstatnejšie názov uzlu, na ktorom bola škodlivá udalosť (aktivita) detekovaná a zoznam kategórií danej udalosti.

Napriek tomu, že je modul celkovo zložitý a načítanie a triedenie udalostí z Wardenu nie je triviálne, dátový návrh je jednoduchý. Udalosť je platná v nejakom rozmedzí nie len v jednom bode. Preto bude dôležitý počiatočný aj koncový čas (*t1* a *t2*). Vo warden udalosti je však viacero časov, ktoré by mohli nadobudnúť hodnoty *t1* a *t2*. Keďže nie vždy sú všetky nadefinované, vyberie sa vždy ten najvhodnejší.

3.4.3 Modul MISP receiver

Modul MISP receiver (prijímač) prijíma verejne dostupné správy z platformy MISP. MISP receiver prijíma len notifikácie o zmenách v MISP - ide teda vždy len o zmeny oproti aktuálnemu stavu (nová udalosť, nový atribút v udalosti, zmena atribútu atď.). Inicializáciu, stiahnutie všetkých údajov z MISP do NERD, vykonáva skript misp updater. Celkovo posiela MISP veľa druhov správ, ale pre NERD sú podstatné len niektoré. Ako bude prebiehať spracovanie prichádzajúcej správy záleží na jej druhu. Či sa jedná o novú udalosť, odstránenie udalosti, príchod nového, zmenu alebo vymazanie atribútu, oznámenie o pozorovaní. Všetky ostatné, iné typy správ sa zahodia a nijak do NERDu nevstupujú. Modul vytvára úlohu teda podľa typu správy a následne ju vloží do frontu úloh.

Z hľadiska návrhu atribútu je MISP správa asi najzložitejšia. Preto sa otvára viacero možností ako správu reprezentovať v novom systéme, keďže platforma DP^3, ako každá iná, má svoje limitácie. Jednou z možností je uložiť udalosti ako *observations*. Keď sa v MISP objaví udalosť s IP adresou, pre túto adresu sa v NERD uloží data-point so základnými informáciami o udalosti (ako v súčasnej verzii NERD). Tieto data-pointy by mali *platnosť pred* 0 a *platnosť po* nekonečno, takže by zostali platné, kým by prípadne neprišiel ďalší data-point, ktorý by sa generoval v prípade zmeny udalosti. Nový data-point by tak "prepísal" starý. Problémom je, že v MISP môže byť viacero udalostí s rovnakou adresou, takže atribút pre tieto data-pointy by musel byť viachodnotový. Lenže potom by prepisovanie nefungovalo. Platforma DP^3 by musela mať špeciálny typ medzi *single-value* a *multi-value*, kde by sa mohol definovať určitý podkľúč hodnoty data-pointu v tomto prípade *event_id* a data-pointy s rovnakým kľúčom by sa správali ako *single-value* to znamená, že by nová hodnota by prepísala staršiu, zatiaľ čo data-pointy s rôznymi kľúčmi by mohli existovať vedľa seba ako pri *multi-value*. DP^3 toto ale nevie. Druhou možnosťou je použiť podporu relácie v DP^3 a vytvoriť nový typ entity *MISP event*. Udalosti MISP by sa potom zrkadlili do NERD ako entity tohto typu a záznamy IP adries by obsahovali len odkazy na ne. To by bolo viac v súlade s dlhodobým plánom reprezentovať údaje viac ako graf, to znamená rôzne typy entít prepojené navzájom. Ďalšou výhodou je, že v databáze by neboli duplicitné informácie, ak je v udalosti viacero IP adries (v prvej verzii sa informácie o takejto udalosti ukladajú pre každú IP adresu znova). Na druhej strane by bol prechod pravdepodobne komplikovanejší. a ak by bolo potrebné v zázname IP niečo odvodiť na základe údajov v udalosti MISP (napr. kategóriu hrozby), dalo by sa to vypočítať len v rámci snímky. To ale asi nemá veľký význam.

Keďže prvá možnosť by si vyžadovala úpravy DP^3 a k druhej možnosti by sa vývoj systému NERD v rámci prechodu na prístup založený viac na grafoch postupne prepracoval

zvolil som ako lepšiu alternatívu druhú možnosť. Atribút entity IP adresa je typu *observation* a obsahuje iba odkaz na entitu *MISP event*. Ako reprezentovať entitu v databáze má tiež viacero možností. Prvou je reprezentovať celú správu jedným atribútom, ktorého dátový typ bude slovník obsahujúci správu. Druhou možnosťou je, že každá časť správy bude definovaná ako samostatný atribút. Druhá možnosť má tú výhodu, že je možné upraviť len jeden atribút bez nutnosti prepisovania celej správy. Nevýhodou je väčší počet data-pointov, ktoré je nutné odoslať. V oboch prípadoch sú atribúty typu *plain*, pretože nadobúdajú len jednu hodnotu a ich platnosť nevyprší. Z dôvodu lepšej prehľadnosti a keďže existuje prípad keď je vhodné meniť len jeden z atribútov, som zvolil druhú možnosť reprezentácie MISP udalosti. Nová verzia modulu bude teda vytvárať niekoľko data-pointov pre reprezentáciu MISP udalosti a jeden data-point entity IP adresa, ktorý bude na udalosť odkazovať.

3.4.4 Modul OTX receiver

Tento modul prijíma pulzy z OTX komunitného portálu. Prijímajú sa len pulzy, ktoré CESNET pre potreby NERDu odoberá. Modul pri prvom spustení stiahne všetky pulzy za posledných 80 dní. Následne sťahuje nové pulzy každé 4 hodiny. Čas posledného stiahnutia ukladá do súboru *otx_last_update.txt* aby v nasledujúcom sťahovaní získal všetky novo pridané pulzy. Modul potom vyberie z prijatých dát pre každú IP adresu potrebné informácie pre systém a vytvorí nový pulz, ktorý sa vloží do frontu úloh.

Na prvý pohľad prevod na novú platformu nebude zložitý. Avšak OTX pulz sa môže v čase meniť, rovnako ako MISP udalosť. Vzniká tu teda rovnaký problém aký bol pri module *MISP receiver*, viz. 3.4.3, pretože jedna IP adresa môže odkazovať na viacero pulzov zároveň preto musí byť povolená viachodnotovosť atribútu. Postup návrhu atribútov som preto zvolil rovnaký ako pri module *MISP receiver*. Atribút entity IP adresa je typu *observation* a obsahuje odkaz na entitu *OTX pulse* a ďalšie informácie špecifické pre každú IP adresu uložené v slovníku. Entita OTX pulz je potom reprezentovaná atribútmi typu *plain*, ktoré obsahujú informácie o pulze. Tento návrh má obrovskú výhodu v tom, že výrazne šetrí miesto na disku. Viacero IP adries, niekedy až tisícky, môže odkazovať na rovnaký pulz. Keby sa ukladal každý pulz spolu s informáciami rozdielnymi pre každú IP adresu, zaberalo by to násobne viac miesta a zároveň by sa nezapezpečila aktualizácia pulzu. Časová značka *t1* je čas aktualizácie pulzu (čas vytvorenia môže byť aj niekoľko rokov dozadu)a časová značka *t2* je exspirácia IP adresy ak je zadaná. Hodnota platnosti po je 30 dní. Pričom 30 dní je dĺžka za aký čas sa stane pulz neaktívny.

3.4.5 Modul DShield

Modul DShield pravidelne raz za deň sťahuje dáta od združenia SANS[1]. Dáta obsahujú počet nahlásení IP adresy a počet nahlasovateľov inak povedané cieľov daných útokov. Sťahovanie prebieha o 5:00 UTC, kedy je tam pridaný zoznam z minulého dňa. Po stiahnutí dát spočíta počet nahlásení a nahlasovateľov. Skontroluje či bola IP adresa nahlásená za deň viac ako 10 z viac ako 3 nahlasovateľov. V tom prípade modul vloží tieto informácie do frontu úloh NERDu.

Z implementačného hľadiska je modul veľmi jednoduchý. Keďže poskytuje 2 číselné hodnoty a to v pravidelnom intervale 1 deň je vhodné tento atribút navrhnúť ako pravidelnú časovú radu. Vďaka tomu bude možné sledovať útočnú aktivitu jednotlivých entít ako sa

[1] https://www.sans.edu

vyvíja v čase. Časová značka data-pointu pre každú IP adresu končí momentom stiahnutia záznamov a začína v rovnakom čase deň dozadu.

3.5 Sekundárne moduly

Sekundárne moduly, tiež nazývané zásuvné moduly pracujú v systéme NERD nad dátami, ktoré sa už v systéme nachádzajú. Fungujú na základe zaregistrovaných spätných volaní na nejakú udalosť, napríklad príchod novej entity. Sekundárne moduly v DP^3 pracujú na rovnakom princípe. Preto návrh prechodu týchto modulov je priamočiary. Sekundárne moduly NERDu sa stanú sekundárnymi modulmi aj v platforme DP^3. NERD má v súčasnej podobe okolo 20 modulov. Nie však všetky budú potrebné aj pre novú verziu. To je z dôvodu prechodu na novú kategorizáciu hrozieb, ale zároveň aj preto, lebo časť úloh, ktoré mali sekundárne moduly zabezpečujú na platforme DP^3 priamo jej komponenty, preto tieto moduly nie sú naďalej potrebné. Všetky moduly v NERDe majú rovnakú základnú architektúru. Obsahujú hlavnú triedu modulu, ktorá implementuje rozhranie *NERDModule*. DP^3 používa podobný model, keď všetky sekundárne moduly obsahujú hlavnú triedu modulu, ktorá je odvodená z abstraktnej triedy *BaseModule*. Preto budú potrebné z tohto pohľadu len malé zmeny. Keďže sekundárne moduly v systéme NERD registrujú spätné volania na niektoré udalosti bude potrebné zaregistrovať ekvivalentné spätné volania aj na platforme DP^3. Keďže však DP^3 nepodporuje posielanie udalostí, ktoré spôsobia spustenie funkcie spätného volania zaregistrovanú na danú udalosť, táto funkcionalita nebude v novej verzii podporovaná a ani potrebná. Viac o spätných volaniach v DP^3 v sekcii 2.2.7. NERD moduly nevkladajú úlohy do frontu úloh, ale predávajú úlohu jadru ako návratovú hodnotu zavolanej funkcie. Podobne pracujú aj moduly DP^3. V NERDe používajú všetky moduly spoločné konfiguračné súbory. Každý modul musí tieto súbory načítať a získať informácie, ktoré potrebuje načítať. V DP^3 má každý sekundárny modul vlastný konfiguračný súbor, ktorý má modul k dispozícii na začiatku a nemusí si ho načítavať, ale môže rovno čítať dôležité informácie. Bude teda potrebné vytvoriť pre každý sekundárny modul vlastný konfiguračný súbor. V tejto sekcii ďalej hovorím o návrhu použitia jednotlivých sekundárnych modulov na platforme DP^3. Zameriam sa na návrh atribútov, ktoré moduly pridávajú.

3.5.1 Modul DNS

Modul DNS zisťuje doménové meno adresy pomocou reverzného DNS vyhľadávania [4]. Modul má zaregistrované spätné volania na tieto udalosti:

- príchod novej entity,
- vyžiadanie aktualizácie doménového mena (v novej verzii nepodporované),
- aktualizácia raz za týždeň.

Modul prijme IP adresu, uplatní na ňu metódu reverzného DNS vyhľadávania a vráti výsledok procesnému jadru.

Úloha modulu je jednoduchá a rovnako aj štruktúra dát, ktoré získava. Keďže sa jedná len o obyčajný textový reťazec, ktorý označuje viac-menej stálu informáciu nový atribút bude typu *plain*. To znamená, že jeho platnosť nikdy nevyprší. Jeho aktuálnosť bude zabezpečená periodickým aktualizovaním raz za týždeň.

3.5.2 Modul DNS blacklist resolver

Modul vyhľadá adresu v sekundárnych čiernych listinách a aktualizuje záznam v databáze. K fungovanie je potrebné načítať zoznam čiernych listín z konfiguračného súboru. Funkcia spätného volania posiela požiadavky na všetky nakonfigurované čierne listiny a v prípade prítomnosti pridá túto čiernu listinu do zoznamu čiernych listín na ktorých sa adresa nachádza. Modul má zaregistrované spätné volania na tieto udalosti:

- príchod novej entity,
- vyžiadanie aktualizácie prítomnosti na sekundárnych čiernych listinách (v novej verzii nepodporované),
- aktualizácia raz za týždeň.

Informácie, ktoré získava tento modul sú prítomnosti entity na jednotlivých čiernych listinách. Jedná sa teda o rovnaký typ dát ako vkladá do systému primárny modul *blacklists* 3.4.1. Pre tento modul nie je potrebné vytvárať nový atribút, pretože použije už navrhnutý pre 3.4.1.

3.5.3 Modul Passive DNS resolver

Modul zisťuje doménu IP adresy pomocou pasívneho DNS a overuje prítomnosť získanej domény na čiernych listinách. Modul na začiatku načíta čierne listiny, ktoré sú uložené v Redis databáze systému. Hlavná funkcia získa pomocou CESNET Passive DNS API doménovú adresu a hľadá jej výskyt v načítaných čiernych listinách. Ak sa doména na čiernej listine nachádza pridá sa do výsledku trojica hodnôt: meno čiernej listiny, doména a hodnota 1, čo označuje prítomnosť na čiernej listine ak sa nenachádza pridá sa rovnaká trojica, ale s hodnotu 0. Modul má zaregistrované spätné volania na tieto udalosti:

- príchod novej entity,
- aktualizácia raz za deň.

Návrh nového atribútu je nasledovný. Atribút bude typu *observation* s platnosťou pred/po 24 hodín. Dátový typ bude slovník, obsahujúci 2 kľúče s hodnotami meno a doména. Hodnota prítomnosti domény na čiernej listine nebude potrebná. Je to z dôvodu, že atribút je typu *observation* a platnosť data-pointu vyprší po jednom dni. Ak sa na ďalší deň data-point neobnoví, jeho neprítomnosť bude implikovať aj neprítomnosť domény na čiernej listine.

3.5.4 Modul Redis blacklist

Modul zisťuje prítomnosť IP adresy na čiernych listinách. Modul na rozdiel od primárneho modulu 3.4.1 porovnáva prítomnosť IP adresy na čiernych listinách uložených v Redis databáze. Na začiatku modul čierne listiny načíta. Hlavná funkcia zisťuje výskyt IP adresy na čiernej listine a vracia procesnému jadru názvy listín kde sa IP adresa objavuje. Modul má zaregistrované spätné volania na tieto udalosti:

- príchod novej entity,
- aktualizácia raz za deň.

Tento modul nepotrebuje nový atribút, stačí mu rovnaký, aký používa primárny modul 3.4.1.

3.5.5 Modul Geolocation

Modul určuje približnú geografickú pozíciu IP adresy na základe MaxMind GeoLite2 databázy[2]. Ak je nájdená zhoda v databáze sú vrátené informácie o zemi, meste a časovom pásme.

- príchod novej entity,
- vyžiadanie aktualizácie geografickej polohy (v novej verzii nepodporované).

Atribút pre geografickú polohu bude typu *plain*, pretože je to odvodený atribút a v budúcnosti sa nebude meniť. Dátový typ atribútu je textový reťazec obsahujúci krajinu, mesto a časové pásmo oddelené čiarkami.

3.5.6 Modul Hostname class

Modul klasifikuje IP adresy na základe ich doménového mena. Načíta z konfigurácie databázu známych domén a regulárnych výrazov. Podľa nej prichádzajúcej IP adrese pridelí štítky doménového mena (hostname tags) a aktualizovaný zoznam štítkov pošle naspäť procesnému jadru. Modul má zaregistrované spätné volania na tieto udalosti:

- príchod nového atribútu doménového mena,
- vyžiadanie aktualizácie štítkov doménového mena (v novej verzii nepodporované).

V súčasnej verzii systému NERD sa *hostname tags* používajú na vyhodnocovanie štítkov aj s mierou spoľahlivosti, takže ako samostatné štítky sa vo výstupe neobjavujú. Avšak tento typ štítkov s dôverou sa dobre neujal a preto som po diskusii s vedúcim rozhodol, že v novej verzii NERDu bude tento modul vytvárať rovno štítky bez miery spoľahlivosti. Otázkou je ako štítky reprezentovať v databáze. Keďže sa štítky odvodzujú z iných atribútov budú typu *plain*. Tým pádom nie je možné použiť všeobecný atribút *tags*, pretože typ *plain* nepodporuje viachodnotovosť. Dátový typ pole štítkov tiež nedáva zmysel z dôvodu, že štítky môže určovať viacero modulov (štítky z rôznych modulov by sa navzájom "prepisovali"). Preto najlepší variant je aby každý štítok bol osobitným atribútom. Tento prístup je síce zložitejší z hľadiska konfigurácie, ale je najsprávnejší, preto som ho zvolil.

3.5.7 Modul Shodan

Modul získava informácie o IP adresách z databázy Shodan[3] pomocou *InternetDB API*. API vráti porty, značky (tags) a CPE (Customer-premises equipment) IP adresy, ak sú k dispozícii, alebo None, ak adresa nebola nájdená. Podľa toho modul určí čo sa má vykonať. Ak sa nič pre adresu nenašlo a v NERD databáze sa informácie zo Shodanu nachádzajú, modul nariadi ich vymazanie. Ak API vráti informácie o IP adrese modul ich vráti jadru na zapísanie do databáze. Požiadavky na API Shodanu sa posielajú len pre IP adresy, ktoré sú dlhodobo aktívne aby sa tak zmenšil počet požiadaviek a nenaplnilo obmedzenie rýchlosti API (rate limit). Modul má zaregistrované spätné volania na tieto udalosti:

- príchod novej entity (v novej verzii nepodporované, aby sa obmedzil počet požiadaviek),

[2] https://www.maxmind.com/en/geoip-databases
[3] https://www.shodan.io

- vyžiadanie aktualizácie dát zo Shodanu (v novej verzii nepodporované),
- aktualizácia raz za týždeň.

Shodan poskytuje pre NERD 3 typy dát. Nadefinoval som preto tri rôzne atribúty typu *observation*. Atribúty sú dátového typu reťazec a dovoľujú viachodnotovosť (*multi-value*), takže napríklad každý port bude obsahovať samostatný data-point. Data-pointy obsahujú len jednu časovú značku a to je čas stiahnutia. Platnosť po je nastavená na 6 dní. Tak je zabezpečené, že sa nachádzajú v systéme vždy len tie najaktuálnejšie informácie. Keď sa o týždeň budú informácie aktualizovať a v Shodan databáze už nebudú, platnosť data-pointu skončí a tým sa zabezpečí funkcionalita "vymazávania" ako to je v súčasnej verzii systému NERD.

3.5.8 Moduly Reserved IP a Tags

Modul Reserved IP označuje všetky adresy ktoré patria do rezervovaného bloku IP adries podľa [3]. Modul Tags je modul na prideľovanie štítkov pre jednotlivé entity. Značky a ich mieru spoľahlivosti prideľuje na základe jednotlivých atribútov danej entity.

Modul Tags je v súčasnej podobe veľmi komplikovaný modul. Spôsob odvodzovania štítkov podľa iného typu štítkov, napríklad z modulu Hostname class s pridaním miery dôvery sa v NERDe neosvedčilo. Je to zbytočne komplikovaný postup a výsledok nie je veľmi odlišný. Navrhol som teda nový modul, ktorý nahradí moduly Reserved IP a Tags. Pričom iné moduly, ktoré vytvárajú štítky ich budú vytvárať priamo, takže to už nebude robiť tento modul ako v minulosti. Nový modul si načíta definované pravidlá na štítkovanie a bude označovať jednotivé entity na základe hodnôt ich atribútov. Týmto spôsobom vznikne všeobecný štítkovací modul, ktorý bude v budúcnosti možné jednoducho rozšíriť pridaním pravidiel a funkcií spätného volania.

3.5.9 Zvyšné sekundárne moduly NERD

Ako som spomínal na začiatku sekcie v súčasnej podobe NERD používa okolo 20 sekundárnych modulov. Nasleduje zoznam takých, ktoré sa v novej verzii systému už používať nebudú, teda sa nebudú ani reimplementovať. Ku každému pridávam aj dôvod vyradenia z novej verzie:

- **CAIDA AS class** - Určuje typ AS podľa súboru údajov CAIDA - od roku 2021 však tieto údaje nie sú k dispozícii.
- **Cleaner** - Jeho funkcionalita nahradená priamo platformou DP³.
- **EML ASN rank** - Historický modul, ktorý zisťuje ASN Rank z The Email Laundry API, tam však už CESNET nemá prístup.
- **Event counter** - Vytvára atribúty s počtom udalostí Warden za posledný deň, 7 dní a 30 dní. V novej verzii už nebude potrebné, aby modul takto explicitne počítal.
- **Event type counter** - Určuje najčastejší typ udalosti pre danú adresu. Bude nahradený novým systémom klasifikácie hrozieb, ktorý je vo vývoji.
- **FMP** - Vypočíta Future Misbehavior Score (skóre budúceho nesprávneho správania) IP adresy - pravdepodobnosť útokov v nasledujúcich 24 hodinách odvodenú z predchádzajúcich údajov a strojového učenia. Táto implementácia nefunguje veľmi dobre a výsledky sa nijako nepoužívajú.

- **Reputation** - Vypočítava reputáciu jednotlivých entít. Bude nahradený novým systémom klasifikácie hrozieb.
- **Tags** - Určuje značky pre jednotlivé entity. V súčasnej podobe už nebude pre novú verziu potrebný.
- **TTL updater** - Jeho funkcionalita nahradená priamo platformou DP³.
- **Update planner** - Jeho funkcionalita nahradená priamo platformou DP³.
- **Who is** - Modul zisťujúci ASN, BGP prefix, IP block a organizáciu. V novej navrhovanej verzii zatiaľ nepodporované entity a atribúty.
- **BGP rank** - Modul získava BGP ohodnotenie entít typu ASN pomocou *BGP Ranking API*[4]. V novej navrhovanej verzii nepodporovaný atribút.

3.5.10 Potrebné rozšírenie DP³

Pre rovnakú funkcionalitu sekundárnych modulov je potrebné niektoré atribúty periodicky obnovovať, napríklad raz za týždeň. V súčasnej dobe DP³ neposkytuje jednoduchý spôsob ako atribúty periodicky obnovovať, teda volať obslužné funkcie v jednotlivých moduloch. Jeden zo spôsobov ako podobné správanie simulovať, je použiť korelačné spätné volanie, konkrétne počas vytvárania snímky. Problémom je, že snímky sa vytvárajú niekoľkokrát za deň a atribúty by sa mali obnovovať len raz za deň alebo raz za týždeň. Na prvý pohľad je to iba výkonnostný problém, pretože niekoľkokrát za deň obnovovať informácie pre stovky tisíc entít je dosť výkonnostne náročné. Väčším problémom je, ale obmedzenie rýchlosti (rate limit) niektorých sekundárnych dátových zdrojov. To by malo za následok, že by sa obnovil len zlomok entít. Jedným z navrhovaných riešení je použiť tieto korelačné spätné volania a kontrolovať poslednú zmenu konkrétneho atribútu. Ak zmena nastala za posledný deň alebo týždeň a podľa toho vykonať alebo nevykonať aktualizáciu atribútu. Druhou možnosťou je rozšíriť funkcionalitu platformy. DP³ by tak poskytovalo možnosť zaregistrovať periodické spätné volanie, ktoré by postupne volalo obslužné funkcie modulov nad všetkými entitami rozložené v čase. Prvá možnosť má výhodu v tom, že nie je potrebné rozširovať platformu. Na druhú stranu by sa naraz vykonalo veľké množstvo aktualizácií a vznikal by nárazový dopyt po výpočetných zdrojoch. Pričom by sa problém obmedzenia rýchlostí sekundárnych dátových zdrojov vyriešil len z časti. Preto som zvolil za lepšiu druhú možnosť napriek tomu, že vyžaduje rozšírenie platformy DP³.

[4] https://bgpranking-ng.circl.lu/

Kapitola 4

Implementácia

Táto kapitola hovorí o implementácii jednotlivých častí systému ako boli navrhnuté v kapitole 3. V sekcii 4.1 hovorím o implementácii primárnych modulov vo všobecnosti a potom aj o konkrétnych moduloch. Sekcia 4.2 sa venuje implementácii sekundárnych modulov.

NERD v súčasnej podobe beží na staršej verzii Pythonu *3.6*. DP^3 podporuje avšak verziu *3.9* a vyššiu. Preto bola na implementáciu zvolená verzia Python *3.9*. To umožní zlepšiť kvalitu a čitateľnosť kódu jednotlivých častí systému, vďaka novej funkcionalite. Na konfiguráciu entít, modulov a platformy bol použitý jazyk YAML.

4.1 Primárne moduly

Všetky primárne moduly majú podobnú štruktúru. Na začiatku si načítajú potrebnú konfiguráciu. Ďalej obsahujú sadu funkcií, ktorá získava dáta z primárneho dátového zdroja. Ďalšia sada funkcií slúži na spracovanie prijatých dát a na extrakciu potrebných informácií. Poslednou časťou sú funkcie, ktoré slúžia na odoslanie získaných informácií do systému. Proces prijímania dát a jeho spracovávania sa pre novú verziu systému meniť nemusí, keďže priamo s platformou, ktorej modul dáta posiela nijako neinteraguje až na pár výnimiek, ktoré v tejto sekcii ďalej popisujem. Ako som už spomínal v kapitole 3, potrebné bude zmeniť načítanie konfigurácie a hlavne spôsob vytvárania správ - data-pointov a ich následné posielanie do systému.

4.1.1 Konfigurácia

Konfigurácia primárnych modulov v pôvodnej verzii fungovala 2 spôsobmi. Načítaním všeobecnej konfigurácie pre celý systém alebo pomocou argumentov príkazového riadku. Viac o pôvodnej konfigurácii píšem v sekcii 3.4. Spôsob konfigurácie som v novej verzii sčasti zachoval. Jednoduchšie moduly používajú iba argumenty príkazového riadku. Zložitejším modulom je potom pomocou argumentu predaný ich konfiguračný súbor, ktorý obsahuje viaceré informácie ako napríklad filtračné pravidlá alebo zoznamy dátových zdrojov. Pre zachovanie konzistencie medzi modulmi som konfiguráciu rovnakých informácií ako napríklad URL adresu NERD serveru implementoval ako argument príkazového riadku napriek tomu, že modul používa aj konfiguračný súbor.

4.1.2 Posielanie data-pointov

V ranných fázach vývoja boli data-pointy v moduloch posielané na HTTP API DP³ samostatne, každý data-point bol odoslaný zvlášť ako samostatná HTTP POST požiadavka. Tento model sa používal v pôvodnej verzii systému keď hneď po získaní informácií sa vložili do hlavného frontu úloh. Posielanie data-pointov na API samostatne sa však ukázalo ako mimoriadne neefektívne. V niektorých prípadoch modul chce poslať desiatky tisíc data-pointov hneď za sebou, čo je mimoriadna záťaž.

Keďže všetky primárne moduly potrebujú posielať data-pointy na API, vytvoril som knižnicu funkcií *utils.py*, ktorá zabezpečuje posielanie data-pointu a v budúcnosti, pri rozšíreniach môže poskytovať aj ďalšie spoločné funkcie pre moduly. Funkcia *send_datapoints* prijíma pole data-pointov, ktoré následne odosiela pomocou HTTP požiadavky POST. Dáta, ktoré prijíma API majú túto formu: Je to pole objektov vo formáte JSON, inak povedané pole data-pointov. Keďže pole data-pointov, ktoré príjme funkcia môže mať v niektorých prípadoch aj desaťtisíce položiek, modul posiela data-pointy na API v dávkach. Veľkosť jednej dávky určuje modul, ktorý chce data-pointy odoslať. V základnom prípade je veľkosť dávky 100.

4.1.3 Moduly

Modul Blacklists

Modul sťahuje čierne listiny v rôznych časoch počas dňa. Tieto časy sú naplánované podľa konfigurácie čiernych listín a sú nastavené na čas kedy sa čierna listina aktualizuje. Týmto spôsobom sa aktualizujú hodnoty z každej čiernej listiny raz za deň. Keď sa čierna listina stiahne a vytvorí sa z nej zoznam IP adries, ktoré sa na nej nachádzajú, pre každú IP adresu sa vytvorí data-point. Po ich vytvorení pre všetky adresy na čiernej listine sa všetky naraz pošlú na API. Modul dobre využíva optimalizáciu posielania popísanú v podsekcii 4.1.2.

Modul Warden receiver

Modul spracováva prichádzajúce udalosti z Wardenu. Udalosti sú čítane z priečinka, kde ich uložil skript *warden_filer*, pomocou funkcie *read_dir*, ktorá stále sleduje obsah priečinka a pri príchode novej udalosti ju prečíta a vymaže. Užívateľ si môže nakonfigurovať filtre, ktoré odfiltrujú udalosti nepotrebné pre NERD. Filtre sú v module uložené ako objekt triedy *Warden filter*. Ak udalosť prejde filtrom je pre ňu vytvorený data-point, ktorý je následne odoslaný. V tomto prípade sa posielanie data-pointov nedalo optimalizovať v zmysle posielania viacerých data-pointov v jednej HTTP požiadavke, pretože modul beží nepretržite a udalosti z Wardenu prichádzajú rôzne. Nevieme kedy ďalšia udalosť príde preto ukladanie data-pointov nemá z tohto pohľadu zmysel. Zároveň ich v jeden čas nikdy nebude toľko aby API DP³ zahltili.

Modul MISP receiver

MISP receiver je z pohľadu reimplementácie asi najzložitejší modul. Modul prijíma viacero druhov správ. Podľa typu správy je potom zavolaná obslužná funkcia. Funkcie potom vytvoria data-pointy, ktoré následne pošlú. Modul reaguje na tieto typy správ:

- **Uverejnenie MISP udalosti.** Je potrebné pridať novú udalosť. Z prijatej správy sa extrahujú dôležité informácie a vytvoria sa data-pointy jednotlivých atribútov.

- **Vymazanie MISP udalosti.** Na vykonanie vymazania sa vytvoria data-pointy s prázdnou hodnotou. To znamená, že sa záznam o MISP udalosti v databáze prepíše na prázdne hodnoty. V súčasnom dátovom modeli DP³ sa nedá vymazať konkrétny data-point a keďže atribút entity IP adresa je viachodnotový, nedá sa tento záznam z databáze odstrániť. To však nie je problém, keďže atribút bude odkazovať len na entitu s prázdnymi hodnotami atribútov.
- **Pridanie atribútu.** Funkcia overí validitu atribútu následne z MISP stiahne udalosť do ktorej atribút patrí, vytvoria sa data-pointy a odošlú.
- **Úprava atribútu.** Rovnako ako v predošlom prípade sa overí validita stiahne prislúchajúca udalosť, vytvoria sa data-pointy a odošlú.
- **Vymazanie atribútu.** Po overení prebieha toto vymazanie rovnako ako vymazanie udalosti.
- **Oznámenie o pozorovaní.** Modul najprv overí či sa udalosť, ku ktorej patrí pozorovanie už v databáze nachádza. Používa k tomu API endpoint, ktorý DP³ poskytuje (opísané v podsekcii 2.2.4). Ak sa už v databáze nachádza, odošle sa data-point len s atribútom *sightings*. Využíva sa tu, že jednotlivé atribúty udalosti sú v databáze uchovávané ako samostatné atribúty. Ak sa však udalosť ktorej patrí pozorovanie v databáze ešte nenachádza, odošlú sa informácie o udalosti spolu pozorovaním.

Modul OTX receiver

Modul po inicializácii stiahne z OTX API všetky zmenené a pridané pulzy, od posledného stiahnutia. Pri spracovávaní jednotlivých pulzov je najskôr vytvorený data-point pulzu (typ entity *otx_pulse*)a následne data-pointy pre všetky jeho IP adresy, ktoré budú podľa návrhu v podsekcii 3.4.4 na entitu pulzu odkazovať. Po spracovaní všetkých pulzov sa data-pointy odošlú na API DP³.

Modul DShield

Modul pomocou knižnice *urllib.request*[1] stiahne dáta zo stránky inštitútu SANS[2]. Záznamy potom riadok po riadku spracováva a ak IP adresa spĺňa podmienky zaradenia do databáze je vytvorený data-point. Keď sa vytvoria všetky data-pointy sú odoslané.

4.2 Sekundárne moduly

Sekundárne moduly sú tvorené hlavnou triedou modulu, ktorú procesné jadro inštancuje. Hlavná trieda je odvodená z abstraktnej triedy *BaseModule*. Aby moduly správne fungovali a procesné jadro ich dokázalo spustiť musia dediť po tejto triede. Po spustení systému jadro inštancuje jednotlivé triedy modulov a načíta ich konfiguráciu. Ako som už spomínal v návrhu, každý modul má vlastný konfiguračný súbor. Konfiguračné súbory som vytvoril pre všetky moduly, ktoré to potrebovali. Načítanie konfigurácie v pôvodnej verzii systému fungovalo pomocou knižnice *common*, ktorú obsahoval NERD. Keďže táto knižnica v novej verzii nebude dostupná (potrebná), moduly používajú na načítanie knižnicu *dp3.common*, ponúkanú samotnou platformou.

[1] https://docs.python.org/3/library/urllib.request.html
[2] https://www.sans.edu

Moduly používajú na prácu s dátami funkcie spätných volaní (callbacks functions). Zoznam spätných volaní je v sekcii 2.2.7. Moduly však majú zaregistrovaných viacero spätných volaní, inak povedané je potrebné aby vykonali svoju úlohu pri rozličných udalostiach. To spôsobuje problém, pretože požadovaný podpis funkcie (required signature) môže byť rôzny, pretože podľa typu spätného volania funkcia dostáva iné atribúty, ktoré by mohla potrebovať. Vyriešil som tento problém tak, že funkcia spätného volania nie je zároveň procesnou funkciou, ale iba akýmsi rozhraním, ktoré následne volá procesnú funkciu s atribútmi, ktoré daná funkcia potrebuje. Nasleduje zoznam sekundárnych modulov a spätných volaní, ktoré majú zaregistrované podľa návrhu v sekcii 3.5:

- **IP to DNS hostname** nahrádza pôvodný modul DNS. Tento modul som nereimplementoval z pôvodného systému NERD. Je prevzatý a upravený pre NERD z interného projektu ADiCT združenia CESNET. Keďže tento modul bol len upraveným pôvodným modulom NERDu, použitý pre ADiCT, ktorý pracuje nad platformou DP[3]. Nemalo preto zmysel prepisovať znovu to isté. Zoznam zaregistrovaných volaní:
  - po vytvorení novej entity typu *ip* (*on_entity_creation*),
  - periodická aktualizácia každých 7 dní (*periodic_update*).
- **DNS blacklist resolver** sa dotazuje na IP adresy do konfigurovaných čiernych listín pomocou DNSBL techniky. Využíva na to knižnicu *pycares*[3]. Zoznam zaregistrovaných volaní:
  - po vytvorení novej entity typu *ip* (*on_entity_creation*),
  - periodická aktualizácia každých 7 dní (*periodic_update*).
- **Passive DNS resolver** načítava čierne listiny z Redisu sa ukladá ich ako objekty triedy *Blacklist* (viac o nej píšem neskôr v tejto sekcii) v nich hľadá domény, ktoré získal z API pasívneho DNS združenia CESNET pomocou HTTP knižnice requests[4]. Zoznam zaregistrovaných volaní:
  - po vytvorení novej entity typu *ip* (*on_entity_creation*),
  - periodická aktualizácia raz za deň (*periodic_update*).
- **Redis blacklist** rovnako ako *Passive DNS resolver* využíva objekty triedy *Blacklist* na ukladanie čiernych listín z Redis databáze. Zoznam zaregistrovaných volaní:
  - po vytvorení novej entity typu *ip* (*on_entity_creation*),
  - periodická aktualizácia raz za deň (*periodic_update*).
- **Geolocation** rovnako ako IP to DNS hostname je tento modul mierne upravenou verziou modulu z projektu ADiCT. Kde bol rovnako prebraný a upravený z NERDu. Zoznam zaregistrovaných volaní:
  - po vytvorení novej entity typu *ip* (*on_entity_creation*).
- **Hostname class** modul označuje entity podľa ich doménového mena. Modul vytvorí množinu štítkov, ktorú následne odošle. Zoznam zaregistrovaných volaní:

[3] https://pypi.org/project/pycares/
[4] https://pypi.org/project/requests/

  - po prijatí nového atribútu typu *hostname*(*on_new_attr_hook*).

- **Shodan** modul sa dotazuje na Shodan API pomocou HTTP knižnice requests. V prípade chyby 429 - *Too Many Requests* ukončí spracovanie entity. Zoznam zaregistrovaných volaní:

  - periodická aktualizácia každých 7 dní (*periodic_update*).

- **Tags** na rozdiel od ostatných modulov nie je upraveným pôvodným NERD modulom, ale úplne novým. Ako som písal v návrhu 3.5.8 modul načítava súbor pravidiel, podľa ktorého entitám priraďuje štítky. Načítanie a vyhodnocovanie pravidiel realizuje pomocou knižnice *LabelRuleLoader*. Knižnica zjednodušuje proces definovania a správy pravidiel pre označovanie zariadení. Slúži ako analyzátor jednoduchého konfiguračného jazyka. Knižnica je vyvinutá interne v združení CESNET vývojármi projektov DP³ a ADiCT a nie je dostupná verejnosti vo verejnom repozitári. Modul používa ako klasifikačné pravidlá regulárne výrazy rezervovaných IP adries prebraté z pôvodného modulu *reserved_ip*. Ďalšou funkcionalitou je prideľovanie štítku *whitelist* (biela listina) entitám, ktoré majú pridelený štítok *reserved_ip* a *research_scanner*. Zoznam zaregistrovaných volaní:

  - po vytvorení novej entity typu *ip* (*on_entity_creation*),
  - po prijatí nového atribútu typu *reserved_ip* (*on_new_attr_hook*),
  - po prijatí nového atribútu typu *research_scanner*(*on_new_attr_hook*).

Moduly po tom čo získajú informácie o entite, vytvoria potrebný počet data-pointov. Tieto data-pointy sa však neposielajú na API platformy. Je potrebné vytvoriť objekt triedy *DatapointTask*. Trieda slúži na reprezentáciu úlohy, ktorá sa pošle do frontu úloh (task queue) a následne spracuje. Udalosť musí obsahovať typ entity, ID entity a pole datapointov. Po úspešnom vytvorení sa pošle procesnému jadru ako návratová hodnota funkcie spätného volania. Posiela sa pole objektov *DatapointTask*. Aktuálne moduly však pracujú vždy len s entitou jedného typu, takže posielajú iba pole s jedným objektom.

Načítanie čiernych listín

Moduly Passive DNS resolver a Redis blacklist používajú na klasifikáciu IP adries čierne listiny, ktoré sú uložené v Redis databáze. V pôvodnej verzii systému mali obidva moduly samostatnú triedu *Blacklist* na načítanie čiernych listín z Redisu. Triedy boli však s malými rozdielmi úplne totožné. Preto som upravil triedu na použitie pre oba moduly a presunul ju aj s ďalšou spoločnou funkciou do samostatného modulu *redis_utils*. Odstránila sa tak duplicita kódu a zvýšila znovu použiteľnosť.

Kapitola 5

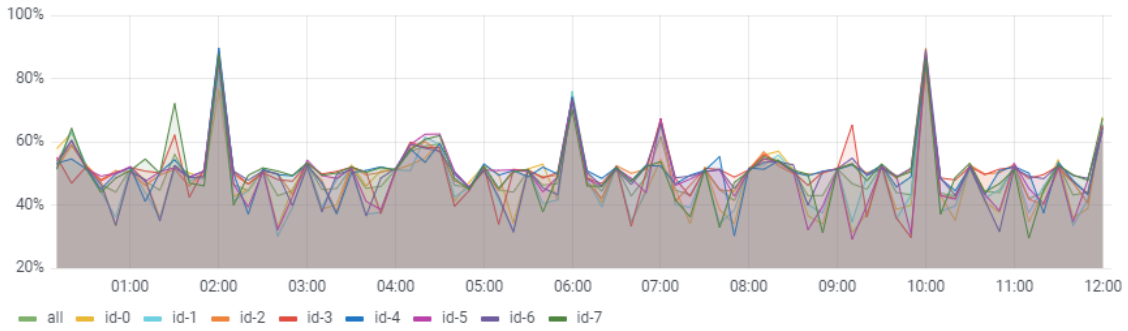
Testovanie

V tejto kapitole postupne opíšem testovanie novej verzie systému NERD a overenie jej správneho fungovania. Na začiatku reimplementácie, pri implementovaní jednotlivých modulov, boli moduly, ak to bolo možné, testované na offline dátach. Zachytil som si do súboru dáta, väčšinou jednotky dát, ktoré by testované moduly normálne sťahovali. Počas prvej fáze testov moduly tieto zachytené dáta čítali a vytvárali výstup. Takýmto spôsobom bola v ranných fázach vývoja overená správna forma tvorby data-pointov a zároveň bolo overené, že sa dáta v module nikde nestrácajú a všetky prejdú až na výstup. Výstupné dáta upravených modulov sa následne porovnávali z výstupnými dátami pôvodných modulov. Tento postup sa dal efektívne uplatniť, ale len pri primárnych moduloch. Sekundárne moduly reagujú len na dáta, ktoré sa už v systéme nachádzajú. To znamená, že pri testovaní je potrebné nahrávať nové dáta a overovať výstup, ktorý moduly vytvárajú. Ďalším problémom je, že väčšina modulov má zaregistrované spätné volanie na príchod novej entity. V praxi to znamená, že je potrebné pri testovaní poslať do systému vždy novú entitu, ktorú dokáže modul klasifikovať (vytvoriť pre ňu nejaký výstup) inak sa spätné volanie nespustí. Ako zdroj takýchto IP adries sa používala v súčasnosti bežiace webové rozhranie služby NERD[1]. Validita výstupu jednotlivých modulov sa kontrolovala dvomi spôsobmi na dvoch miestach. Prvý spôsob bol jednoduchý výpis pomocou bežnej Python logovacej knižnice[2]. Druhý spôsob bol priamym nazeraním do databáze testovacieho systému pomocou mongosh[3]. Tento spôsob bol síce zložitejší ako obyčajný výpis, ale overoval aj predpokladaný prechod a spracovanie data-pointu platformou DP³. Overovanie správnosti prebiehalo dvomi spôsobmi. Prvý bol kontrolovaním za použitia pôvodných modulov a porovnávaním s ich výstupom. Druhým spôsobom bolo porovnávanie priamo s dátami zobrazenými na webovom rozhraní NERD. Takýmto spôsobom bola overená ekvivalentnosť modulov, pretože sa všetky výstupy zhodovali. Zhoda s webovou službou NERD nebola vždy úplná. To bolo spôsobené, že sa tam nachádzali novšie informácie, ktoré staršie zachytené dáta ešte neobsahovali.

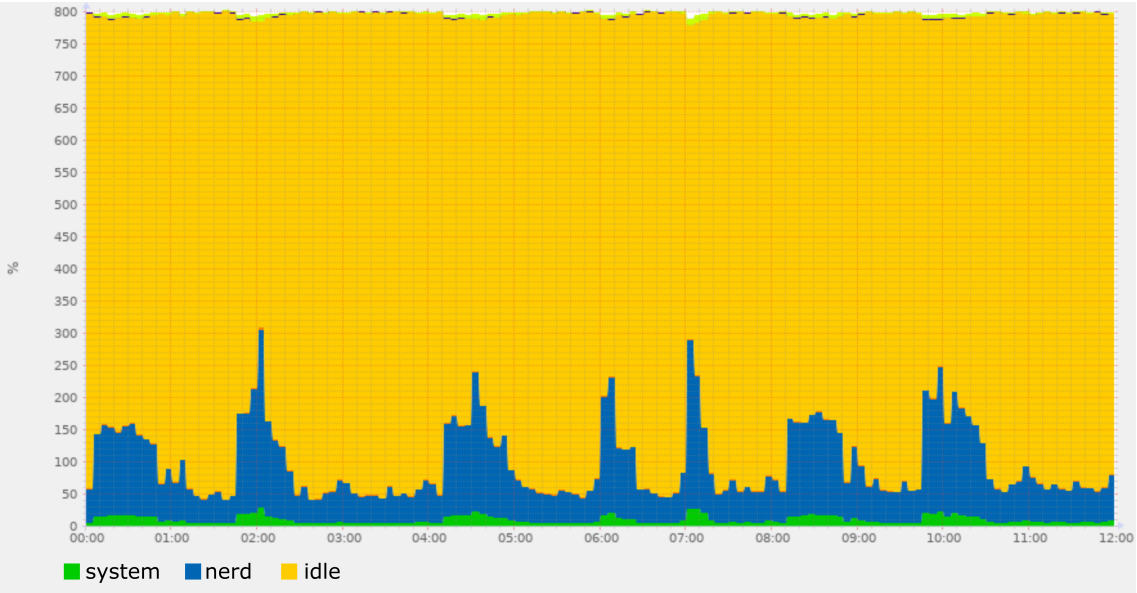
Druhou fázou testovania bolo testovanie za použitia online dát. Moduly boli spustené ako budú spustené v reálnom nasadení a sťahovali aktuálne dáta. V tomto prípade už nebolo možné overovať validitu dát, pretože moduly spracovávali desiatky až stovky tisíc IP adries. Cieľom testovania bolo nájsť chyby, ktoré sa objavujú len zriedka. Keďže sekundárne moduly potrebujú na spustenie nové dáta ako zdroj dát boli použité primárne moduly. Bolo objavených len pár problémov. Napríklad pri sťahovaní čiernych listín. Ak sa tam nachádzala

[1] https://nerd.cesnet.cz/nerd/ips/
[2] https://docs.python.org/3/library/logging.html
[3] https://www.mongodb.com/docs/mongodb-shell/

neplatná IP adresa tak modul spadol. Keďže tieto testy boli celkovo úspešné, prešiel som na ďalšiu fázu testovania.



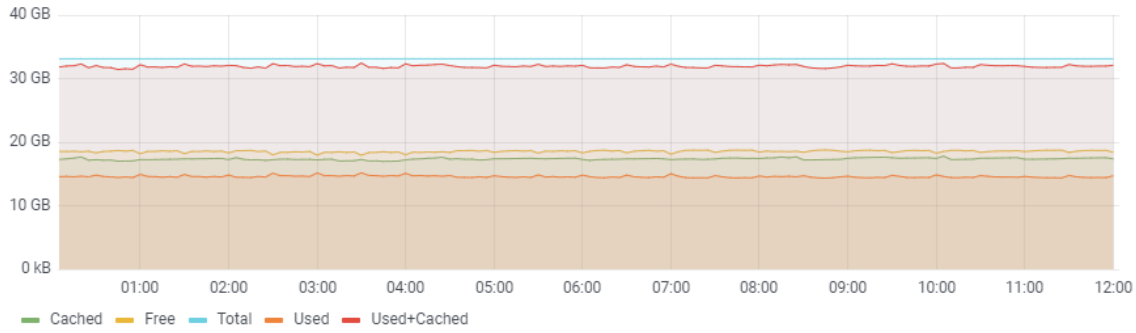
Obr. 5.1: Graf využitia CPU novej verzii NERD. V legende id-1 až 8 označuje 8 jadier procesora.



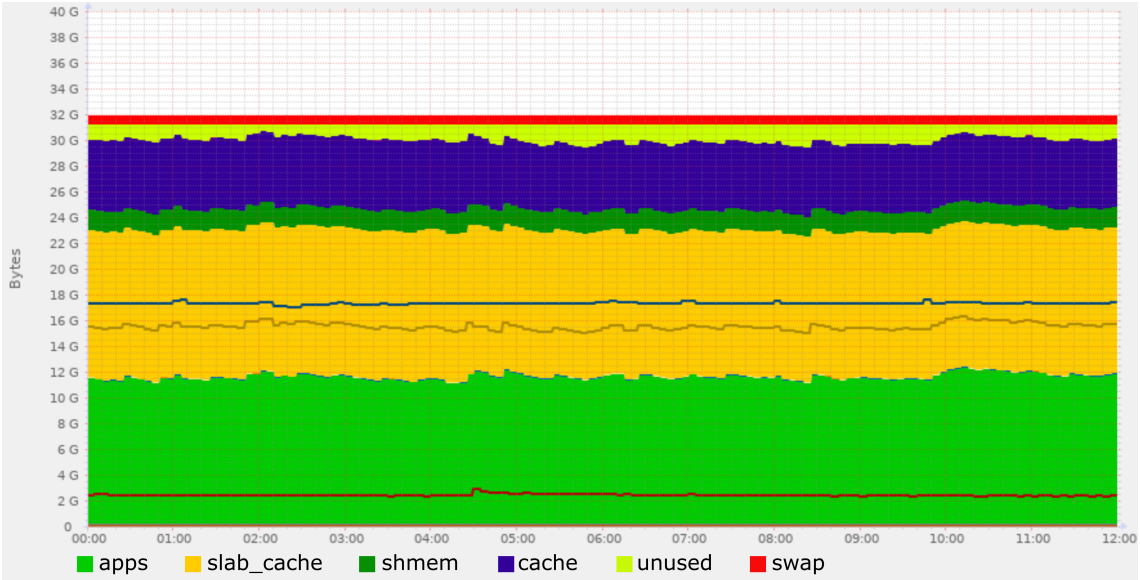
Obr. 5.2: Graf využitia CPU pôvodnej verzii NERD. Server má 8 jadier, preto má osa y 800%.

Treťou fázou bolo spustenie systému ako celku na dlhý čas. Všetky komponenty systému boli spustené ako budú v budúcnosti nasadené. Toto dlhodobé testovanie v čase písania tejto práce stále beží bez problémov. Aplikácia je nasadená na testovacom serveri, ktorý má rovnaké parametre ako aktuálny produkčný server NERD. Paralelne na serveri beží osem pracovných procesov DP³. Všetky procesy a komponenty DP³ spolu s implementovanými primárnymi modulmi a ďalšími potrebnými časťami NERDu riadi supervízor *nerd*. Štatistiky o behu aplikácie, zaťaženie procesoru a využitie pamäte zaznamenáva nástroj *icinga-monitoring*, interne vyvíjaný združením CESNET. Na zobrazovanie grafov sa používa Grafana a DP³ zásuvný modul[4]. Produkčný server používa starší systém Munin[5]. Ten

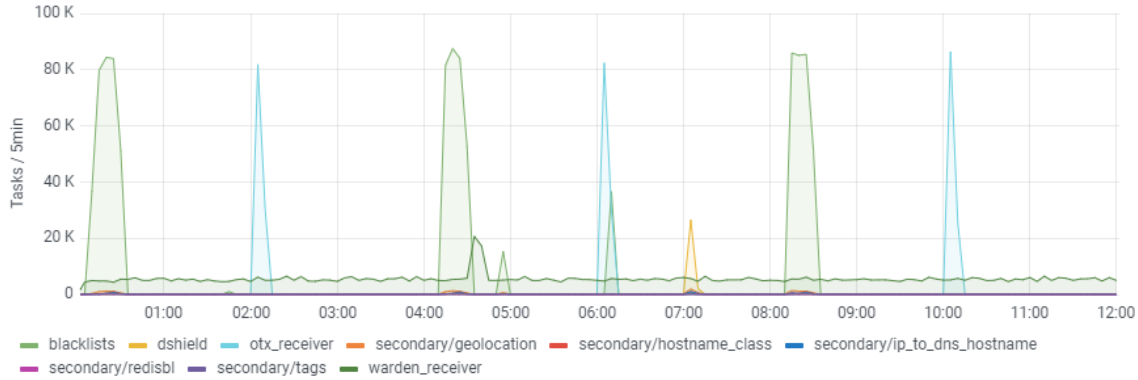
[4] https://grafana.com/grafana/plugins/cesnet-dp3-app/
[5] https://munin-monitoring.org



Obr. 5.3: Graf využitia pamäte v novej verzii NERD.



Obr. 5.4: Graf využitia pamäte v pôvodnej verzii NERD.



Obr. 5.5: Počet spracovaných úloh podľa zdroja v novej verzii NERD.

Obr. 5.6: Počet spracovaných úloh podľa zdroja v pôvodnej verzii NERD.

už na novších distribúciách Linuxu nie je podporovaný. Na grafe 5.1 vidíme zaťaženie procesoru v novej implementácii. Väčšie vrcholy grafu sú spôsobené príchodom veľkého objemu úloh, ktoré vznikli napríklad stiahnutím čiernej listiny. Menšie sú spôsobené periodickým vytváraním snímok. Na grafe 5.2 vidíme zaťaženie na produkčnom nasadení NERDu. Grafy sú z rovnakého časového úseku. Nová implementácia v porovnaní s pôvodnou potrebuje viac výkonu. To nie je problém, keďže zlepšenie efektivity nebolo cieľom práce. DP^3 je všeobecnejšia platforma a poskytuje viac možností (najmä pre budúce rozšírenie), väčšia spotreba CPU je preto pochopiteľná. Ako je možné vidieť na grafoch 5.3 a 5.4 využitie pamäte majú obidve verzie veľmi podobné. Grafy 5.5 a 5.6 zobrazujú počet spracovaných úloh. Vrcholy (maximá), ktoré vidno na grafoch sú časy sťahovania veľkých primárnych dátových zdrojov, väčšinou čiernych listín. Tieto grafy sú takmer identické, keďže sa čas sťahovania a objem prichádzajúcich dát je rovnaký pre obe verzie.

Kapitola 6

Záver

Cieľom práce bolo naštudovať fungovanie systému NERD (Network Entity Reputation Database) a platformy DP^3 (Dynamic Profile Processing Platform) a následne navrhnúť a implementovať funkcionalitu existujúceho systému NERD v podobe aplikácie nad platformou DP^3. V práci bol zhodnotený a zdokumentovaný aktuálny stav týchto systémov.

V návrhu boli tieto systémy porovnané a boli identifikované zmeny, ktoré boli potrebné urobiť pre prevod systému NERD na novú platformu. Boli nájdené časti systému, ktoré je potrebné upraviť pre novú verziu a zároveň aj časti, ktoré už naďalej nebudú potrebné. Bol navrhnutý spôsob reimplementácie jednotlivých častí systému so zameraním na návrh dátového modelu nového systému aby svojim správaním korešpondoval so súčasnou verziou systému NERD.

V práci sa podarilo vytvoriť nový dátový model a reimplementovať moduly primárnych dátových zdrojov, ako aj sekundárne moduly pre získavanie dát zo sekundárnych dátových zdrojov a prácu s dátami v systéme. Zároveň bola pripravená aj konfigurácia platformy DP^3 pre produkčné nasadenie.

Výslednú implementáciu, vypracovanú v tejto práci, sa úspešne podarilo nasadiť na testovacom serveri. Novovytvorený systém je pripravený na produkčné nasadenie, ale ešte pred tým, bude potrebné adaptovať nové webové užívateľské rozhranie, ktoré vzniklo ako diplomová práca Ing. Oltmanovej [5]. To však používa ešte aplikačné programové rozhranie starej verzie systému, ktoré bude potrebné upraviť. Keďže jedna z motivácii pre túto prácu bola možnosť jednoduchšieho vývoja systému, v budúcnosti bude možné systém naďalej rozširovať. Jedno z rozšírení, ktoré sa vyvíja, je napríklad nový systém kategorizácie hrozieb. Ďalšie rozšírenia by mohli byť pridanie ďalších dátových zdrojov a entít.

Literatúra

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

[2] Bartoš, V. *NERD - Network Entity Reputation Database* [online]. CESNET, 2024 [cit. 2024-04-15]. Dostupné z: https://github.com/CESNET/NERD/wiki.

[3] Cotton, M., Vegoda, L., Bonica, R. a Haberman, B. *Special-Purpose IP Address Registries* [RFC 6890]. RFC Editor, apríl 2013. DOI: 10.17487/RFC6890. Dostupné z: https://www.rfc-editor.org/info/rfc6890.

[4] Mockapetris, P. *Domain names - implementation and specification* [RFC 1035]. RFC Editor, november 1987. DOI: 10.17487/RFC1035. Dostupné z: https://www.rfc-editor.org/info/rfc1035.

[5] Oltmanová, K. *Moderní webové rozhraní systému NERD [online]*. 2023 [cit. 2024-04-15]. Diplomová práce. Vysoké učení technické v Brně. Dostupné z: https://theses.cz/id/tsf5c6/.

[6] Sedláček, O. a Benko, D. *Dynamic Profile Processing Platform (DP³)* [online]. CESNET, 2024 [cit. 2024-04-16]. Dostupné z: https://cesnet.github.io/dp3/.

Príloha A

Obsah priloženého pamäťového média

```
│
├── text_source
├── nerd
│   ├── dp3_config
│   │   ├── db_config
│   │   └── modules
│   ├── primary_modules
│   ├── secondary_modules
│   ├── install
│   └── scripts
```

Obr. A.1: Adresárová štruktúra pamäťového média.

Obrázok A.1 zobrazuje adresárovú štruktúru priloženého média. Priečinok *text_source* obsahuje zdrojový kód pre vytvorenie tohto textu. V priečinku *nerd* sú všetky zdrojové kódy a konfiguračné súbory pre novú verziu systému NERD. Priečinok *dp3_config* obsahuje konfiguráciu platformy DP^3. Priečinok *db_entities* obsahuje konfiguráciu entít a priečinok *modules* konfiguráciu modulov. Priečinok *primary_modules* obsahuje primárne moduly a priečinok *secondary_modules* obsahuje sekundárne moduly. Priečinok *install* obsahuje konfiguráciu supervízora. Priečinok *scripts* obsahuje pomocné skripty pre NERD prebrané z pôvodnej verzie systému.