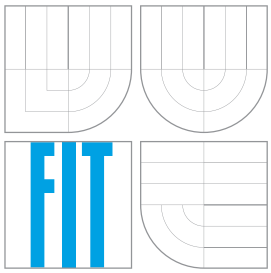
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ANOTACE NETFLOW DAT Z POHLEDU BEZPEČNOSTI

ANNOTATION OF NETFLOW DATA FROM PERSPECTIVE OF NETWORK SECURITY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. LUKÁŠ KADLETZ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN ŽÁDNÍK, Ph.D.

BRNO 2016

Zadání diplomové práce/18858/2015/xkadle29

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Kadletz Lukáš, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Anotace NetFlow dat z pohledu bezpečnosti**
Annotation of NetFlow Data from Perspective of Network Security
Kategorie: Počítačové sítě

Pokyny:

1. Nastudujte problematiku měření síťového provozu pomocí technologie NetFlow. Seznamte se s nejčastějšími síťovými bezpečnostními událostmi, které lze v NetFlow datech detekovat.
2. Analyzujte způsob manuální i automatizované detekce bezpečnostních událostí.
3. Navrhněte vhodný formát pro anotaci NetFlow datové sady z pohledu bezpečnostních událostí.
4. Navrhněte nástroj využívající kombinaci nastudovaných detekčních přístupů pro offline anotaci NetFlow datové sady. Pokud je to možné, využijte již implementované detekční přístupy a rovněž využijte data reportovaná třetími stranami (např. systémem Warden).
5. Navržené řešení implementujte.
6. Funkčnost implementace demonstrujte na vhodně zvoleném vzorku dat.
7. Zhodnoťte dosažené výsledky a navrhněte možná rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Žádník Martin, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Tato práce se zabývá návrhem a implementací aplikace pro offline anotaci NetFlow datové sady z pohledu síťové bezpečnosti. V práci je podrobně popsána architektura NetFlow spolu se způsoby detekce bezpečnostních událostí v zachycených datech. Návrh aplikace je vytvořen na základě analýzy manuální anotace a podpořen několika UML diagramy. Vytvořená aplikace využívá systém Nemea pro detekci bezpečnostních událostí a systém Warden jako zdroj informací o nahlášených událostech na síti. Webová aplikace je postavena na PHP 5 s využitím Nette frameworku, knihovny jQuery a Bootstrap frameworku. Sdružení CESNET poskytlo NetFlow data pro testování aplikace. Pomocí vytvořené aplikace je možné analyzovat a následně anotovat zachycené NetFlow záznamy a vytvořenou datovou sadu využít například pro ověření, zda detekční nástroje pracují správně.

Abstract

This thesis describes design and implementation of application for offline NetFlow data annotation from perspective of network security. In this thesis is explained the NetFlow architecture in detail along with methods for security incidents detection in the captured data. The application design is based on analysis of manual annotation and supported by several UML diagrams. The Nemea system is used for detecting security events and Warden system as a source of information about reported security incidents on the network. The application uses technologies such as PHP 5, Nette framework, jQuery library and Bootstrap framework. The CESNET association provided NetFlow data for testing the application. The result of this thesis could be used for analysis and annotation of NetFlow data. Resulting data set could be used to verify proper functionality of detection tools.

Klíčová slova

NetFlow, anotace, anotovaná datová sada, bezpečnost, bezpečnostní incident, síťový útok

Keywords

NetFlow, annotation, annotated data set, security, security incident, network attack

Citace

KADLETZ, Lukáš. *Anotace NetFlow dat z pohledu bezpečnosti*. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Žádník Martin.

Anotace NetFlow dat z pohledu bezpečnosti

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Martina Žádníka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Kadletz
24. května 2016

Poděkování

Rád bych na tomto místě poděkoval vedoucímu mé diplomové práce panu Ing. Martinovi Žádníkovi, Ph.D., za metodické vedení, osobní přístup a mnoho cenných rad, kterých jsem při realizaci této práce využil.

Obsah

Kapitola 1

Úvod

Bezpečnost počítačových sítí je v dnešní době velmi důležitou součástí každé síťové infrastruktury. Správci počítačových sítí se stále více zaměřují na ochranu integrity a důvěryhodnosti sítě. Je pro ně prioritou ochránit síť před útoky zvenčí a zajistit tím bezpečí klientských stanic. Útoků stále přibývá a to především těch, které lze snadno automatizovat.

Cílem této práce je vytvořit nástroj pro offline anotaci NetFlow datové sady. Aplikace má využívat stávajících detekčních přístupů a dat o výskytu bezpečnostních událostí nahlášených třetími stranami. Funkčnost implementace bude demonstrována na vhodně zvoleném vzorku provozu sítě CESNET. Vytvořená a správně anotovaná datová sada může sloužit k testování již existujících aplikací za účelem detekce bezpečnostních událostí. Aplikaci bude možné využít pro analýzu síťového provozu.

Kapitola 2 se zabývá sběrem síťového provozu a významem uchovávání dat. Následuje představení sdružení CESNET a topologii sítě, která je zdrojem dat k anotaci. Dále je v práci popsána kompletní architektura NetFlow včetně podrobného popisu všech funkčních součástí. Kapitola obsahuje popis několika bezpečnostních událostí spolu se způsobem jejich odhalení v NetFlow záznamech. Jsou zde zmíněny například útoky DoS, hádání hesel, skenování sítě a další. Následuje kapitola 3 obsahující analýzu manuální anotace NetFlow dat, koncepční návrh aplikace a navržený formát pro uložení anotací. Použité nástroje a technologie jsou uvedeny v kapitole 4. Návrh aplikace včetně UML diagramů a grafického uživatelského prostředí je popsán v kapitole 5. Implementací aplikace se zabývá kapitola 6, která mimo jiné vysvětluje strukturu aplikace a uvádí implementační detaily vybraných částí aplikace. Předposlední kapitola 7 obsahuje postup testování aplikace a popis vytvořené anotované datové sady. Závěrečná kapitola 8 zhodnocuje dosažené výsledky a uvádí možnosti dalšího vývoje projektu.

Práce navazuje na můj semestrální projekt, který rozšiřuje do podoby diplomové práce. V rámci semestrálního projektu jsem vypracoval kapitolu 2 včetně analýzy detekce bezpečnostních událostí s využitím NetFlow dat. Navrhl jsem formát pro uložení anotovaných dat a popsal jej v kapitole 3.3.

Kapitola 2

Síťový provoz a detekce bezpečnostní události

Tato kapitola obsahuje seznámení se sběrem dat o síťovém provozu a detekcí bezpečnostních událostí. Nejprve se pokusím zamyslet nad důvody ke sběru dat a vyjádřit k nim své stanovisko. Stručně představím sdružení CESNET z.s.p.o., které mi poskytlo přístup k nasbíraným datům z reálného provozu celé sítě. Následně uvedu protokol NetFlow. Popíši celou architekturu a vysvětlím, jak funguje exportér, kolektor a komunikace mezi nimi. Uvedu také možnosti zobrazení NetFlow dat v grafické podobě.

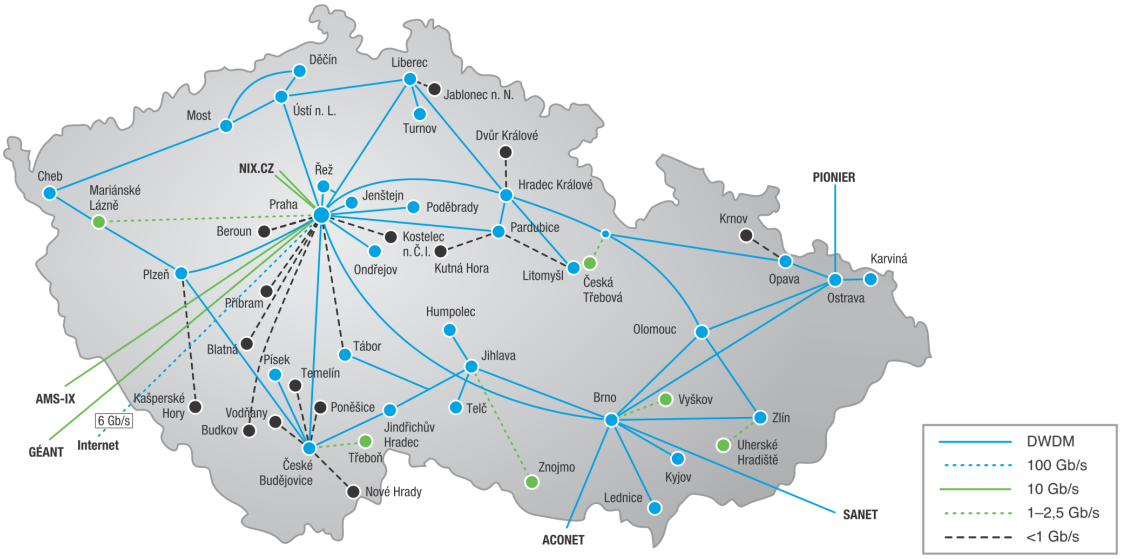
2.1 Důležitost uchovávání dat o síťovém provozu

Existuje mnoho protokolů, které byly sestaveny pro sběr dat ze sítě a následnou analýzu. Jsou jimi například SNMP (Simple Network Management Protokol), WMI (Windows Management Instrumentation), ICMP (Internet Control Message Protocol) nebo právě protokol NetFlow. Společným cílem zmíněných protokolů je kontrola funkčnosti počítačové sítě. Některé protokoly data ukládají (SNMP, NetFlow), jiné slouží především pro zpětnou vazbu o vyskytnutí chyby na síti (ICMP).

Data o síťovém provozu jsou uchovávána proto, aby v nich bylo možno nalézt vzory chování sítě z pohledu dnů, týdnů a měsíců. Nasbírané údaje následně pomáhají správcům sítě k plánování údržby nebo zálohování serverů. Poskytovatelé internetového připojení jsou dokonce za zákona povinni uchovávat data o uživatelích po dobu několika měsíců. Data používají například k účtování nebo k analýze pro rozprostření zátěže při budoucím rozšiřování sítě. Další je na řadě otázka bezpečnosti. Získaná data totiž mohou sloužit i pro pozdější detekci síťových útoků nebo odhalení nakažených počítačů.

To vše jsou jednoznačné argumenty pro uchovávání dat o síťovém provozu. Je třeba si dát pozor na zachování soukromí uživatelů. Uchovávaná data lze snadno zneužít, a proto musí správci sítě zajistit jejich bezpečné uložení.

V následujících podkapitolách představím sdružení CESNET jako zdroj dat pro anotaci a blíže představím samotný protokol NetFlow. Vysvětlím jeho architekturu, jednotlivá zařízení a také jeho silné a slabé stránky.



Obrázek 2.1: Topologie páteřní komunikační sítě CESNET2 na konci roku 2014 [4, s. 24].

2.2 Sdružení CESNET

Sdružení CESNET z.s.p.o., bylo založeno v roce 1996 veřejnými vysokými školami a Akademií věd České republiky. Hlavními cíli sdružení jsou provozování a rozvoj páteřní sítě spojující všechny členy sdružení, výzkum a vývoj pokročilých síťových technologií a rozvoj e-infrastruktury CESNET určené pro výzkum a vzdělávání [4, s. 10]. Sdružení je od začátku stavěno jako neziskové a je tedy plně závislé na podpoře státu, dotacích a finančních darech. Je členem významných mezinárodních i národních organizací, jako jsou například: CZ.NIC, NIX.CZ, TERENA, DANTE, CEENet, GÉANT Association a další.

Nyní se zaměřím na podstatné informace o sdružení CESNET přímo související s tématem této práce. Nejprve představím infrastrukturu sítě a následně uvedu informace o měřících bodech, které budou sloužit jako zdroj NetFlow dat k následné anotaci.

Infrastruktura sdružení CESNET

Páteřní komunikační infrastruktura sítě CESNET2 je koncipována jako vícevrstvý systém propojený v jednotlivých vrstvách výzkumnými a zahraničními sítěmi. Topologii sítě názorně zobrazuje obrázek 2.1. Každá linka opouštějící obrys České republiky, nese vlastní identifikátor. Topologie odpovídá stavu ke konci roku 2014 [4, s. 24]. Následující výčet zobrazuje aktuální uzly sítě CESNET2 (leden 2016) spolu s přibližným objemem nasbíraných NetFlow dat za jeden měsíc:

- `ACONET` – 600 GB
- `AMSIX` – 800 GB
- `GEANT` – 200 GB
- `NIX` – 1 TB
- `PIONEER` – 150 GB

- `SANET` – 350 GB
- `TELIA` – 1,5 TB

Celkem se v síti CESNET2 zachytává okolo 4,5 TB NetFlow záznamů za jeden měsíc. Jedná se o obrovské množství dat, i když jde pouze o metadata ke každému toku procházející skrze měřící bod.

2.3 Protokol NetFlow

NetFlow je síťový protokol pro účely monitorování provozu. Byl vyvinut firmou Cisco[1], která jej nejprve implementovala do svých zařízení jako proprietární protokol. Později, v roce 2004, byl standardizován a popsán v RFC 3954 [2]. NetFlow sbírá data o síťových tocích pomocí analýzy hlaviček IP paketů. Nyní je třeba definovat pojem *síťový tok*.

> „*Síťový tok je posloupnost paketů mající společnou vlastnost a procházející bodem pozorování v jednom směru za určitý časový interval.*“ [27, s. 322]

Pro identifikaci toku se využívá celkem sedm klíčových položek v hlavičce paketu [27]:

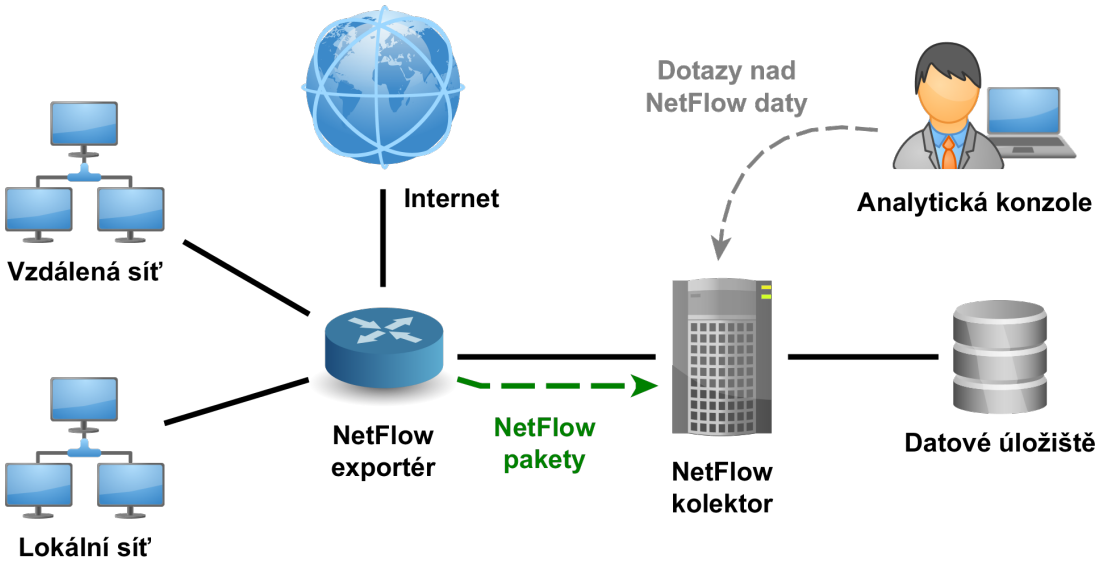
- Zdrojová IP adresa
- Cílová IP adresa
- Zdrojový port
- Cílový port
- ToS – Type of Service
- Typ protokolu na 3. síťové vrstvě
- Logické rozhraní – Vstupní rozhraní toku, tzv. ifIndex (například Et0/0, Se3/0.16 atd.)

Každý paket mající shodných všech sedm položek patří do jednoho síťového toku. Samotných sedm položek nám již podá základní informaci o toku, ale pro lepší analýzu se u každého unikátního toku sbírají další doplňující informace. Každá verze NetFlow přináší rozšíření o možnost ukládání dalších doplňujících informací o aktivním toku. Pro jednu síťovou komunikaci je nutné vytvořit minimálně dva toky, za předpokladu, že bude požadována odpověď druhé strany. V tom případě se zamění zdrojová IP adresa za cílovou a zdrojový port za cílový port. Může se však stát, že odpověď druhé strany bude skrz internet směrována jinou trasou a nebude v zachycených datech jednoho exportéru viditelná.

Architektura NetFlow

Na začátek sběru NetFlow dat v síti je vhodné nejprve určit body, které jsou z hlediska IP toků zajímavé. Především takové, jimiž proudí největší množství dat. Většinou se jedná o hraniční směrovače, stejně je tomu i v případě zobrazeném na obrázku 2.2. Směrovač podporující protokol NetFlow se nazývá **exportér**. Jeho hlavním úkolem je uchovávat záznamy o tocích, případně provádět vzorkování nebo agregaci. Pokud hraniční směrovače protokol

[1] Cisco – Cisco Systems, Inc., `http://www.cisco.com/`



Obrázek 2.2: Diagram NetFlow architektury zobrazuje hraniční směrovač sloužící zároveň jako NetFlow exportér. Analytická konzole pak slouží administrátorovi k dotazování nad NetFlow daty uloženými v datovém úložišti.

nepodporují, je třeba zařídit duplikaci provozu a odklonit jej na externí Exportér, který po analýze pakety zahodí. Podobný princip využívá například zařízení FlowMon[2] od firmy Flowmon Networks, a.s.[3]. FlowMon sonda se připojí k síti pomocí zrcadlení portů SPAN[4] nebo pomocí síťového zařízení TAP[5] [27].

Samotný exportér potřebuje ke své činnosti server přijímající data o exportovaných tocích. Takovému zařízení se říká **kolektor**. Kolektor má většinou pod správou více NetFlow exportérů, kteří jej pravidelně zásobují informacemi o detekovaných tocích. Kolektor komunikuje s exportéry pomocí síťového protokolu NetFlow. Většinou je v jedné zprávě obsaženo více toků, které mohou být od exportérů již agregovány. Kolektor disponuje velkou pamětí pro uchování historie síťového provozu v řádu několika měsíců až let. Dle obrázku 2.2 je posledním prvkem NetFlow architektury analytická konzole. Jde o možnost nechat provádět nad daty různé dotazy a jejich výstupy zobrazit do snadno čitelné podoby. Většinou se jedná o grafické znázornění časových řad provozu [27].

V následujících odstavcích se zaměřím konkrétněji na každý prvek NetFlow architektury a podrobně rozvedu jeho úlohu v rámci celého systému.

Exportér

NetFlow exportér má za úkol analyzovat veškerý síťový provoz a dle IP hlaviček každého z paketů rozhodnout, ke kterému toku patří. Exportér vyhledává klíčové položky IP hlaviček

[2]FlowMon – Neinvazivní řešení pro monitorování, analýzu a dlouhodobé uchování NetFlow dat v síti. https://www.flowmon.com/cs/products/flowmon/probe

[3]Flowmon Networks, a.s. – Vývoj a podpora řešení FlowMon. Firma vznikla rozdělením firmy InveaTech. https://www.flowmon.com/

[4]SPAN – duplikace síťového provozu zařízení na jeden port (Switched Port Analyzer) [27, s. 278]

[5]TAP – hardwarové síťové zařízení sloužící k duplikaci a odbočení provozu za účelem monitorování (Test Access Point) [16]

uvedené v úvodu kapitoly 2.3. Načtené položky pak spolu představují klíč jedinečně identifikující tok. Tento klíč pak ve formě haše používá exportér jako ukazatel do paměti. Hašovací tabulka pro ukládání informací o tocích se v NetFlow terminologii nazývá **NetFlow Cache**. Výhodou hašovací tabulky je rychlý výpočet indexu do paměti, který je ihned využit pro vkládání nebo aktualizaci dat. Pro každý paket exportér zkontroluje v paměti přítomnost záznamu o příslušném toku. Pokud záznam existuje, exportér aktualizuje informace o aktivním toku. Každý záznam navíc obsahuje celkový počet přenesených paketů, celkovou velikost přenesených paketů, časovou značku začátku a konce toku a další údaje. Dle verze protokolu NetFlow se liší ukládaný typ informací. Pokud záznam o toku neexistuje, exportér vytvoří nový záznam se všemi klíčovými položkami, vloží časové razítko začátku toku a doplní výše zmíněné informace.

Exportér je schopen velmi přesně určit začátek toku, jelikož se na místě indexu nenachází žádný záznam. Většinou je však mnohem obtížnější určit konec síťového toku. Následující výčet přehledně zobrazí způsoby určování časové značky konce toku [27, s. 327]:

- **Aktivní časovač** – Pokud je některý z toků aktivní po delší dobu (ve výchozím nastavení po 30 minut), dojde k ukončení toku na straně exportéru a tok je předán k odeslání na kolektor. Další příchozí paket bude již patřit do nového toku. Na kolektoru je možné takovéto toky opět sloučit. Aktivní časovač je velmi důležitý v případě analýzy v reálném čase na straně kolektoru. Bez aktivního časovače by se kolektor o daném toku nemusel v určitém časovém intervalu dovědět a nemohl jej tedy zahrnout do analýzy. Jedná se především o toky trvající několik hodin či dnů.

- **Pasivní časovač** – Jakmile jsou síťové toky po určitou dobu neaktivní (ve výchozím nastavení se jedná o 15 sekund), jsou prohlášeny za ukončené a předány k odeslání na kolektor. Pasivní časovač je jediný možný způsob zjištění konce toku u nespojovaných komunikací, například skrze protokol UDP.

- **Komunikace TCP** – Protokol TCP je spojovaný, což znamená, že zahajuje každou komunikaci tzv. *TCP 3-way handshake* procesem [19, kap. 3.4]. Ukončení komunikace se provádí odesláním příznaku `FIN` (případně `RST`) na který druhá strana odpoví příznakem `ACK` [19, kap. 3.5]. Z předchozí věty plyne, že u TCP komunikace lze jednoznačně určit konec toku jako čas průchodu paketu, který potvrzoval předchozí příznak `FIN`.

- **Přeplnění NetFlow Cache** – Zvláště u vysokorychlostních sítí může nastat stav, kdy je celá NetFlow Cache zaplněna a nelze tedy přidávat další záznamy o tocích. V tom případě se již nahromaděné toky ukončí a jsou urychleně předány k odeslání na kolektor. NetFlow terminologie nazývá tuto funkci jako `free-up`. Aktivní i pasivní časovač se při použití funkce `free-up` neuplatňují. Jde o co nejrychlejší uvolnění paměti pro další aktivní toky. Definice NetFlow přímo neříká, které toky je třeba ukončit. Nejlogičtější variantou je nejprve odebrat zatím nejdéle trvající toky nebo vyprázdnit celou NetFlow Cache. Záleží na konkrétní implementaci exportéru.

Další z funkcí exportéru může být vzorkování. Vzorkování je proces, při kterém se pomocí předem určené heuristiky rozhodne, které pakety mají být zaznamenány a které nikoliv. Může se jednat i o náhodný výběr, kdy se určí pravděpodobnost ignorování procházejícího paketu. Vzorkování s náhodným výběrem ignorovaných paketů se s výhodou využívá pro určení trendů sítě, jež jsou i přes ztrátu některých informací v rámci malé odchylky srovnatelné. Vzorkování navíc podobně jako agregace snižuje počet ukládaných a přenášených dat.

Exportér také může provoz filtrovat rovnou při sběru dat, soustředit se například jen na jeden protokol, jednu cílovou stanici nebo rozsah portů. Vše závisí na konkrétní implementaci NetFlow exportéru.

Kolektor

NetFlow kolektor je software běžící nejčastěji na výkonné serverové stanici. Má schopnost přijímat data o tocích z exportérů vhodně rozmístěných v síti a ukládat je na diskové úložiště [27, s. 322]. Server kolektoru často disponuje nejen velkým výpočetním výkonem, ale také velkým úložným prostorem. Zachycená data jsou pouze ve formě metadat, přesto je celkový objem uložených dat obrovský. Data je třeba uchovávat i několik měsíců až let. V České republice dle zákona o elektronických komunikacích číslo 127/2005 Sb. §97 odst. 3 je poskytovatel veřejné komunikační sítě povinen uchovávat provozní a lokalizační údaje po dobu 6 měsíců [13, §97]. Většinou však poskytovatelé uchovávají data déle především kvůli případným soudním sporům nebo pozdější analýze dlouhodobých trendů. Pro představu se dle mých zkušeností pohybujeme v řádu několika jednotek až desítek gigabytů dat denně v závislosti na velikosti sítě a použitém vzorkování nebo agregaci.

Kolektor přijímá data ze všech exportérů v síti a má tedy o celé síti přehled. Nad přijatými daty může kolektor provádět agregaci nebo jiné operace. Agregace dat je způsob, jak zmenšit objem ukládaných dat na úkor ztráty některých informací. Určitá data v závislosti na typu sítě totiž nemusí být vůbec důležitá a je tedy vhodné využít agregaci pro úsporu diskového prostoru. Pokud je potřeba například zjistit, kolik dat bylo přeneseno mezi dvěma body za určité období, lze toky agregovat dle přenášených dat a zanedbat tak zdrojové a cílové IP adresy nebo porty. Agregace záznamů o tocích je dostupná až od NetFlow verze 8 a může ji provádět exportér i kolektor. Naopak vzorkování typicky provádí pouze exportér, jelikož on je prvním článkem setkávajícím se s daty a může na základě předem určené heuristiky některá data vynechat.

Protokol NetFlow

Nyní se zaměřím na samotný protokol NetFlow, tedy na formát a typ dat odesílajících se z exportéru na kolektor. Protokol k přenosu dat využívá nespojovanou službu UDP z důvodu menšího zatížení sítě. NetFlow pakety putují sítí velmi často a použití spojované služby TCP by zvýšilo provoz na síti. Exportér je schopen v rámci jednoho exportu na kolektor odeslat okolo 30 toků v závislosti na verzi NetFlow [27, s. 329]. Existuje několik verzí protokolu NetFlow, každá vyšší verze přináší něco nového. Většinou rozšiřuje možnosti předchozí verze o další zaznamenatelné údaje o tocích. Ve stručném výčtu jsou uvedeny dosud používané verze NetFlow [18, s. 87]:

- **Verze 1** – První základní verze protokolu NetFlow, dnes se již téměř nepoužívá. Společnost Cisco doporučuje využívat NetFlow verze 5 nebo 9 [27, s. 330].
- **Verze 5** – Přidává podporu protokolu BGP[6], tedy schopnost uchovávat informace o autonomních systémech. Dále stávající protokol rozšiřuje o sekvenční čísla toků sloužící k detekci ztráty dat.
- **Verze 7** – Speciální verze použita pouze v zařízeních Cisco Catalyst 5000.

[6] BGP – směrovací protokol v prostředí autonomních systémů (Border Gateway Protocol)

```
Top 10 flows ordered by bytes:
Date flow start         Duration Proto    Src IP Addr:Port         Dst IP Addr:Port   Flags Tos  Packets    Bytes      pps      bps    Bpp Flows
2005-08-30 06:50:11.218  700.352 TCP     126.52.54.27:47303 ->    42.90.25.218:435   ......   0    1.4 M    2.0 G     2023    5.6 M   1498     1
2005-08-30 06:47:06.504  904.128 TCP   198.100.18.123:54945 ->    126.52.57.13:119   ......   0   567732  795.1 M      627    2.5 M   1468     1
2005-08-30 06:47:06.310  904.384 TCP     126.52.57.13:45633 ->  91.127.227.206:119   ......   0   321148  456.5 M      355    4.0 M   1490     1
2005-08-30 06:47:14.315  904.448 TCP     126.52.57.13:45598 ->  91.127.227.206:119   ......   0   320710  455.9 M      354    4.0 M   1490     1
2005-08-30 06:47:14.316  904.448 TCP     126.52.57.13:45629 ->  91.127.227.206:119   ......   0   317764  451.5 M      351    4.0 M   1489     1
2005-08-30 06:47:14.315  904.448 TCP     126.52.57.13:45634 ->  91.127.227.206:119   ......   0   317611  451.2 M      351    4.0 M   1489     1
2005-08-30 06:47:06.313  904.384 TCP     126.52.57.13:45675 ->  91.127.227.206:119   ......   0   317319  451.0 M      350    4.0 M   1490     1
2005-08-30 06:47:06.313  904.384 TCP     126.52.57.13:45619 ->  91.127.227.206:119   ......   0   314199  446.5 M      347    3.9 M   1490     1
2005-08-30 06:47:06.321  790.976 TCP     126.52.54.35:59898 ->   132.94.115.59:2466  ......   0   254717  362.4 M      322    3.7 M   1491     1
2005-08-30 06:47:14.316  904.384 TCP     126.52.54.35:59773 ->  55.107.224.187:11709 ......   0   272710  348.5 M      301    3.1 M   1340     1
```

Obrázek 2.3: Výstup síťových toků programem `nfdump` seřazený dle počtu paketů.

- **Verze 8** – Přidána možnost seskupení více NetFlow záznamů do jediného toku. Zásadně se tato funkcionalita projevila na velikosti přenášených dat po síti směrem ke kolektoru a také na snížení objemu ukládaných dat. Cisco zařízení mají přednastavená určitá schémata agregace, která lze výhodně využít.

- **Verze 9** – Dnes je spolu s verzí 5 nejpoužívanější verzí NetFlow. Verze 9 zavedla tzv. šablony, díky nimž je možné exportovat informace o toku v určité předem dohodnuté šabloně. Zásadní výhodou šablony je možnost uživatelsky definovat jednotlivá políčka, jejich datový typ či délku a tím umožnit rozšíření NetFlow záznamu o doplňující informace.

- **IPFIX** – Jedná se v podstatě o desátou verzi protokolu NetFlow, která je samostatně standardizovaná v RFC 5101 [7]. IPFIX[7] znovu definuje způsob exportu dat z exportérů a ukládání na kolektor. Vychází z NetFlow verze 9, ale jedná se o samostatný standard, a proto tyto protokoly nejsou navzájem plně kompatibilní.

V posloupnosti verzí jsou mezery, které jsou způsobeny tím, že se většinou jednalo pouze o návrhy, které nakonec nikdy nebyly zrealizovány.

Záznam o síťovém toku v protokolu NetFlow tedy obsahuje pouze metadata. Neobsahuje žádné informace o aplikačních datech, proto zde není ani URL adresa HTTP toku, ani zprávy zasílané signalizačním protokolem SIP a jiné. Bylo by přínosné, kdyby záznamy obsahovaly výše zmíněné informace, to by ovšem způsobilo dramatický nárůst uchovávaných dat. Na vysokorychlostních sítích by ani nebylo fyzicky možné všechna data zachytit.

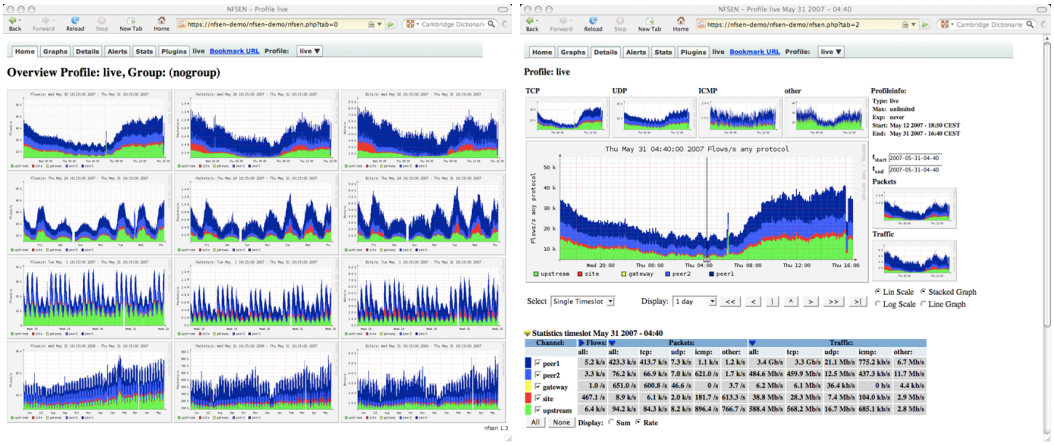
Analýza a zobrazení dat

Jakmile se podaří získat data o tocích v síti, je třeba se zamyslet nad jejich praktickým využitím. Data mohou být analyzována softwarem, který v nich automatizovaně provádí určené operace. Může se jednat o detekci anomálií, síťových útoků, virů nebo červů napadajících okolní stanice v síti. Pro vlastní náhled na nasbíraná data lze s výhodou použít například program `nfdump`[8]. Nfdump je sadou nástrojů šířených pod licencí BSD[9] umožňující manipulaci s NetFlow daty. Obsahuje tyto nástroje:

- `nfcapd` – Nástroj pro čtení NetFlow dat přímo ze sítě nebo z interního úložiště.

- `nfdump` – Přijímá NetFlow data z nástroje nfcapd a zobrazuje je v uživatelsky čitelné podobě, případně v podobě snadno zpracovatelné jinými softwarovými prostředky (výstupy ve formátu CSV, surová data oddělena znakem roura „|“, atd.).

[7] IPFIX – Internet Protokol Flow Information Export [7]
[8] Nfdump – sada nástrojů pro analýzu a zobrazení NetFlow dat, http://nfdump.sourceforge.net/
[9] Licence BSD – licence pro svobodný software, http://www.linfo.org/bsdlicense.html



Obrázek 2.4: NfSen - snímky webové aplikace při vizualizaci NetFlow dat. Převzato z oficiálních webových stránek: http://nfsen.sourceforge.net/

- `nfprofile` – Vytváří filtrované soubory (`profiles`) pro pozdější využití.
- `nfreplay` – Načítá data z nástroje nfcapd a přeposílá je do sítě na určitou adresu.
- `nfclean.pl` – Jednoduchý skript pro odstranění starých a nepoužitých dat.
- `ft2nfdump` – Načítá NetFlow data z různých zdrojů a konvertuje je do formátu vhodného pro zpracování nástrojem nfdump.

Nástroj `nfdump` lze snadno ovládat z příkazové řádky, kde lze pomocí parametrů vložit požadovaný dotaz a čekat na výsledek. Je možné zobrazit pouze metadata ze souboru s NetFlow daty a ta poté použít v dalším zpracování. Obrázek 2.3 zobrazuje výpis jednotlivých záznamů o tocích pomocí nástroje `nfdump`. Data jsou seřazena dle celkového počtu přenesených bytů. Hlavička výpisu obsahuje položky, jež jsou součástí každého NetFlow záznamu. Kromě klíčových položek se ukládají například i TCP příznaky (`SYN`, `ACK`, `FIN` atd.). Obvykle je nástroj `nfdump` používán dalšími programy zpracovávajícími určitá NetFlow data. Programátoři aplikaci musí brát v úvahu, že při dotazování nad obrovskými daty může vykonání nevhodně zvoleného dotazu trvat i několik minut. Nejvyšší zpoždění většinou zaviní diskové pole, které není schopno data načíst v potřebné rychlosti a navíc se je často ani nepodaří uložit do operační paměti RAM. Samotný procesorový výkon na zpracování dotazu dostačuje.

Pro ruční analýzu dat je vhodné grafické rozhraní. V grafickém znázornění provozu jsou jasně viditelné špičky, anomálie nebo výpadky exportérů. Existuje mnoho softwarových produktů, které lze pro vizualizaci nasbíraných dat použít. Osobně jsem pracoval se systémem **NfSen**[10], jenž je postaven jako webová aplikace s mnoha možnostmi zobrazení NetFlow dat. Na obrázku 2.4 lze vidět snímky spuštěné aplikace. NfSen je napsán v programovacích jazycích PHP a Perl. Pro dotazování se nad NetFlow daty využívá program `nfdump`.

Obecně lze pomocí NetFlow získat velké množství dat, a proto je nutné znát jejich využití. Získaná data je třeba řádně chránit proti zneužití. Jedná se o velmi citlivé a snadno zneužitelné informace, například ke sledování činnosti konkrétního stroje v daný časový interval. Tyto znalosti mohou být zneužity přímo útočníky nebo škodlivými programy.

[10] NfSen – webová aplikace pro grafické zobrazení NetFlow dat, http://nfsen.sourceforge.net/

Příklady použití NetFlow v praxi

NetFlow má v praxi široké využití. Používají jej administrátoři počítačových sítí i poskytovatelé internetového připojení. Nyní se pokusím stručně nastínit možnost využití NetFlow za konkrétním účelem [6].

- **Monitorování sítě** – Online monitorování vytížení sítě umožňuje pružnou reakci na náhlé komplikace nebo problémy v síti.
- **Plánování síťových událostí** – Díky dlouhodobému sběru NetFlow dat lze detekovat trendy chování sítě a dle trendů naplánovat spuštění aplikací. Může se jednat o zálohování, datové přenosy, bezpečnostní aktualizace apod.
- **Online i offline analýza síťového provozu** – Správci sítě mohou prostřednictvím doplňkového softwaru analyzovat síť a to buď přímo online nebo offline. Snadno lze detekovat anomálie v provozu a tím odhalit nakažené počítače. Obecně mohou NetFlow využívat systémy IDS[11] a IPS[12]. Například online detekce DDoS[13] útoků podle velkého množství malých toků z různých IP adres na jednu konkrétní IP adresu v jeden časový okamžik. Dále lze v NetFlow datech detekovat viry, červy a jiné nepřátelské programy, které mají určitý vzor chování spočívající v kontaktování ostatních klientů v síti.
- **Účtování** – Dle IP adresy zákazníka lze spočítat, kolik dat bylo v síti přeneseno směrem k zákazníkovy i směrem od něj. Poskytovatel připojení má poté přesný obraz o provozu v síti a může vystavit vyúčtování.

2.4 Detekce bezpečnostních událostí s využitím NetFlow dat

V této podkapitole bych se rád blíže podíval na bezpečnostní události, které lze na síti detekovat. Nejprve definuji samotný pojem a následně rozvedu jednotlivé události. Zaměřím se také na spojitost s NetFlow a zobrazení jednotlivých útoků právě v NetFlow záznamech o tocích.

Bezpečnostní událost

Pro správné pochopení pojmu bezpečnostní události se nabízí využít přímo zákon. Dle zákona České republiky číslo 181/2014 Sb. o kybernetické bezpečnosti je součástí §7 odstavce 1 následující definice kybernetické bezpečnostní události:

> *„Kybernetickou bezpečnostní událostí je událost, která může způsobit narušení bezpečnosti informací v informačních systémech nebo narušení bezpečnosti služeb anebo bezpečnosti a integrity sítí elektronických komunikací.“* [12, §7]

Jinými slovy se jedná o událost, která je potenciálním zdrojem problému.

Pojem bezpečnostní událost je spojen s dalším pojmem a tím je bezpečnostní incident. Následující odstavec stejného zákona definuje bezpečnostní incident následovně:

[11] IDS – systém pro odhalení úspěšného útoku (Intrusion Detection System)
[12] IPS – systém pro detekci a prevenci útoku (Intrusion Prevention Systems)
[13] DDoS – distribuovaný útok za účelem odmítnutí služby (Distributed Denial of Service)

> *„Kybernetickým bezpečnostním incidentem je narušení bezpečnosti informací v informačních systémech nebo narušení bezpečnosti služeb anebo bezpečnosti a integrity sítí elektronických komunikací v důsledku kybernetické bezpečnostní události.“* [12, §7]

Komplikace tedy může nastat například v případě, že při detekci bezpečnostní události nejsme schopni rozhodnout, zda došlo k narušení bezpečnosti a integrity sítě a označit detekovanou událost za bezpečnostní incident.

V následujících kapitolách se budu věnovat vybraným bezpečnostním incidentům, které lze alespoň částečně v NetFlow datech detekovat. Některé bezpečnostní incidenty se mohou skrývat ve velké spoustě dat nasbíraných za dlouhý čas a jejich detekce je velmi náročná. V rámci anotace se budu primárně zabývat bezpečnostními událostmi. Pokud budu schopen nalézt relevantní informace o porušení bezpečnosti a integrity sítě, explicitně uvedu, že se jedná o bezpečností incident s odkazem na zdroj těchto informací.

Skenování sítě

Skenování sítě je základní metoda získávání informací o síťové infrastruktuře. Útočník nebo útočící program (dále jen útočník) se snaží o síti zjistit co nejvíce informací a v nich poté vyhledat možné cíle útoku. Skenování sítě rozlišujeme dvojího typu [1]:

- **Horizontální skenování** – Útočník vyhledává aktivní zařízení v síti v platném IP rozsahu, většinou v dané podsíti. Cílem je získat topologii sítě aktivních prvků. Nejjednodušší horizontální skenování lze provést pomocí příkazu `ping`, který je dostupný na většině platforem. Příkaz odešle zprávu typu `ICMP Echo` a po určitý čas vyčkává na odpověď ve tvaru `ICMP Echo Reply` [20, s. 14]. Využívá se zde protokol ICMP[14], který je základním standardizovaným protokolem pro správu sítě. Pokud v daném časovém okně není obdržena žádná zpráva, cílová stanice je považována za nedostupnou. Často je paket typu `ICMP` blokován přímo na úrovni stanice nebo sítě. Nelze tedy s jistotou tvrdit, že se na dané IP adrese zařízení nenachází.
- **Vertikální skenování** – Útočník se soustředí na jednu konkrétní stanici a snaží se odhalit, jaké porty jsou otevřené. Pokud útočník dostane odpověď na dotaz u konkrétního portu. Skrze otevřené rezervované porty je možné odhadnout, které aplikace na portech naslouchají.

Detekce v NetFlow datech

Oba dva druhy skenování mají podobné charakteristiky. Většinou se jedná o krátké toky obsahujících jeden až několik paketů. Při testování dostupnosti se v NetFlow datech objeví dva toky. První tok bude například zpráva `ECHO` zaslaná v jednom paketu a druhý tok zase odpověď `ECHO Reply`. Budou to dva různé toky, jelikož byla vyměněna zdrojová IP adresa za cílovou a zdrojový port za cílový port.

Hádání hesel

Hádání hesel je jeden z nejznámějších útoků obecně. Pokud se útočníkovi nepodaří získat přihlašovací údaje uživatelů pomocí jiných metod (např.: malware, phishing, útok na databázi atd.), často se uchýlí k jednoduché alternativě, kterou je hádání hesel. Cílem útoku jsou

[14] ICMP – standardizovaný protokol určený pro správu sítě (Internet Control Message Protokol) [20]

většinou služby SSH, FTP, HTTP atd. Útočník nejprve detekuje služby běžící v počítačové síti a poté se na ně pokusí zaútočit. Existují dvě základní metody hádání hesel:

- **Útok hrubou silou** – Útočník se snaží vyzkoušet všechny možné kombinace hesla. Jedná se o extrémně náročný útok, jehož obtížnost se dramaticky zvyšuje, pokud je služba chráněna dostatečně dlouhým heslem s využitím čísel a speciálních znaků.
- **Slovníkový útok** – Jedná se o modifikaci útoku hrubou silou a to tak, že útočník testuje hesla ze slovníku. Útočník může využít různé slovníky dle lokalizace nebo přímo slovníky nejčastějších hesel. Případně může hesla ze slovníku modifikovat (velká a malá písmena, číslice atd.) a spojovat. Útok je mnohdy úspěšný, jelikož je pro uživatele jednodušší zapamatovat si jednoduché slovníkové heslo.

Společným problémem obou těchto metod je snadná detekce útoku na straně oběti. Administrátoři sítí často využívají možnosti odepření přístupu ke službě na základě mnoha nesprávných pokusů o vložení hesla. Útočníka lze dle IP adresy zablokovat trvale nebo pouze po určitou dobu.

Detekce hádání hesel pomocí NetFlow

Hádání hesel lze ze záznamu o tocích detekovat poměrně snadno. Detekce souvisí se znalostí sítě a služeb na ní běžících. Pokud se například objeví množství IP toků v krátkém čase směřující z jedné IP adresy na port 22, standardně určený pro SSH spojení cílové stanice, téměř určitě jedná o útok. Uživatel se obvykle při zadávání hesla zdrží a je tedy jasné, že nemůže v jednu sekundu vyzkoušet například 10 a více hesel.

Útoky odepření služby

Účelem útoku odepření služby DoS[15] je vyřadit z provozu nebo alespoň zabránit legitimním uživatelům využívat služby oběti útoku. Často je realizován vytvořením obrovského množství požadavků na server, který tyto požadavky nestíhá zpracovávat. Proto se požadavky legitimních uživatelů k serveru vůbec nedostanou, budou zahozeny nebo odpověď na ně trvá tak dlouho, že klientský počítač prohlásí server za nedostupný [8, s. 559]. Distribuované útoky typu DoS jsou značně nebezpečnější, jelikož se útoku účastní velké množství různých strojů z různých IP adres po celém světě. Je proto mnohem obtížnější určit, jestli se jedná o útočníka nebo o legitimního uživatele. Příklady některých DoS útoků [8, s. 560]:

- **Přetečení vyrovnávací paměti** (angl. Buffer overflow) – Cílový proces obdrží mnohem více požadavků než dokáže reálně obsloužit. Pokud není v systému integrována ochrana, dojde k přetečení dočasné paměti a tím dojde k nežádoucímu chování aplikace. Aplikace se může automaticky restartovat nebo v horším případě zastavit a systémový administrátor musí aplikaci ručně ukončit a uvést zpět do provozu.
- **Útok SYN** – Využívá se protokolu TCP, který musí na začátku komunikace nejprve ustanovit spojení metodou *3-way handshake* [19, kap. 3.4]. Útočník vygeneruje množství paketů typu `SYN` pro navázání spojení, avšak již dále neodpovídá. Cílový počítač musí odpovědět paketem typu `SYN-ACK` a po určitou dobu udržovat aktivní sezení, při němž čeká na příjem potvrzovacího paketu typu `ACK`. Druhý paket však již nikdy nedojde. Pokud si počítač alokuje paměť pro nově příchozí sezení dříve, než

[15] DoS – útok za účelem odmítnutí služby (Denial of Service)

obdrží druhý paket `ACK`, může dojít k vyčerpání paměti nebo zdržení v komunikaci. Legitimní uživatel se nemusí dočkat odpovědi `SYN-ACK` a prohlásit cílový počítač za nedostupný.

- **Smurf Attack** – Hlavní roli hraje protokol ICMP. Nejprve útočník nalezne síť, kterou použije k odrazu a znásobení útoku. Jakmile odešle ICMP paket s fiktivní zdrojovou IP adresou na všesměrovou IP adresu celé sítě, obdrží všechny počítače v síti tento paket a odpoví na něj. Stanice ovšem neodpovídají útočníkovi, ale cílové stanici, ta je poté zaplavena dotazy a není schopna vyřizovat další požadavky.

- **Zesílení útoku pomocí DNS serveru** – Jedná se o speciální typ DoS útoku využívající vlastnosti systému DNS[16]. V principu jde o to, že útočník odešle na několik serverů malé DNS dotazy s podvrženou zdrojovou IP adresou (IP adresou oběti). Tyto servery provedou vyřízení požadavku a sestaví paket odpovědi a tu odešlou na IP adresu oběti. Problém je v tom, že odpověď DNS serveru může být v určité konfiguraci až několikanásobně větší než samotný dotaz. Oběť tedy musí čelit obrovskému náporu provozu, který nestíhá zpracovávat. Podobně jako v minulých případech dojde k odmítnutí legitimního uživatele, jehož požadavek nebyl vyřízen v požadovaném čase a cílový server je považován za nedostupný.

Detekce DoS a DDoS útoků v NetFlow záznamech

Společným znakem DoS útoků je tvar vstupních paketů, jejich velikost, cílová stanice či port. Lze tedy měřit počet toků směřujících od jedné stanice k jednomu cíli v určitý časový úsek. Jakmile počet toků překročí zvolenou hranici, lze předpokládat, že se nejedná o legitimní provoz, ale o útok typu DoS a zdrojovou IP adresu zablokovat. Tato reakce se zdá býti ideální ochranou, ale pokud útočník využije například veřejné servery DNS nebo jiné důležité body v síti, potom by jejich zablokování mohlo vést i k nefunkčnosti celého systému. Z tohoto hlediska jsou mnohem záludnější DDoS útoky, neboť je počet útočících stanic velký a nemusí stačit některé z nich pouze zablokovat.

Malware

Malware[17] lze jednoduše popsat jako škodlivý software. Jedná se o viry, červy, trojské koně a různé další programy pro získávání, zneužívání nebo poškozování uživatelských dat.

Detekce Malwaru pomocí NetFlow

Společnou vlastností většiny takových programů je schopnost šířit se po síti a napadat další počítače. Malware musí nejprve proskenovat síť, na niž se nachází a vyhledat potenciální příjemce škodlivých dat. Patrně se bude snažit zkontaktovat i ty body v síti, které by při regulérním provozu nebyly takovýmto způsobem vůbec kontaktovány (např. tiskárny, síťové disky atd.). Při pokusu o šíření škodlivých dat má zdrojová stanice otevřených několik spojení s ostatními prvky sítě a to ji dělá nápadnou z hlediska infikovanosti. Dle známého vzoru chování určitých škodlivých programů lze s velkou pravděpodobností malware odhalit. Nové viry nebo červy lze odhalit též, ale s výrazně větším počtem falešně pozitivních hlášení.

[16] DNS – hierarchický systém doménových jmen (Domain Name System)

[17] Malware – zákeřný nebo škodlivý software (malicious software), http://techterms.com/definition/malware

VoIP provoz

Jedná se o telefonii provozovanou digitálně skrze počítačovou síť. Pro přenos signalizace se v IP telefonii využívá například protokol SIP[18] definovaný v RFC 3261 [22]. Tento protokol je velmi oblíbený a tudíž i náchylný k útokům. Obecně VoIP telefonie patří mezi kritickou infrastrukturu. Z mé vlastní zkušenosti mohu říct, že se správci sítě snaží VoIP provoz odlišit od ostatního a na síťových prvcích takovému provozu zvýšit prioritu. Nejzranitelnějším místem je ústředna, která propojuje síť VoIP telefonů s internetem. Je třeba se zaměřit na ochranu ústředen před útoky typu DoS a DDoS. Útoky za účelem odepření služby je možné v NetFlow datech detekovat a pokud nastane bezpečnostní incident, patřičně na něj reagovat.

Detekce útoků na protokol SIP pomocí NetFlow

Provoz VoIP je ze své podstaty těžko zachytitelný z NetFlow dat, jelikož informace o komunikačním protokolu se vyskytují až na aplikační úrovni. Pro detekti a ochranu proti útokům je třeba sbírat více dat než jen standardní NetFlow záznamy. Lze využít například šablony v NetFlow verze 9 a často i specializovaný hardware, jelikož zaznamenávání dat z aplikačních hlaviček je další zátěž pro exportér. Je možné uchovávat hlavičky na aplikační úrovni protokolu SIP a ty dále použít pro analýzu provozu a vyhledání útoku. Detektor je například schopen analyzovat požadavky typu `INVITE` o uskutečnění hotovu na telefonní číslo v rámci sítě PSTN[19] [11, s. 91-94]. Cílem je nalezení IP adresy, která opakovaně provádí pokusy o volání na telefonní čísla se stejnou předponou. Nasbíraná data lze použít i pro určení úspěšnosti útoku [11, s. 97].

[18] SIP – signalizační protokol pro VoIP telefonii (Session Initiation Protokol) [22]
[19] PSTN – veřejná telefonní síť (Public Switched Telephone Network)

Kapitola 3

Analýza a koncepční návrh aplikace

Hlavním cílem této práce je vytvořit nástroj pro anotaci NetFlow datové sady z pohledu bezpečnosti počítačové sítě. Program by měl využívat existujících řešení pro detekci různých bezpečnostních událostí a vše zkombinovat do ucelené formy za účelem co nejpřesnější anotace dostupných dat. Vytvořená anotovaná datová sada může sloužit jako referenční pro testování nových detekčních přístupů a algoritmů. Bez precizně anotované datové sady je testování detekčních nástrojů velmi komplikované. Programátor je často zbytečně zdržován úvahou, jestli nalezená událost je detekována správně či nikoliv. Případně je pro něj těžké získat relevantní množství dat obsahující různé modifikace útoků.

Pro vytvoření aplikace provádějící anotaci dat, je nejprve nutné nastudovat postup manuální anotace. Analýzou postupu je možné rozhodnout, které části lze automatizovat plně, které pouze částečně a které bude muset anotovat uživatel manuálně. Důraz bude kladen na zvýšení rychlosti při práci s velkými daty. Například program `nfdump` je velmi užitečný, ale jeho dotazy na větším množství dat jsou často velmi výpočetně náročné. Uživatel musí na výstup čekat i desítky sekund. Poslední část kapitoly se zabývá návrhem formátu pro uložení anotací. Formát by měl být snadno čitelný a umožňit uložení podrobností o anotované události.

3.1 Anotace NetFlow dat

Anotace je proces přidávání užitečné informace ke stávajícím datům. V rámci této práce je užitečnou informací myšlen záznam o anomáliích. Může to být skenování sítě, útok hrubou silou na některou přístupnou službu nebo i legitimní provoz, který se však chová nestandardně.

Nabízí se otázka, jaká jsou pro vytvořenou anotovanou datovou sadu využití. V prvé řadě mohou sadu využít programátoři detektorů síťových útoků. Ti poměrně snadno zjistí, jak je vytvořený detektor úspěšný v analýze reálných dat. Lze vypočíst procentuální podíl úspěšných hlášení vůči neúspěšným nebo počet falešně pozitivních. Tyto parametry jsou pro detektory velmi důležité, jelikož lze na jejich základě posouvat parametry detektoru až k optimální hranici. Ladění detektoru a jeho parametrů je pak mnohem snazší a programátoř ušetří podstatnou část práce. Navíc pokud programátor objeví v anotované sadě chybu nebo jeho detektor nalezne neanotovanou událost, lze ji do sady přidat. Postupem času se bude sada vylepšovat. Anotovanou sadu však bude třeba za nějaký čas vytvořit znovu, jelikož její

starší verze by nemusela obsahovat nové útoky nebo anomálie. Je možné však využívat obě vytvořené sady. Druhá možnost pro využití anotované sady je ve výuce na středních nebo vysokých školách. Studentům lze ukázat průběhy různých útoků, případně je nechat útoky vyhledávat. Data by musela být anonymizována, ale průběhy útoků zůstanou zachovány. Posledním důvodem k vytváření anotovaných sad by mohl být neustálý nárůst zachycených dat, jejichž použití v praxi je bez kvalitní anotace komplikované.

3.2 Analýza manuální anotace

Aplikace by měla usnadnit a urychlit náročný proces anotace, a proto bude nejprve provedena analýza manuální anotace. Sestavím číslovaný seznam bodů, které bude nutné pro úspěšnou anotaci dat provést. Navržený seznam bodů:

1. Získat NetFlow záznamy o tocích v určité síti za zvolené období.
2. Požádat o export dat ze systému Warden nebo jiného podobného systému.
3. Instalace detektorů bezpečnostních incidentů.
   (a) Nastudování práce s detektory.
   (b) Otestování funkce detektorů.
4. Stanovit formát pro ukládání anotací.
5. Zobrazit graficky časové průběhy provozu (TCP, UDP, ICMP atd.).
6. Postupně procházet data pomocí programu `nfdump`.
   (a) Vyhledat podezřelý provoz pomocí statistik.
   (b) Využít detektory k získání informací o incidentech.
   (c) Korelace informací z detektorů s informacemi ze systému Warden a reálnými daty.
   (d) Průběžně zapisovat anotace a označovat je v časové řadě.
   (e) Provádět anotaci iterativně, dokud výsledek nebude dostatečný.

Výše uvedené body budu v následujících odstavcích rozebírat, testovat jejich provedení na svém počítači a zaměřovat se na oblasti, které lze automatizovat.

Sdružení CESNET mi poskytlo zachycené NetFlow záznamy o provozu na všech hraničních linkách za období: **18. dubna 2015 – 21. května 2015**. Toto období bylo vybráno z důvodu nejmenšího podílu výpadků exportérů. Data jsou z velké části kompletní a nejlépe proto odpovídají reálnému provozu v celé síti CESNET2. Jde přibližně o měsíc provozu v NetFlow záznamech, které jsou uloženy v komprimované podobě. Práce s komprimovanými daty je paradoxně rychlejší, než práce s nekomprimovanými. Je to z důvodu rychlejšího načtení dat z disku do paměti, a proto nástroj `nfdump` pro analýzu NetFlow záznamů využívá pouze výpočetní výkon procesoru. Dokonce i přes komprimaci jejich celková velikost přesáhla 5 TB. Podrobné statistiky poskytnutých dat zobrazuje tabulka 3.1.

Dále mi byla poskytnuta data ze systému Warden a to přesně v rozsahu sledovaného období. Data jsou uložena v souboru ve formátu CSV a obsahují celkem přes 3,5 mil. hlášení. Celý soubor na disku zabírá 403 MB. Okolo 80% jsou to však hlášení o horizontálním nebo

Linka	Velikost [GB]	Toky [mld.]	Pakety [mld.]	Přenesená data [PB]
Aconet	819,59	25,03	731,36	642,21
Amsix	915,34	27,20	452,27	363,94
Geant	198,36	6,42	247,41	238,52
Nix2	345,37	11,89	295,71	261,60
Nix3	636,13	20,74	526,59	423,45
Pioneer	177,64	5,29	195,03	171,04
Sanet	395,40	13,77	548,85	494,87
Telia	1 832,49	58,63	819,01	641,94
Celkem	**5 320,32**	**168,97**	**3 816,23**	**3 237,57**

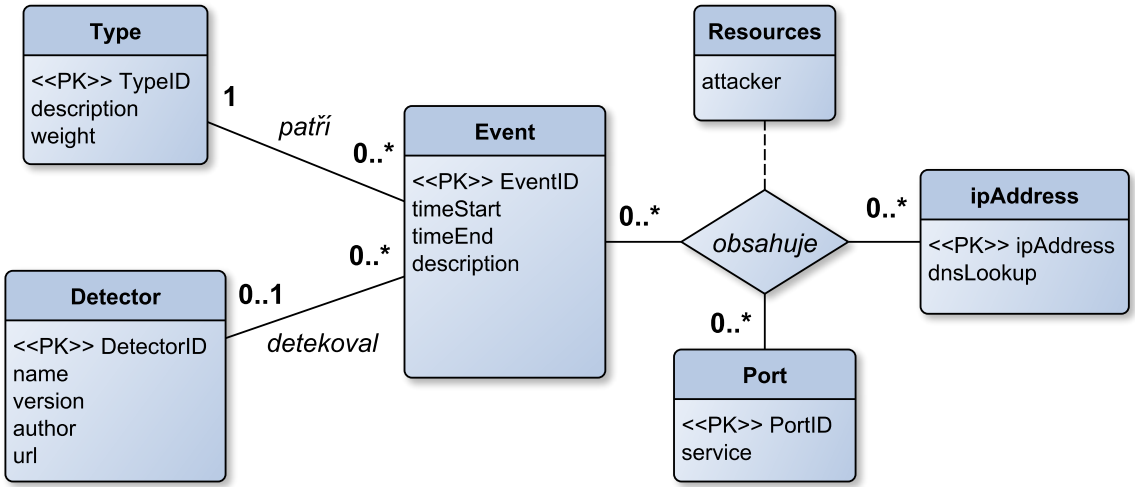
Tabulka 3.1: Statistika zachycených NetFlow dat od 18. dubna 2015 do 21. května 2015

vertikálním skenování sítě. Tyto hlášení typicky poskytují stroje s označením **Honeypot**. Jsou to stroje rozmístěné v síti mající pouze jeden účel. Čekají, až je někdo kontaktuje a jelikož na nich neběží žádná legitimní služba, tak naprostá většina požadavků přichází od útočníků nebo nakažených stanic. Honeypot sdílí informace o pokusech o připojení a na jejich základě se generují hlášení o bezpečnostních incidentech.

Vyhledávání v samotném souboru lze provést pomocí linuxového nástroje `grep`, který po použití přepínače `-E` vyhledá dle rozšířeného regulárního výrazu. Lze tak pomocí několika regulárních výrazů vyhledat hlášení dle časového intervalu, typu nebo IP adresy. Potíž nastává v použití výsledků a zobrazení do snadno čitelné podoby. Aplikace pro anotaci by měla umět CSV dokument efektivně zpracovat a umožnit snadné vyhledávání. Pravděpodobně nepůjde soubor nahrát celý do paměti, jelikož řada aplikací často obsahuje paměťové limity. Při budoucím zpracování většího souboru by mohl nastat problém. Pokusím se nejprve soubor zmenšit pomocí programu `grep` a teprve poté jej načíst do paměti. Předzpracování bude nutné uložit do trvalé paměti, aby byla příští manipulace s obsahem souboru rychlejší.

Body 3.*a* a 3.*b* se zabývají instalací detekčních nástrojů a jejich použitím. Aplikace bude využívat systém Nemea. Postup instalace a zprovoznění je uveden v kapitole 4.1. Využít některý z detektorů je možné pomocí minimálně tří programů. První z nich je program `nfdump_reader` sloužící pro načtení NetFlow dat ze souboru. Druhým programem je `logger`, který zpracovává výstupy z detektoru a posílá je na standardní výstup nebo ukládá do souboru. Poslední je samotný detektor. Spojení modulů lze snadno provést skrze rozhraní TRAP, kdy po úspěšném propojení všech programů se automaticky spustí detekce. Po skončení detekce lze výstup použít pro účely anotace. Zde je velký prostor pro automatizaci postupu od načítání dat až po analýzu výstupu. Uživatel aplikace pro anotaci dat by měl pouze vybrat detekční modul, spustit detekci a po skončení použít výstup. Pokusím se o maximální zjednodušení, aby uživatel nemusel znát závislosti modulů, rozhraní TRAP a další podrobnosti.

Posledním bodem je závěrečná anotace zvoleného intervalu. Jedná se o uživatelsky nejkomplikovanější část, jelikož uživatel musí korelovat výsledky z detektorů systému Nemea a informace ze systému Warden s reálnými daty. K datům má přístup skrze program `nfdump`, který dokáže vytvářet statistiky o provozu nebo jen vypisovat komunikaci určitých IP adres. Aplikace pro anotaci dat by měla co nejvíce zjednodušit tuto část práce. Uživatel by měl mít možnost přehledně zobrazit všechny informace a na jejich základě se rozhodnout a provést anotaci. Navíc tu bude hrát velkou roli předběžné načítání výstupů z programu `nfdump`, jelikož některé dotazy na delší interval provozu trvají i desítky sekund a je zby-



Obrázek 3.1: Entity-Relationship diagram zachycuje vztahy mezi entitními množinami dle návrhu formátu CSV pro budoucí realizaci relační databáze.

tečné je provádět několikrát. Pokusím se obtížně získaná data uložit a při příštím přístupu je uživateli ihned načíst z paměti. Je možné využít ukládání do souboru nebo přímo do databáze.

Jakmile uživatel dokončí anotaci, bude nutné, aby aplikace nabídla export anotovaných událostí do vhodného formátu. Výběrem formátu pro uložení anotovaných dat se podrobně zabývám v kapitole 3.3. Uživatel dostane anotace a spolu s jeho sadou dat je může použít v dalších aplikacích.

3.3 Formát pro uložení anotací (CSV a IDEA)

Nyní bude přiblížen formát pro uložení anotovaných dat. Formát musí být snadno strojově zpracovatelný a musí obsahovat správně naformátované položky. Nejprve se zaměřím na oblíbený formát CSV[1] definovaný v RFC 4180 [24]. Jde o minimalizovaný formát pro uložení dat ve formě tabulky. Prvním řádkem je hlavička určující jméno sloupce. Pokročilé aplikace často vyžadují výskyt datového formátu přímo v hlavičce pro bezproblémové strojové zpracování. Hlavička pro tři sloupce může vypadat například takto:

```
ipaddr IP, time TIME, uint8 EVENT_TYPE
```

První sloupec obsahuje IP adresu, druhý časovou značku a třetí typ události. První slovo je zapsáno malými písmeny a udává datový typ. Druhé slovo označuje název sloupce, využívám velká písmena a místo mezery podtržítko. Při návrhu formátu jsem se inspiroval formátem, který používá program `logger` v systému Nemea. Pokud aplikace tyto datové typy podporuje, je možné takový soubor zpracovat poměrně rychle a jeho uložení v paměti bude díky znalosti datového typu efektivní.

Přehledně vypíši údaje, dle mého názoru, nutné k uvedení do anotace. Předpokládám, že během podrobného návrhu či implementace narazím na další položky, jež bude vhodné do výstupu zahrnout:

[1] CSV – souborový formát pro ukládání dat, snadno strojově čitelný (Comma-separated values) [24]

- `Časová značka začátku a konce události` – Záznam by odpovídal času přijetí prvního a posledního zaznamenaného paketu. Je vhodné použít takovou časovou značku, která je popsána standardem, nabízí se RFC 3339 [17]. Pravděpodobně vyberu některý z formátů: `2016-01-18T18:36:48.12Z` nebo `2016-01-18T16:36:48+02:00`. V ideálním případě bych chtěl využívat časovou značku bez označení časové zóny, protože jde o čas vázaný k analyzovaným NetFlow datům.
- `Zdrojová a cílová IP adresa` – V závislosti na počtu IP adres by bylo možné do jedné kolonky umístit více IP adres. Adresy by však musely být odděleny jiným znakem než je oddělovač CSV formátu. Je možné uvést rozsah pomocí oddělovače `-`.
- `Zdrojový a cílový port` – Podobně jako u IP adres lze v případně výskytu více portů uvést rozsah (např.: 22-110, 1024-55555 apod.)
- `Identifikace útočníka` – Obsahuje jednu IP adresu nebo jejich rozsah. Pokud nelze útočníka jednoznačně určit, zůstane identifikace nevyplněna.
- `Typ události` – Zde je formou předem domluveného řetězce identifikován typ události.
- `Slovní vysvětlení` – Zobrazuje řetězec v lidsky čitelné podobě, který pomůže k rychlé identifikaci události. Mohou zde být odkazy na související události, ale stále se jedná o obyčejný text (strojově nezpracovatelný).
- `Jak byla událost zjištěna` – Zaznamenává, který detektor poskytl informace (název a verze) a jestli byla provedena automatická nebo manuální anotace.

Na základě uvedeného výčtu jsem navrhl tento formát:

```
time TIME_START, time TIME_END, ipaddr IP_SRC, ipaddr IP_DEST, port
PORT_SRC, port PORT_DEST, ipaddr ATTACKER, uint8 TYPE, string DETECTOR,
string DESCRIPTION
```

Pro běh aplikace není vhodné, aby byla data ukládána přímo do souboru ve formátu CSV, proto použiji relační databázi MySQL. Kdykoliv poté bude možno exportovat libovolnou část anotovaných dat do souboru. Relační databáze umožňuje data více separovat a vyjádřit vztahy mezi nimi. Navíc zde mohu pro účely aplikace uchovávat i podrobné informace, například reverzní DNS záznam k dané IP adrese. Entity-Relationship diagram uvedený na obrázku 3.1 naznačuje, jak by bylo možné data a jejich vztahy reprezentovat.

Ukládání informací do formátu CSV jsem navrhl dle osobního uvážení, které položky budou nutné pro kvalitní informaci o anotaci. Podívám se však i na formát, jenž je na podobné účely nasazen v praxi. Je to formát IDEA[2], již používaný v systému Warden jako hlavní formát vstupních dat. Systém Warden slouží pro sdílení a uchovávání hlášení o detekovaných bezpečnostních incidentech. Podrobně se systémem Warden zabývám v kapitole 4.3. Anotace je v principu hlášení o bezpečnostním incidentu, proto tento formát mohu využít. IDEA je sama o sobě velmi rozsáhlá, budu tedy vybírat jen ty položky, které lze naplnit relevantními daty. IDEA pro uložení dat využívá standardizovaný formát JSON[3] definovaný v RFC 4627 [10]. JSON je strohý textový formát pro přenos dat mezi různými systémy. Ve spoustě programovacích jazyků má knihovní podporu a lze jej využít v mnoha aplikacích. Na rozdíl od formátu XML[4] je JSON z hlediska řídících znaků (znaky nesoucí informaci

[2]IDEA – Intrusion Detection Extensible Alert, https://idea.cesnet.cz/en/index
[3]JSON – JavaScript Object Notation, http://www.json.org/
[4]XML – Extensible Markup Language, http://www.xml.com/

o struktuře) výrazně úspornější.

Na základě prostudování celého formátu IDEA jsem vybral relevantní položky použitelné pro anotaci. Následuje příklad anotovaného útoku hrubou silou za účelem přihlášení do počítače oběti:

```
{
    "Format": "IDEA0",
    "ID": "4390fc3f-c753-4a3e-bc83-1b44f24baf75",
    "DetectTime": "2016-01-18T11:39:58Z",
    "EventTime": "2015-05-13T02:27:00Z",
    "CeaseTime": "2015-05-13T02:45:00Z",
    "Category": ["Attempt.Login"],
    "Confidence": 1,
    "Note": "Successful attack. Login: admin",
    "Source": [
       {
          "IP4": ["93.81.210.5"],
          "IP6": ["2001:718:1c01:16:214:22ff:fec9:ca5"],
          "Hostname": ["server.priklad.cz"],
          "Proto": ["ssh"],
          "Port": []
       }
    ],
    "Target": [
       {
          "IP4": ["182.9.23.113"],
          "IP6": ["2001:c1f:1cd1::2f4:2cf:faa:fb11"],
          "Hostname": ["obet.utoku.cz"],
          "Proto": ["ssh"],
          "Port": [22]
       },
    ],
}
```

Příklad obsahuje všechny záznamy, které lze dle mého názoru do anotace přiřadit. Ostatní položky ve formátu IDEA by bylo obtížné vyplnit. Časové značky jsou stejné jako u formátu CSV. Navíc je zde možnost přidat k jedné události více zdrojových a cílových IP adres s informacemi o portech a protokolech. Díky flexibilitě lze vložit pole čísel portů. Ve formátu CSV je možné oddělit čísla portů pouze znakovým oddělovačem. Navíc formát IDEA je již v praxi používán, a proto je tu potenciál, že bude postupem času implementován v dalších aplikacích. Z výše zmíněných důvodů se pravděpodobně zaměřím na formát IDEA a použiji ho jako hlavní formát pro výstupy anotací.

Kapitola 4

Použité nástroje a technologie

Tato kapitola popisuje aplikace a nástroje, které jsou nutné pro implementaci aplikace dle koncepčního návrhu. Obrázek 4.1 přehledně zobrazuje hlavní komponenty, ze kterých se aplikace skládá a zároveň zachycuje jejich hierarchii.

Je nutné se na tomto místě zamyslet nad problematikou velkého počtu komponent v celém systému. Mnohokrát se mi v praxi potvrdilo, že platí přímá úměra: „Čím více prvků a komponent systém obsahuje, tím větší komplikace to přináší.“ Komplikacemi nemyslím jen složitější začleňování nových komponent, ale především udržovatelnost celého systému. Snažil jsem se proto zvolit takové aplikace, které jsou dlouhodobě stabilní a mají širokou komunitu uživatelů a programátorů. Takové aplikace mají v dnešní době šanci na udržení stávající kvality. Případný přechod na nové verze bývá často velmi dobře zdokumentován a pro programátora to nepředstavuje velký problém.

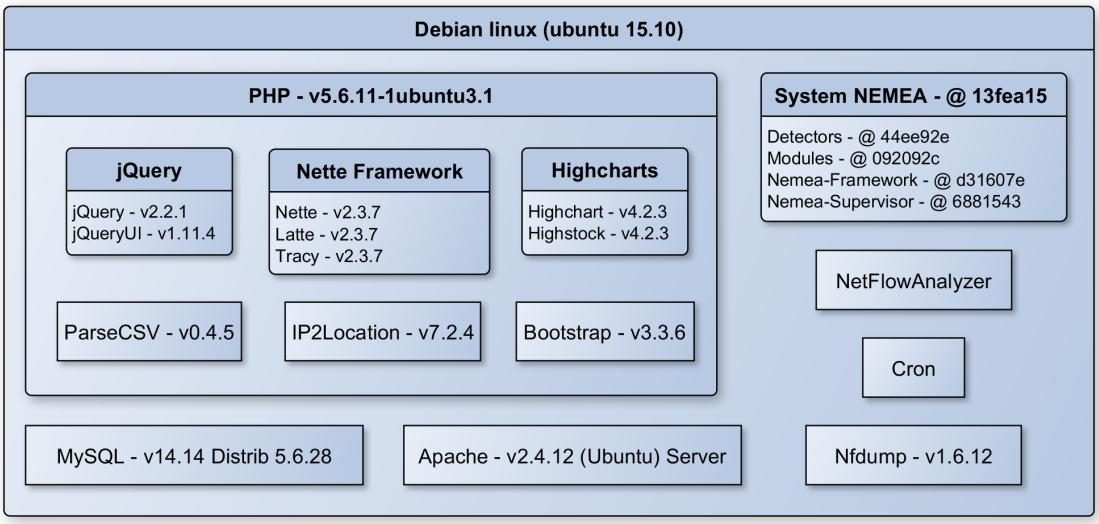
První část je věnována systému **Nemea** pro analýzu síťového provozu, který vyvíjí sdružení CESNET. Již delší dobu je systém nasazen v praxi. Ve druhé části se ve stručnosti věnuji webových technologiím. Mimo jiné představím **Nette framework** sloužící jako základ celé aplikace, **jQuery** knihovnu pro pokročilou práci s JavaScriptem a **Bootstrap framework** zajišťující responzivní design webových stránek. Třetí část je shlukem několika dalších užitečných aplikací, které využívám povětšinou jen z části, ale určitě by zde neměly chybět. Zmíním například systém **Warden** pro sběr informací o bezpečnostních incidentech nebo projekt **Highcharts** určený pro pokročilou práci s grafy.

4.1 Systém NEMEA

Součástí této práce je využití již stávajících detektorů síťových útoků a anomálií pro zpřesnění a urychlení tvorby anotace. Právě zde se nabízí využít systém Nemea[1]. Jedná se o modulární systém pro analýzu síťového provozu a detekci anomálií, který je vyvíjen sdružením CESNET z.s.p.o.. Systém je využíván především pro detekci útoků v reálném čase, ale díky jeho komplexnosti lze s výhodou využít i pro zpracování již nasbíraných dat [3]. Základní stavební bloky systému[2] jsou:

- **Nemea framework** – Obsahuje knihovní jádro celého systému. Jsou zde implementovány veškeré nástroje a podpůrné prostředky k běhu modulů a detektorů.

[1] NEMEA – Network Measurements Analysis, https://www.liberouter.org/technologies/nemea/
[2] Repozitář systému NEMEA: https://github.com/CESNET/Nemea

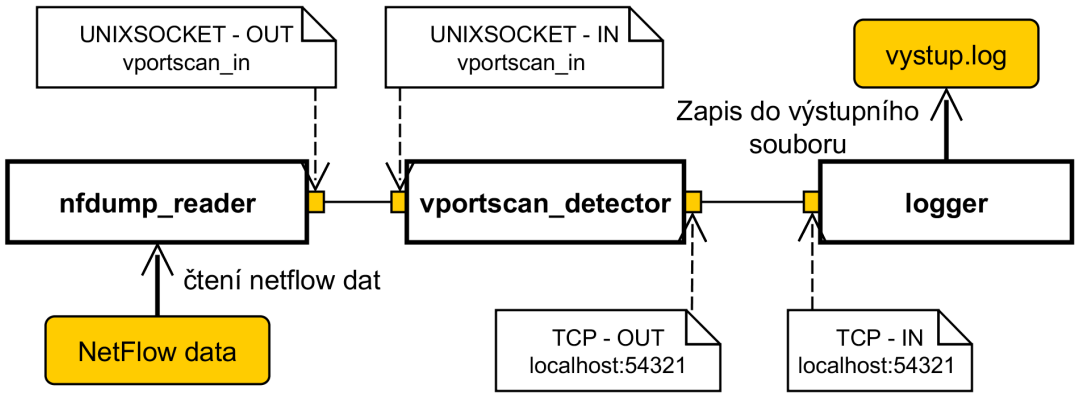


Obrázek 4.1: Grafické znázornění celého systému a závislosti jednotlivých komponent.

- **Nemea modules** – Zde se vyskytují všechny moduly sloužící jako podpora pro detektory. Typickým zástupcem je `logger`, který je umístěn na každé výstupní rozhraní detekčního modulu a zajišťuje uložení údajů o detekovaných událostech. Dále modul `nfdump_reader` sloužící ke čtení NetFlow záznamů a k přeposílání na vstupní rozhraní detektoru. V neposlední řadě je zde implementován i `anonymizer` sloužící k anonymizaci NetFlow záznamů.
- **Nemea detectors** – Hlavní detekční moduly systému Nemea. Moduly jsou schopné detekovat útoky typu DoS, DDoS, horizontální a vertikální skenování, DNS amplifikaci, útoky hrubou silou (angl. brute-force attacks), apod. Celý systém umožňuje velmi pohodlnou implementaci nových detekčních modulů, kdy se programátor může zcela soustředit na vlastní algoritmus. Proto je do budoucna velká naděje ke vzniku nových detekčních modulů, které pak bude možné využít i na reálné síti.
- **Nemea Supervisor** – Jedna z nejdůležitějších částí systému sloužící pro konfiguraci a monitorování Nemea modulů. Supervizor skrze konfiguraci ve formátu XML spouští, zastavuje a monitoruje již běžící moduly. Supervizor může také běžet v interaktivním módu i jako proces na pozadí operačního systému, kdy se k jeho konfiguraci přistupuje skrze nástroj `supervisor_cli`. Nově přibyla i možnost spuštění supervizora jako systémové služby. Supervizor bude důležitý pro implementaci aplikace, jelikož usnadňuje práci s procesy a lze díky obnovení konfigurace přidat nebo odebrat libovolný modul. Aplikace pro anotaci dat bude totiž využívat detekčních modulů a prostřednictvím supervizora moduly spouštět.

Nemea framework

Následující řádky slouží k hlubšímu pochopení základu, na kterém celý systém Nemea staví. Nemea framework obstarává především komunikaci mezi moduly, formát předávaných zpráv, společné datové struktury a algoritmy. Představím jednotlivé prvky a stručně vysvětlím k čemu slouží. Této části se podrobněji věnuji především proto, že od uživatele



Obrázek 4.2: Grafické znázornění spojení modulů skrze rozhraní včetně parametrů spojení.

vyvíjené aplikace očekávám alespoň základní znalost systému Nemea a schopnost jej použít. Důležité bude znát alespoň základy komunikačního rozhraní `TRAP`.

- **Libtrap** – Knihovna TRAP implementuje vrstvu pro komunikaci mezi jednotlivými moduly. Každý modul pak snadno pomocí jednotného rozhraní může vytvářet vstupní či výstupní komunikační kanály k ostatním modulům. Spojení lze provést skrze linux socket nebo TCP spojení. Knihovna zároveň sjednocuje konfiguraci modulů, kdy všechny moduly mají společný přepínač `-i` skrze který se jednotným formátem definují vstupní i výstupní rozhraní. Na obrázku 4.2 je uveden příklad spojení třech modulů systému Nemea. Nejprve je spuštěn program `nfdump_reader` s přepínačem `-i u:vportscan_in`. Jako druhý v pořadí se spustí program `logger` s přepínačem `-i t:54321:1`. Nyní jsou oba konce linie spuštěny a lze spustit detektor. Program `vportscan_detector` je spuštěn s přepínačem `-i u:vportscan_in,t:54321`. Spojení všech modulů je kompletní a může mezi nimi probíhat komunikace. Průběžně jsou výstupy detektoru zapisovány, dle konfigurace programu `logger`, do souboru `vystup.log`.
- **UniRec** – Univerzální datový formát pro standardizaci komunikace mezi moduly. Univerzálnosti se zde dosahuje pomocí UniRec šablon, kdy při komunikaci s modulem proběhne kontrola položek v šablonách vstupního i výstupního rozhraní. Pokud bude platit, že vstupní rozhraní je podmnožinou rozhraní výstupního z jiného modulu, bude možné spojení navázat. UniRec obsahuje statickou a dynamickou část. Statická část obsahuje základní parametry síťových toků, např.: `DST_IP`, `SRC_IP`, `BYTES`, `PACKETS`, `DST_PORT`, `SRC_PORT` apod. Dynamická část oproti statické obsahuje libovolné další položky dle aktuální šablony, např.: `URL`, `TIME_FIRST`, `TIME_LAST` a další.
- **Common** – Společný základ datových struktur a funkcí.
- **Python** – Implementace knihovny TRAP v jazyce Python.
- **Pycommon** – Další společná část obsahující mimo jiné i konvertor výstupu do formátu IDEA pro reportování do systému Warden.

Instalace systému

Instalace systému Nemea je možná více způsoby v závislosti na zvoleném operačním systému nebo preferencí programátora. Nejpohodlnější způsob je skrze binární balíky RPM, které jsou dostupné pro systémy *CentOS7* a *Scientific Linux 6*. Já jsem však použil variantu instalace pomocí *GNU/Autotools*. Postup instalace je podrobně popsán přímo v souboru `README.md` dostupného v hlavním repositáři Nemea na adrese: https://github.com/CESNET/Nemea. Ve zkratce jde o zadání těchto příkazů:

```
$ git clone --recursive https://github.com/CESNET/nemea
$ ./bootstrap.sh
$ ./configure --prefix=/usr --bindir=/usr/bin/nemea
              --sysconfdir=/etc/nemea --libdir=/usr/lib64
$ make
$ sudo make install
```

Může se stát, že skript `configure` během kontroly narazí na problémy se závislostmi, ty je třeba vyřešit. V mém případě (Debian Linux - Ubuntu 15.10) jsem doinstaloval pouze knihovnu `libnfdump`[3] sloužící jako aplikační rozhraní pro čtení NetFlow záznamů. Po instalaci se všechny spustitelné soubory nacházejí v adresáři `/usr/bin/nemea`. Každý detektor obsahuje podrobné informace o jeho použití přímo v repositáři v souboru `README.md`. Moduly lze skládat za sebe dle výše popsaného rozhraní TRAP.

4.2 Webové technologie

Webová část aplikace je realizována pomocí skriptovacího jazyka PHP, relační databáze MySQL a jazyků HTML, CSS a JavaScript. V následujících odstavcích stručně představím jednotlivé technologie a frameworky na těchto technologiích postavené.

Jazyk PHP

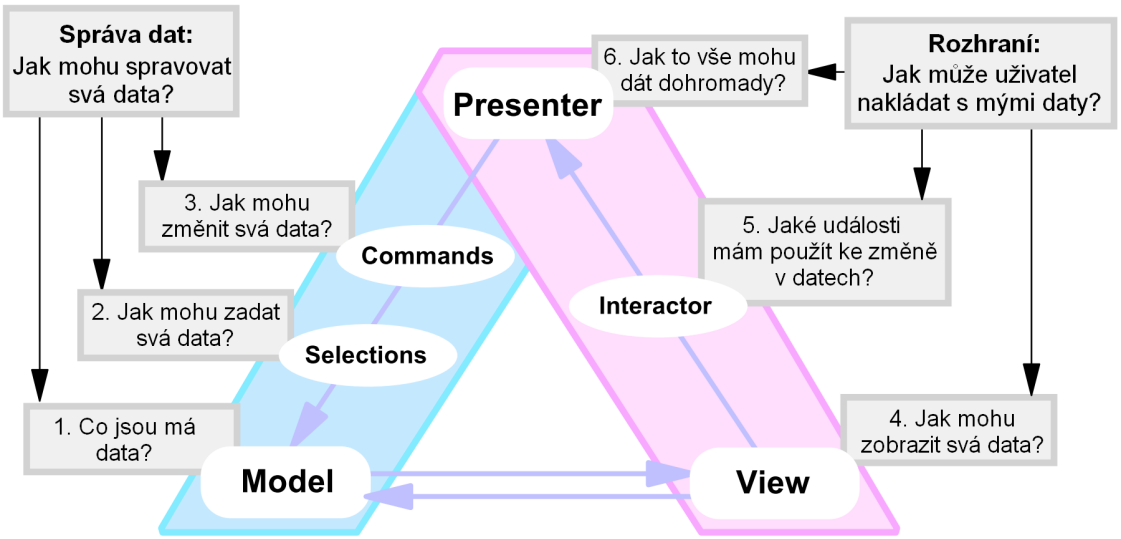
Skriptovací jazyk PHP[4] je určen pro tvorbu dynamických webových stránek. Pro většinu z poskytovatelů hostingových služeb je PHP základem, protože je mezi programátory velmi oblíbený. První webové stránky byly v PHP napsány kolem roku 1994 [26]. Od té doby je PHP aktivně vyvíjeno a zdá se, že ještě několik let bude jedním z hlavních programovacích jazyků pro webové aplikace. Podpora objektově orientovaného programování je samozřejmostí. Obsahuje knihovny pro práci databázemi (MySQL, Oracle, PostgreSQL apod.), tvorbu grafů, analýzu souborů XML a mnoho dalšího [26].

Vyvíjená aplikace poběží nejprve na lokálním stroji, ale do budoucna ji bude možné nainstalovat na libovolný server, kdy využití jazyka PHP jistě nebude překážkou. Aplikace je vyvíjena na aktuální verzi PHP 5.6. Minulý rok v prosinci vyšla nová verze PHP7[5], která má být revolučním skokem v životě PHP. Myslím si, že poskytovatelé hostingových služeb nezačnou PHP7 nasazovat ihned, ale vyčkají na požadavky svých zákazníků.

[3] Zdrojové kódy knihovny `libnfdump`: http://libnfdump.sourceforge.net

[4] PHP – rekurzivní zkratka PHP: Hypertext Preprocessor

[5] Zpráva autorů PHP 7.0.0 dne 3.12.2015:
https://secure.php.net/archive/2015.php#id2015-12-03-1



Obrázek 4.3: Architektura MVP, otázky správy a reprezentace dat. (vlastní překlad [21])

Nette framework

Tvorba aplikací v čistém PHP je v dnešní době spíše výjimkou, protože je stále složitější napsat kvalitní webovou aplikaci, která je odolná vůči známým útokům (XSS[6], SQL Injection[7] apod.). Většinu z těchto problémů řeší právě Nette framework. Nette je český framework, jehož autorem je *David Grudl*, který jej vydal v roce 2008 jako open source. Autor se stále aktivně podílí na jeho vývoji [15]. Je myslím dobře, že se autor stále velkou mírou angažuje, ovšem tato skutečnost skýtá několik komplikací do budoucna. Nyní má Nette sice obrovskou komunitu uživatelů, ovšem aktivních programátorů je méně. Autor se snaží postupně zapojovat více programátorů a já věřím, že je to na dobré cestě. Aplikace implementována v Nette frameworku využívá softwarovou architekturu MVP[8]. Na obrázku 4.3 je názorně popsána funkce každého prvku architektury. Nesporná výhoda této architektury spočívá v oddělení logiky aplikace od zobrazovací části a lze tedy aplikaci pohodlně vyvíjet ve větší skupině programátorů. Další výhodou Nette frameworku je nativní podpora AJAJ[9]. Je to technika asynchronního zpracování webových stránek využívající textový formát JSON pro přenos dat [15]. Velmi se podobá technice AJAX, která využívá komunikační formát XML.

Aplikaci jsem postavil na aktuální a stabilní verzi Nette 2.3, která bude dle oficiálních webových stránek minimálně do února 2017 udržována. Framework je šířen pod licencemi: *New BSD* a *GPL (GNU General Public License)*. Uživatel se může rozhodnout, která licence mu více vyhovuje a kterou pro svůj projekt použije. Pro vytvářenou aplikaci využiji licenci New BSD, která dostačuje ve všech ohledech a mimo jiné umožňuje využít framework i v komerčních projektech.

[6] XSS – narušení webových stránek pomocí uživatelsky přidaných „zlých“ skriptů (Cross-site scripting)
[7] SQL Injection – provedení pozměněného SQL dotazu skrze neošetřené uživatelské vstupy
[8] MVP – architektonický třívrstvý model (Model-View-Presenter) [21]
[9] AJAJ – Asynchronous JavaScript and JSON [15]

Relační databáze MySQL

Aplikace má pro potřeby ukládání nastavení a výsledných anotací přístup k databázi MySQL. Systém MySQL je relační databází patřící do skupiny RDBMS[10] [9, s. 4]. Aplikace je postavena na aktuální verzi 5.6, jako formát tabulek jsem zvolil *InnoDB* a kódování v již standardní znakové sadě UTF-8. Aplikaci lze pohodlně migrovat i na jiné databázové systémy, jako jsou například Oracle, PostgreSQL atd. To vše díky Nette frameworku, který obsahuje konektor pro databázovou vrstvu. Logika SQL dotazů je tedy napsána v obecném formátu a po změně konektoru se automaticky přeloží na patřičný typ databáze. Jedná se o prvek zvyšující přenositelnost aplikace mezi různými systémy.

Bootstrap framework

Aplikace využívá standardní technologie pro zobrazení webové stránky, jako HTML, CSS a JavaScript. Všechny jmenované využívá také Bootstrap framework[11]. Jde o open source CSS framework vytvořen pány Mark Otto a Jacob Thornton kolem roku 2011 pro potřeby sociální sítě Twitter [25]. Framework je určen především pro responzivní design aplikace, ale také v sobě zahrnuje spoustu užitečných funkcí a pravidel. Například standardizace CSS, kdy jsou ve frameworku obsaženy inicializace základních vlastností [25]. Při použití většího množství webových prohlížečů pak výsledek vypadá velmi podobně. Navíc framework dodržuje i určitá typografická pravidla a proto většina aplikací působí velmi dobře. Framework je doplněn o JavaScriptovou vrstvu, kterou návrhář využije snadno skrze datové HTML atributy (např.: `data-toggle="collapse"` pro zobrazení nebo skrytí prvku na stránce). Použití Bootstrap frameworku sice není přímo podmíněno využitím knihovny jQuery, ale pokud je knihovna v aplikaci přítomna, lze využít vestavěné funkce například pro skrývání prvků, jejich opětovné zobrazení prvků, apod. Knihovna jQuery je představena v následujících odstavcích.

Knihovna jQuery

Malou chvíli ještě zůstanu u JavaScriptu a představím knihovnu jQuery[12]. Jedná se o velmi úspěšnou knihovnu postavenou nad JavaScriptem. Výhoda spočívá ve snadné selekci jednotlivých, ale i více elementů na stránce. Selekce je prováděna pomocí tzv. *selektorů*. Programátor si snadno vybere ty správné prvky a s nimi potom může manipulovat. Dále knihovna umožňuje měnit kontext stránky, přidávat HTML elementy, modifikovat jejich parametry, apod. [14] Historie jQuery sahá až do roku 2006. Autorem je John Resig, který právě v roce 2006 jQuery poprvé prezentoval na akci BarCamp[13] v New Yorku. Dnes se jQuery skládá z celkem pěti částí:

- **jQuery** – Hlavní knihovna obsahující nejpoužívanější funkce. Běžnému programátorovi by měla tato knihovna dostačovat.
- **Sizzle** – Slouží jako podpora pro selekci prvků na stránce (`https://sizzlejs.com/`).
- **jQuery UI** – Zaměření na uživatelské rozhraní použité na různých zařízeních. Zpracovává interakce s uživatelem, efekty a pokročilé formuláře (`https://jqueryui.com/`).

[10] RDBMS - Relational Database Management System
[11] Webové stránky Bootstrap frameworku: `http://getbootstrap.com/`, `http://getbootstrap.com/css/`
[12] Oficiální webové stránky knihovny jQuery: `http://jquery.com/`
[13] Zpráva autora Johna Resiga o prezentaci jQuery: `http://ejohn.org/blog/barcampnyc-wrap-up/`

Obrázek 4.4: Příklad grafů generovaných skrze Highcharts framework.

- **jQuery Mobile** – Pomáhá optimalizovat aplikace pro mobilní zařízení a dotekové obrazovky. Vytváří responzivní design s využitím moderních prvků HTML5. (https://jquerymobile.com/)
- **QUnit** – Framework pro testování JavaScriptového kódu. (https://qunitjs.com/)

Aplikace pro svůj běh využije jQuery ve verzi 2.2.1 a jQueryUI ve verzi 1.11.4. Knihovna je šířena pod licencí *MIT Licence*, která podobně jako u Nette frameworku umožňuje i komerční použití bez souhlasu autorů a neobsahuje žádné požadavky na webové stránky. Díky velké komunitě uživatelů je velmi pravděpodobné, že bude knihovna stále udržována a vyvíjena.

4.3 Nástroje a aplikace

Tato podkapitola si klade za cíl seznámit čtenáře s dalšími aplikacemi, které byly určitým způsobem použity při implementaci. Většinu aplikací využívám formou knihovny. U každé aplikace uvedu hlavní důvody pro její výběr a funkci v rámci aplikace pro anotaci datové sady.

Highcharts

Highcharts[14] je univerzální framework pro práci s grafy postavený nad jazykem JavaScriptem. Přímo v aplikaci budu využívat část zvanou `Highstock`. Uživatel si bude moci snadno zobrazit časové průběhy sledovaných dat a vyhledávat v nich vhodná místa pro podrobnější analýzu. Framework zajišťuje přibližování grafů, výběr intervalu či vytvoření posuvného časového okénka skrz celý graf. Mimo jiné je uživatel schopen graficky přijatelně zobrazit jednotlivé anotace jako speciální body v grafu. Bude tak na první pohled patrné, ve které části grafu se vyskytují jaké typy anomálií nebo útoků.

Využívám aktuální verzi Highstock JS v4.2.3, která vyšla v polovině února roku 2016. Framework je zdarma pro nekomerční využití a je tedy jasné, že díky zájmu komerčního sektoru je framework kvalitně vyvíjen poměrně velkým týmem programátorů. Obrázek 4.4 zobrazuje dva typy grafů, které budou pravděpodobně vhodnými kandidáty k použití v aplikaci.

[14] Highcharts – oficiální webové stránky frameworku: http://www.highcharts.com/

Obrázek 4.5: Statistika stavu systému Warden ze dne 28. dubna 2016. Obrázek je převzat z oficiálních webových stránek projektu: https://warden.cesnet.cz/

IP2Location

Portál IP2Location[15] se zaměřuje na prodej databází s velkým množstvím informací o každé konkrétní IP adrese, to jak ve verzi IPv4, tak IPv6. Server nabízí PHP modul, který přímo pracuje s jejich databázemi a umí z nich efektivně získat potřebné informace. Tento modul jsem využil k získávání základních lokalizačních údajů na základě IP adresy. Dle licenčních podmínek jsem použil databázi `IP2LOCATION-LITE-DB1`, která obsahuje vazbu mezi IP adresou a státem, do kterého IP adresa patří. Pro orientaci úroveň státu dostačuje a v budoucnu bude možné snadno přejít na jiný typ databáze s podrobnějšími informacemi. Licence mi dále ukládá povinnost viditelně na webových stránkách zveřejnit zpětný odkaz na portál IP2Location. Především si cením možnosti stáhnout databázi v binární podobě určené pro rychlé vyhledávání. Databáze má velikost pouze `1.1 MB`[16] a obsahuje celkem 71 483 záznamů. Komprimace počtu záznamů je výsledkem uložení jednoho záznamu pro interval IP adres jdoucích po sobě a patřící do jednoho regionu.

Warden

Warden[17] je systémem pro sdílení a uchování informací o detekovaných bezpečnostních událostech. Obrázek 4.5 ukazuje aktuální stav systému. Celková velikost databáze okolo 37 GB je obrovská a v aplikacích se většinou využije pouze export části dat za určité období nebo s určitým filtrem. Aplikace využije export událostí ve formátu `CSV`. Warden sice již umí nový datový formát `IDEA`, ale já mám k dispozici export patřící k zachyceným datům z roku 2015 a tedy pouze ve formátu `CSV`. Informace o událostech je možné využít u výstupů programu `nfdump`. Každou IP adresu pak lze snadno zkontrolovat a v případě shody s některou bezpečnostní událostí, zobrazit uživateli varování.

Systém Warden má bohužel také svá úskalí. Často se totiž stane (experimentálně zjištěno procházením dat a ručním dohledání nahlášených incidentů v zachycených datech), že některé časové značky bezpečnostních událostí neodpovídají zachyceným datům. Je to způsobeno špatnou synchronizací času u detekčních systémů. Tento problém se snažím eliminovat vyhledáváním s ochranným intervalem dvou hodin tam i zpět. Jelikož má záznam ze systému Warden za celý měsíc okolo `400 MB`, musím jeho procházení značně omezit. Omezení provedu utilitou `grep` a následné zpracování omezeného rozsahu provedu v PHP.

[15] IP2Location – webový portál se nachází na adrese: https://www.ip2location.com/developers/php
[16] Kompletní databázi lze stáhnout ze stránek: https://lite.ip2location.com/database-ip-country
[17] Warden – Systém pro sdílení informací o bezpečnostních událostech. https://warden.cesnet.cz/

Kapitola 5

Návrh aplikace

Návrh aplikace vychází z poznatků získaných během analýzy manuální anotace NetFlow dat v kapitole 3. Zaměřím se na podrobný návrh aplikace od definice případů užití, přes návrh struktury až k návrhu grafického uživatelského prostředí. V kapitole 4 jsem představil hlavní nástroje, které budou součástí aplikace. Během popisu vytvořených diagramů se budu na tyto nástroje odkazovat.

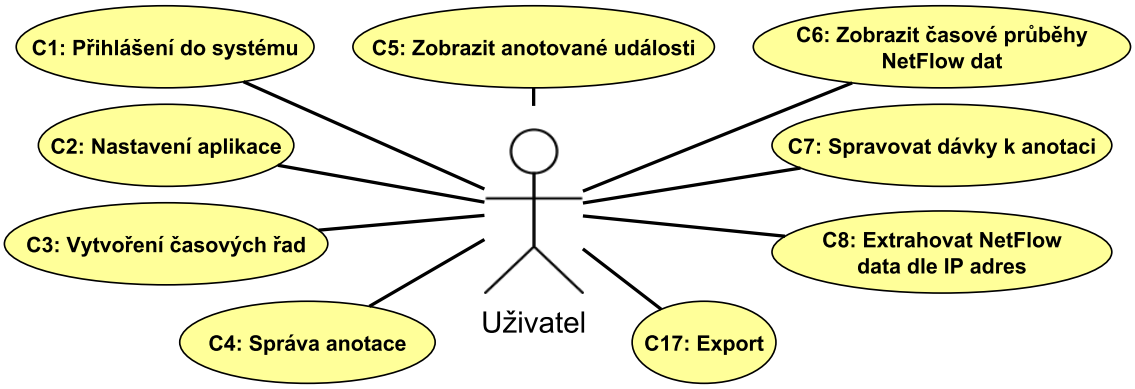
Abych dokázal navržený systém co nejlépe popsat, využiji vyjadřovací sílu jazyka UML[1]. Jazyk UML slouží pro standardizovaný popis programových systémů pomocí diagramů. Byl poprvé standardizován v listopadu roku 1997 standardizační skupinou OMG[2], která dodnes stojí za jeho rozvojem [23]. Aktuální verze UML 2.5[3] vyšla v březnu roku 2015. Nejčastěji se však, dle mé zkušenosti, v praxi využívá verze 1.3 (březen 2000) nebo 2.0 (červenec 2005), jelikož pro většinu návrhů zcela postačuje. Součástí každého UML diagramu jsou standardní prvky a stereotypy, jejichž správné použití výrazně zvyšuje čitelnost [23]. Předpokládám, že pokud čtenář této práce zná základní pravidla a typy UML diagramů, bude pro něj pochopení návrhu výrazně jednodušší.

Nejprve se zaměřím na podrobný návrh aplikace pomocí diagramu *případů užití*. Pro vyjádření návaznosti jednotlivých obrazovek využiji *stavový diagram*. Následně se zaměřím na formu ukládání dat do relační databáze dle koncepčního návrhu v kapitole 3.3. Nakonec této kapitoly dám prostor návrhu grafického uživatelského prostředí. Grafický design a uživatelská přívětivost je jistě důležitou součástí každé větší aplikace.

5.1 Diagram případů užití

Digram případů užití, jak už název napovídá, slouží pro popis použití systému jednotlivými aktéry. Systém obsahuje dva aktéry. Prvním je samotný *Uživatel*, který ovládá celý systém. Druhým zvláštním aktérem je *Čas*, který reprezentuje spouštění naplánovaných událostí v určitém čase bez interakce s uživatelem. V následujících odstavcích představím vybrané případy užití a upozorním na podstatné nebo zajímavé detaily. Pro přehlednost bude popis diagramu prokládán jeho výřezy. Kompletní diagram případů užití je dostupný v příloze B.1.

[1] UML – Unified Modeling Language
[2] OMG – Object Management Group
[3] UML 2.5 – http://www.omg.org/spec/UML/2.5/PDF



Obrázek 5.1: Výřez digramu užití soustředěn na aktéra *Uživatel* (Kompletní diagram je dostupný v příloze B.1)

Základní práce s aplikací

První část diagramu zobrazuje uživatele a několik případů užití, které může uživatel přímo vykonávat. Výřez diagramu je zobrazen na obrázku 5.1. Aplikace je navržena pro více uživatelů, a proto je třeba zařadit do diagramu případy užití **C1** (přihlášení do systému) a **C2** (nastavení aplikace). Každý uživatel musí před zahájením procesu anotace nastavit systém a všechny potřebné konstanty. Jelikož NetFlow data většinou zabírají hodně místa na disku, nelze je pouze nahrát skrze webové rozhraní. Návrh aplikace počítá s tím, že uživatel bude mít přístup k serveru na kterém je aplikace zprovozněna. Data bude mít uložena na pevném disku, případně na jiném externím úložišti přímo připojeném k serveru. Serverem je myšlen počítač s nainstalovanou aplikací. Může se jednat i o virtuální stroj, který si uživatel spustí pod libovolným systémem (např.: pomocí aplikace VirtualBox, VMware apod.).

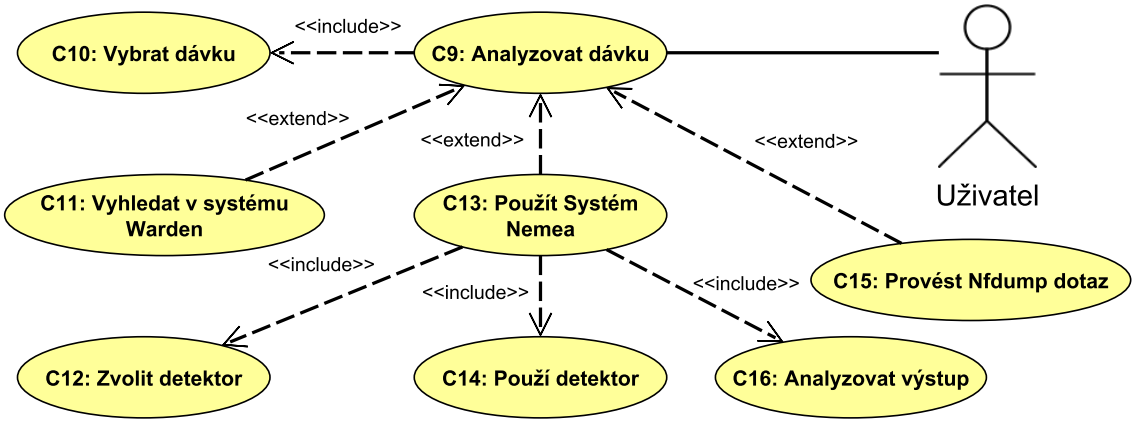
Případ užití **C3** (vytvoření časových řad) bude uživatel spouštět v případě, kdy vložil nová NetFlow data do úložiště. Je totiž potřeba z každého souboru načíst statistiky o provozu a vytvořit z nich časové řady. Bude se jednat o formu předzpracování. Dle mých předběžných měření na SSD disku bude načtení statistik o jednom dni provozu na jedné lince trvat okolo 10ti sekund v závislosti na velikosti dat. Tato operace se však bude provádět pouze jednou a načtená data se uloží do paměti pro pozdější zpracování. Načtené časové průběhy lze zobrazit pomocí případu užití **C6** (zobrazit časové průběhy NetFlow dat).

Aplikace musí umět co nejintuitivnější formou nabídnout uživateli výpis všech již anotovaných událostí a umožnit jejich export do formátů CSV a IDEA. Vše zahrnují případy užití **C5** (Zobrazit anotované události) a **C17** (Export). Případ užití **C4** (správa anotace) slouží k přidání nové anotace. Lze jej také použít k úpravě či odstranění stávající anotace. Uživatel bude moci anotace spravovat na více místech, abych co nejvíce zjednodušil a urychlil anotaci.

Během procesu anotace bude uživatel analyzovat pouze menší část dat, kterou v rámci návrhu označuji jako *dávka*. Případ užití **C7** (spravovat dávky k anotaci) zahrnuje veškerou práci s dávkami. Umožňuje uživateli vytvořit novou dávku, měnit její rozsah nebo ji předávat aktérovi *Čas* pro další zpracování.

Anotace vytvořené dávky

Druhá část diagramu je s aktérem *uživatel* spojena jednou vazbou k případu **C9** (analyzovat dávku). Druhý výřez diagramu je zobrazen na obrázku 5.2. Pro modelaci jsem využil



Obrázek 5.2: Výřez digramu užití zobrazuje aktéra uživatel a strukturu případů jdoucích skrze případ **C9** (analyzovat dávku) (Kompletní diagram je dostupný v příloze B.1)

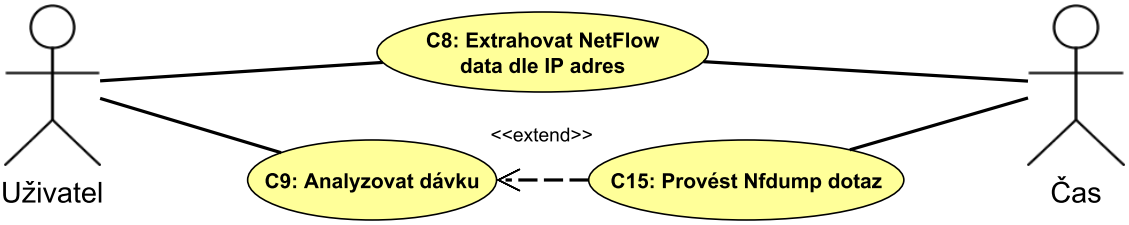
stereotypů «`include`» a «`extend`». Případ užití **C10** (vybrat dávku) je spojen s případem užití **C9** vazbou se stereotypem `include`, který vyjadřuje nutnost spuštění případu **C10** před provedením případu **C9**. Konkrétně uživatel nejprve vybere z již dříve vytvořených dávek tu, kterou chce analyzovat a spustí případ užití **C9**. Naopak stereotyp `extend` slouží k navázání volitelných případů užití. V tomto případě se jedná o případy: **C11** (vyhledat v systému Warden), **C13** (použít systém Nemea) a **C15** (Provést nfdump dotaz). Korektní anotaci lze tedy provést například pouze s použitím nástroje `nfdump`, kdy si uživatel ručně projde zachycená data, nalezne anomálii a vytvoří záznam o události. Použití systému Nemea (případ **C13**) se dále dělí na tři případy, kdy pro plnohodnotné ovládání systému je třeba aktivovat případy užití v tomto pořadí: **C12**, **C14** a **C16**.

Analýza dávky je jednou z nejdůležitějších částí celé aplikace. Uživatel se zde dostává do styku se zachycenými daty, systémem Warden a detektory síťových útoků ze systému Nemea. Může se tedy dozvědět spoustu informací a na jejich základě vytvořit záznam o událostech nalezených v datech.

Případy užití závislé na čase

Zde se budu věnovat případům užití určeným aktérovi *Čas*. Aktér je bude provádět opakovaně v určitou dobu nezávisle na uživateli. Obrázek 5.3 pro úplnost zobrazuje oba aktéry a jejich společné případy užití. Případ užití **C15** (provést Nfdump dotaz) slouží k provádění `nfdump` dotazů a aktér *Čas* jej pravidelně aktivuje pro načtení dlouhotrvajících dotazů do paměti. Takové dotazy bude možné spouštět průběžně a uživatel potom při práci s aplikací nebude muset čekat na výstup i několik desítek vteřin. Výstup každého dotazu se totiž bude ukládat do paměti a při požadavku na stejný dotaz od aktéra *Uživatel* bude vrácen výstup dotazu přímo z paměti. Uživateli bude umožněno dodatečné definování dotazů, které budou určeny pro automatické načtení aktérem *Čas*. Takto nově definované dotazy budou načteny pro každou dávku. Pokud se změní vstupní datová sada, bude možné si od aktéra *Čas* znovu vyžádat načtení těchto dotazů. Předpokládám, že popsaná funkcionalita urychlí práci s aplikací.

Jsem si však vědom toho, že při dohledávání konkrétní události bude uživatel často používat filtry, které nepůjde načítat dopředu. Proto jsem v rámci modelování přidal případ užití **C8** (extrahovat NetFlow data dle IP adres), který by měl tento problém z větší části



Obrázek 5.3: Výřez digramu užití zobrazuje společné případy užití aktéra *Uživatel* a aktéra *Čas*. (Kompletní diagram je dostupný v příloze B.1)

vyřešit. Extrakcí NetFlow dat dle IP adres je myšleno vytvoření dodatečného souboru obsahujícího jen část z celkového provozu. Toho lze docílit výběrem toků, které obsahují ve zdrojové nebo cílové IP adrese některou z předem vybraných adres. Tento dodatečný soubor by měl práci s programem `nfdump` opět výrazně urychlit, jelikož bude oproti originálu několikanásobně menší. Během implementace a testování teprve zjistím, jakým způsobem by bylo možné takovou operaci realizovat.

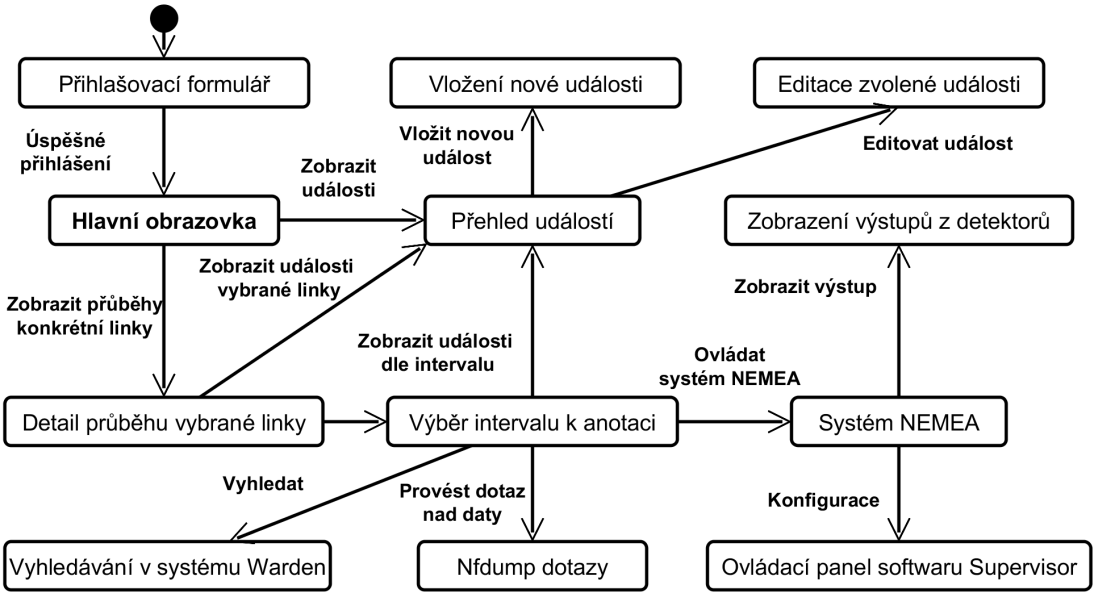
5.2 Návrh Entity-Relationship diagramu

Návrh aplikace se neobejde bez konceptuálního modelu zvaného Entity-Relationship diagram (zkráceně ER diagram). ER diagram není součástí standardu jazyka UML, poprvé jej ve své práci *The Entity-Relationship Model-Toward a Unified View of Data* představil Peter Pin-Shan Chen v roce 1976 [5]. Diagram slouží k zobrazení vztahů mezi entitními množinami. Každá entitní množina obsahuje prvky (entity) s jednoznačným identifikátorem v rámci množiny a sadou atributů. Prvky jsou data reálného světa, která jsou pro účely aplikace podstatná a bude třeba vyjádřit vztahy mezi nimi. Následující odstavce budou popisovat vybrané části diagramu, které jsou v rámci modelování zajímavé. Kompletní ER diagram je uveden v příloze B.2.

Entitní množina `Event` zahrnuje všechny anotované události. Každá entita obsahuje atributy dle formátu IDEA specifikovaného v kapitole 3.3. Značkou `«PK»` je označen jednoznačný identifikátor, tzv. primární klíč (angl. primary key). Každá entita `Event` může zahrnovat libovolné množství entit z množiny `SourceTarget` představující zdrojové nebo cílové prvky. Dle formátu IDEA je umožněno ke každé události připojit více zdrojových nebo cílových IP adres a portů. Na entitní množinu `SourceTarget` je navázána množina `Nfdump`, která daný zdroj nebo cíl doprovází. Výhoda této množiny bude při zpětném dohledávání událostí a dotazů, které vedli k jejich nalezení.

Důležitou částí je entitní množina `User`, která zachycuje uživatele a všechny atributy nutné k plné funkčnosti aplikace. Uživatel musí aplikaci sdělit cesty k NetFlow datům, instalaci systému Nemea, adresáři pro dočasnou paměť (angl. cache) apod. Uživatelé se pak navzájem odlišují nejen unikátním číslem, ale také uživatelským jménem (atribut `username`). Uživatel jako entita je dále odpovědný za vzniklé entity `Event`, `Job` a `NfdumpRequest`. Vložené události však mohou být zpřístupněny všem uživatelům.

Zatím jsem popisoval pouze vazby s kardinalitou `1:N` kdy je vždy jedna entita spojena vazbou s více entitami druhé entitní množiny. Entitní množiny `Job` a `NfdumpRequest` jsou jako jediné spojeny vazbou s kardinalitou `M:N`. Uživatel definuje, které příkazy programu `nfdump` budou provedeny v rámci jedné dávky. Diagram zobrazuje dávky jako entitní množinu `Job`. Každá dávka má jiné časové ohraničení a prováděné příkazy proto budou vracet



Obrázek 5.4: Stavový diagram návaznosti obrazovek

odlišný výstup. Vazba mezi množinami navíc obsahuje atribut `executionTime` sloužící pro zaznamenání doby provádění příkazu. Dle tohoto atributu lze upravit chování aplikace, aby se nespouštělo příliš mnoho dlouhotrvajících dotazů přímo za sebou. To by pak mělo negativní vliv na výkon počítače.

5.3 Grafický návrh uživatelského prostředí

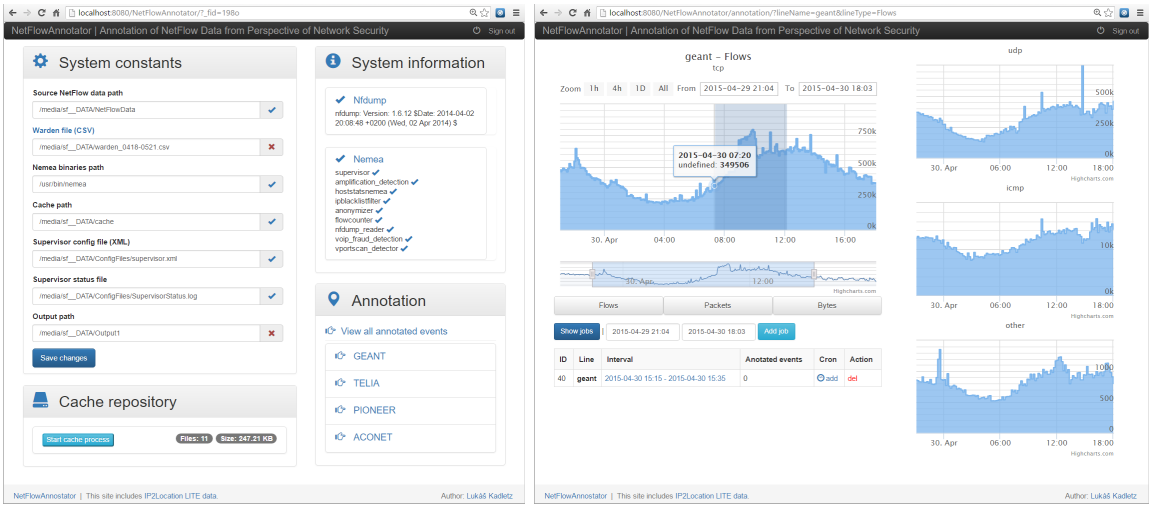
Grafické uživatelské prostředí je důležitou součástí každé aplikace. Webové aplikace nejsou výjimkou a proto se budu návrhem uživatelského rozhraní také zabývat. Je dobré si uvědomit, že uživatel vnímá aplikaci především intuitivně a na to se při návrhu zaměřit. Moje aplikace má potenciál k využití více uživateli a budu s tím v návrhu rozhraní počítat.

Nejprve se pokusím identifikovat potenciálního uživatele a jemu pak grafický návrh přizpůsobit. Předpokládám, že uživatel bude chtít aplikaci použít k anotaci nebo jen k analýze dříve získaných NetFlow záznamů. Měl by mít základní pojem o fungování počítačové sítě. Další požadavky na uživatele jsou:

- Základní znalost systému Nemea a způsob použití detekčních modulů.
- Orientace v síťových útocích a jejich obrazech v NetFlow záznamech.
- Použití nástroje nfdump a znalost základních dotazů.

Analýzou typického uživatele jsem zjistil, že nebude třeba v aplikaci zobrazovat a uvádět podrobnou nápovědu k použití jedné či druhé části. Uživatel se může případně obrátit na dokumentaci, zdrojové kódy aplikace nebo přímo na tuto práci.

Hlavní obrazovka bude uživateli zobrazena po řádném přihlášení. Měla by obsahovat rozcestník pro výběr anotace a nastavení. Jak z použitých nástrojů představených v kapitole 4 plyne, aplikace bude závislá na dalších aplikacích, které poběží samostatně na fyzickém



Obrázek 5.5: Návrh úvodní obrazovky ve frameworku Bootstrap. Úvodní obrazovka vlevo a vpravo zobrazení časových řad.

stroji. Proto bych měl ihned na hlavní obrazovce zobrazit stav těchto aplikací. Uživateli ukáži, která služba je dostupná a která není. Měl bych kontrolovat přítomnost programu `nfdump`, bez kterého nebude větší část aplikace správně fungovat. U systému `Nemea` budu kontrolovat a uživateli zobrazovat jak její přítomnost, tak vypíši konkrétní moduly, které bude možné v aplikaci použít. Dle standardu UML jsem sestavil stavový diagram návaznosti obrazovek celé aplikace, který je uveden na obrázku 5.4. Pro přehlednost jsem do diagramu zahrnul pouze dopředné orientace přechodů. V rámci aplikace lze z libovolné obrazovky přejít na hlavní obrazovku nebo zpět dle zanoření, ve kterém se uživatel nachází.

Framework Bootstrap představený v kapitole 4.2 bude sloužit jako základ grafické stránky aplikace. Velkou výhodou je nativní podpora responzivního designu a velký počet předem vytvořených prvků, které lze využít. Aplikaci poskládám z vhodných prvků doporučených na oficiálních stránkách frameworku. Tímto krokem zjednoduším návrh grafické podoby aplikace a zároveň maximalizuji komfort uživatelů při práci s aplikací. Návrh úvodní obrazovky a obrazovky s časovými průběhy provedený ve frameworku Bootstrap zobrazuje obrázek 5.5.

Kapitola 6

Implementace

Kapitola se zabývá implementací a nasazením aplikace. Cílem není popsat zdrojový kód, ale popsat podstatné implementační detaily. Dokumentace zdrojového kódu je automaticky generovaná pomocí nástroje *Doxygen*[1] a je k dispozici v příloze A ve formátu HTML.

První část kapitoly se zabývá implementací skriptu pro extrakci statistik o zachyceném síťovém provozu pro pozdější zobrazení grafů v aplikaci. Ve druhé části uvedu postup vytvoření relační databáze. Další část kapitoly je věnována webové aplikaci. Představím její strukturu a vysvětlím vybrané implementační detaily. Nakonec se věnuji grafickému prostředí aplikace.

6.1 Extrakce statistik o síťovém provozu

Extrakce statistik zachyceného síťového provozu je potřeba pro vytvoření časových řad, které se budou uživateli zobrazovat ve webové části aplikace. Skript je napsaný v jazyce `Python` a v rámci architektury celého systému jej nazývám `NetFlowAnalyzer`. Nyní objasním důvody pro výběr implementace v jazyce Python. Ještě před realizací webové aplikace jsem implementoval aplikaci pro anotaci v jazyce Python, která sloužila pro anotaci špiček v provozu. Špičky jsem v provozu vyhledával pomocí knihovny `scipy.signal`, vytvářel grafy ve formátu `png` a nabízel je uživateli k anotaci. Již během testování jsem ale postupně zjišťoval, že uživatelské rozhraní není vhodné pro komfortní anotaci dat. Převzal jsem tedy část implementace zabývající se získáváním statistik a začal navrhovat systém pro webovou aplikaci. Výsledný skript včetně původního s detekcí špiček je dostupný v příloze A a lze jej použít zvlášť pro jiné aplikace. Dále se zaměřím pouze na extrakci statistik.

Program `nfdump` umožňuje pomocí přepínače `-I` extrahovat základní statistiky o provozu. Lze získat počet zachycených toků, počet paketů a celkovou velikost přenesených dat. Každý záznam obsahuje zvlášť data dle síťového protokolu `TCP`, `UDP`, `ICMP` a data o ostatním provozu. Statistiky budu sbírat z každého uloženého souboru po pěti minutách. Každý soubor se zachycenými NetFlow daty již tyto statistiky obsahuje a není je nutné počítat znovu. Skript tato data načte a uloží je ve formátu JSON do uživatelem zvoleného adresáře.

Postupně skript vytváří adresářovou strukturu, kde prvním adresářem je název zpracovávané linky, následuje rok a měsíc. Pro každý analyzovaný den provozu je vytvořen jeden soubor a uložen pod názvem `YYYY-MM-DD.json`[2]. Každý soubor je opatřen hlavičkou definu-

[1] Doxygen – Univerzální nástroj pro automatizovanou tvorbu dokumentace vycházející ze zdrojových kódů aplikace. Více na oficiálních webových stránkách: http://www.doxygen.org/

[2] Datový formát – rok: `YYYY` (2015), měsíc: `MM` (04), den: `DD` (24), `2015-04-24`

jící formát dat, která umožňuje pozdější změnu nebo přidání doplňujících dat. Pro rychlejší zpracování jsem každý záznam o pětiminutovém intervalu opatřil přesnou časovou značku bez označení časové zóny. Časová značka je ve webové aplikaci využívána pro zobrazení v grafu. Přiložil jsem kompletní cestu ke zdrojovým datům, aby bylo možné u každého záznamu snadno ověřit, jestli je zdrojový soubor stále k dispozici (provádění dotazů programu `nfdump` apod.).

Experimentálně jsem ověřil, že načtení statistik o provozu během 14ti dní na jedné lince trvá téměř 3 minuty. Využil jsem externí pevný disk zapůjčený sdružením CESNET. Skript je v aplikaci využíván pro načtení statistických údajů o nově vložených souborech vždy pouze jedenkrát. Při následné anotaci se již data načítají z paměti. Do budoucna by bylo možné tuto funkcionalitu implementovat přímo do aplikace, na druhou stranu je nyní možné tento skript využít i v jiných aplikacích.

6.2 Vytvoření relační databáze

Součástí návrhu aplikace uvedeného v kapitole 5.2 je vytvoření Entity-Relationship diagramu (zkráceně ER diagram) zobrazujícího data a jejich vazby. Relační databáze se vytváří transformací ER diagramu a doplněním o další prvky nutné pro chod aplikace. Byla využita databáze MySQL stručně popsaná v kapitole 4.2.

Transformaci ER diagramu lze rozdělit na dvě fáze. Během první fáze provádím transformaci všech vazeb s kardinalitou `1:M` na databázové tabulky obsahující cizí klíče (angl. foreign key), které vyjadřují vazbu s druhou databázovou tabulkou. Ve druhé fázi se zaměřím na jedinou vazbu s kardinalitou `M:N`. Zde je již nutné vytvořit třetí tabulku uchovávající cizí klíče obou tabulek, které se vazby účastní. V následujících odstavcích uvádím příklady transformací entitních množin na databázové tabulky. Ostatní transformace se vytvářejí analogicky. Kompletní schéma vytvořené relační databáze je k dispozici v příloze B.3. Výchozí ER diagram je součástí přílohy B.2.

Transformace vazby s kardinalitou 1:N

Transformace vazby entitní množiny `Event` s množinou `SourceTarget` zahrnuje vytvoření dvou databázových tabulek. Jako první je třeba vytvořit databázovou tabulku `event_idea` obsahující primární klíč `id` (v ER diagramu `EventID`). Tabulka dále obsahuje sloupce dle ER diagramu. Pro co nejefektivnější uložení dat jsem zvolil minimalistické datové typy. Pro časové údaje jsem použil datový typ `timestamp`, který na rozdíl od využití textové reprezentace umožňuje pokročilé SQL dotazy s časovými omezeními. Sloupec `confidence` jsem vyjádřil pomocí výčtového datového typu, jelikož nabývá celkem tří možných hodnot: `yes`, `no` a `unknown`.

Následně vytvořím tabulku `event_source_target` obsahující sloupec `event` sloužící pro uchování cizích klíčů tabulky `event_idea`. Z důvodu použití databázových omezení se nemůže stát, že by se na libovolném řádku tabulky `event_source_target` vyskytl neplatný cizí klíč. U této tabulky jsem pro cizí klíč nastavil akci `CASCADE`, která se automaticky provede při úpravě nebo smazání odkazovaného záznamu v tabulce `event_idea`. Je tak zajištěna referenční integrita dat a zjednodušené mazání záznamů. Stačí smazat záznam v jedné tabulce a všechny řádky ostatních tabulek, které na smazaný řádek odkazují, budou smazány také. Vše zařídí systém SŘBD[3], který je součástí databáze MySQL. Oproti

[3]SŘBD – Systém řízení báze dat

ER diagramu byl přidán sloupec `log_timestamp`, jenž obsahuje časovou značku poslední manipulace se záznamem.

Transformace vazby s kardinalitou M:N

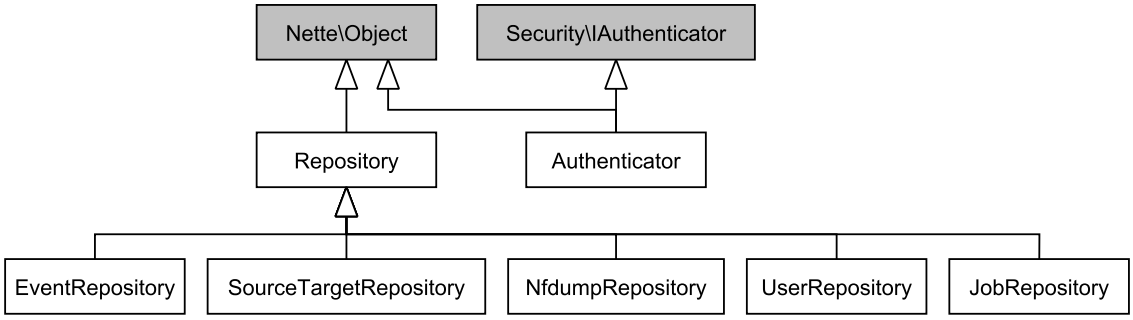
Vazba s kardinalitou `M:N` se v ER diagramu vyskytuje pouze jedenkrát a to mezi entitními množinami `Job` a `NfdumpRequest`. Vazba dokonce obsahuje parametr `executionTime` zachovávající dobu provádění dotazu. Podobně jako v předchozím případě u vazeb s kardinalitou `1:N` budou vazby realizovány pomocí cizího klíče. Navíc zde přibude tabulka, která bude obsahovat cizí klíče a sloupec `executionTime` formátu `float` uchovávající čas v sekundách. Tabulka se jmenuje `nfdump_job`. Nové databázové tabulky `job` a `nfdump_preprocessing` jsou opatřeny všemi sloupci dle ER diagramu a navíc sloupec `log_timestamp` uchovávající poslední manipulaci s řádkem tabulky. Nyní je umožněno k jedné dávce (job) přiřadit několik provedených dotazů programu `nfdump`. V databázi uložen pouze univerzální popis dotazu formou přípony. Výběr předpony (zdrojových souborů pro zpracování) se provede dle dané dávky. Tabulka `nfdump_preprocessing` slouží k uchování často prováděných typů dotazů pro jejich automatické načtení bez interakce s uživatelem.

6.3 Struktura webové aplikace

Webová aplikace je napsána pomocí *Nette frameworku* blíže popsaného v kapitole 4.2. Kompletní balíček s frameworkem obsahuje adresář s názvem *sandbox*, jedná se o minimalizovanou verzi aplikace obsahující doporučenou adresářovou strukturu. Pro účely aplikace není třeba strukturu měnit. Adresářová struktura vytvořené aplikace:

- `/activeLibs` – Obsahuje knihovny, které nelze k projektu připojit skrze rozhraní `composer`[4]. Jsou to knihovny: `ParseCSV` a `IP2Location`.
- `/app` – Adresář obsahující samotnou aplikaci a všechny její zdrojové kódy psané ve skriptovacím jazyce PHP.
  - `/config` – Uchovává konfigurační soubory pro aplikaci nasazenou v produkčním i vývojovém módu.
  - `/exceptions` – Třídy obstarávající práci s výjimkami.
  - `/lkadletz` – Vlastní jádro aplikace, komponenty a podpůrné třídy.
  - `/model` – Vrstva zprostředkovávající komunikaci s relační databází.
  - `/presenters` – Vizuální podoba aplikace, HTML kód a šablony systému Latte.
  - `/router` – Obsahuje továrnu pro definici vlastní URL.
  - `bootstrap.php` – Zde jsou definovány cesty k důležitým adresářům a zavádí se jádro aplikace.
- `/libs` – Knihovny, které lze načíst skrze rozhraní `composer`.
- `/log` – Důležitý adresář z hlediska vývoje a udržovatelnosti aplikace. Zde se uchovávají veškeré výjimky, které během provozu aplikace nastanou. Výjimky jsou uloženy formou HTML dokumentu a navíc zvlášť ve formě textu zapisovány do souborů `exception.log` a `error.log`

[4]Composer – Nástroj pro správu knihoven a ostatních závislostí pro aplikace psané v jazyce PHP. https://getcomposer.org/



Obrázek 6.1: Diagram tříd zobrazující modelovou vrstvu aplikace.

- `/temp` – Obsahuje dočasné soubory a dopředu zkompilované šablony. Šablonový systém `Latte` umožňuje psát šablony ve zjednodušené syntaxi, ale je nutné je následně přeložit. Při nasazení aplikace je první spuštění zpomaleno právě o fázi kompilace a naplnění tohoto adresáře.
- `/vendor` – Zde jsou uchovány všechny zdrojové texty celého Nette frameworku.
- `/www` – Jedná se o veřejný adresář. Zde je kořen aplikace a všechny její viditelné podadresáře. Obsahují obrázky, fonty a doplňující skripty v jazycích Javascript a CSS.
- `composer.json` – Konfigurační soubor aplikace `composer` umožňuje spravovat závislosti aplikace na konkrétních verzích knihoven. Program `composer` je šířen jako svobodný software pod licencí MIT.

Program `NetFlowAnalyzer` představený v kapitole 6.1 je umístěn v adresáři `/libs` a je jako jediný napsán v jazyce Python, a proto jej nelze automaticky zavést pomocí souboru `bootstrap.php`. Tento skript musí být v systému nastaven jako spustitelný.

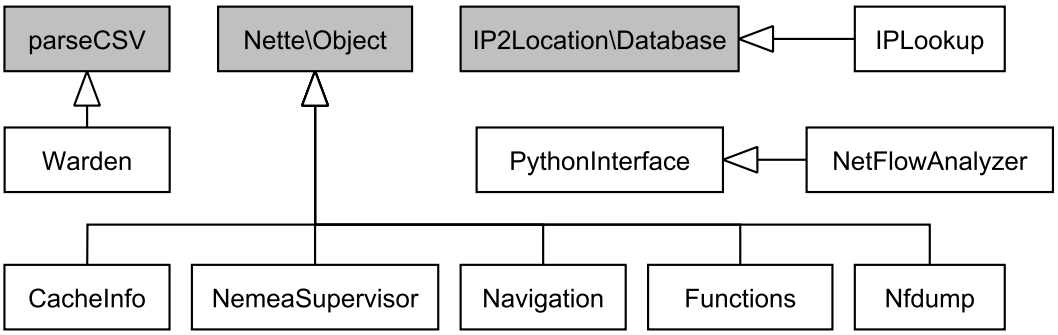
6.4 Implementační detaily vybraných částí aplikace

Následují vybrané části aplikace, které jsou určitým způsobem významné. Soustředím se na ty detaily, jež přispívají k naplnění tématu práce. Nette framework staví na architektonickém modelu MVP. Následující odstavce jsou rozčleněny dle vrstev tohoto modelu.

Modelová vrstva

Modelová vrstva zabezpečuje komunikaci s databází. Celá implementace se nachází v adresáři `/app/model` a obsahuje celkem sedm tříd. Všechny jsou spolu se svými závislostmi zobrazeny v diagramu tříd na obrázku 6.1. Hlavní je třída `Repository`, jež jsem implementoval společné chování pro všechny ostatní třídy, které od ní dědí. Jedná se hlavně o metody: `findBy(), findByID(), insert() a deleteByID()`. Třída `Authentificator` slouží pouze k autentizaci uživatele. Aplikace je sice navržena pro více uživatelů, ale tato funkce není zcela implementována. Pro víceuživatelskou aplikaci bude třeba požádat o autentizaci skrze portál eduID.cz[5], jelikož nejpravděpodobnější je využití studenty případně pracovníky sdružení CESNET, nikoliv široké veřejnosti. Prozatím tedy aplikace obsahuje jen jednoho uživatele, který je po spuštění automaticky přihlášen odesláním nabídnutého

[5]EduID.cz – Česká akademická federace identit, https://eduid.cz/



Obrázek 6.2: Diagram tříd zobrazující podpůrnou část aplikace.

formuláře. Ostatní třídy mají název dle obsluhované databázové tabulky, případně je přetížena metoda `getTable()`, která vrátí aktuální název tabulky se kterou aktuální instance třídy pracuje.

Třída `EventRepository` zabezpečuje práci s událostmi o anotacích. Implementoval jsem zvlášť metodu pro přidávání a aktualizaci událostí včetně definice datových typů parametrů. Jsou to metody `addEvent()` a `updateEvent()`. Abych zachoval logiku databázových operací v modelu, implementoval jsem metodu `findByForm()`, které předám načtené a ošetřené hodnoty a ona vrátí objekt `IRow`. Tento objekt lze snadno procházet pomocí iterátoru a vypsat všechny události dle zvoleného filtru. Výhodou je, že v presenteru stačí zavolat jednu metodu objektu a není třeba uvažovat nad podmínkami pro filtraci. Poslední vybranou metodou je `getEventTags`, která nepracuje se základní tabulkou `event_idea` ale s tabulkou `event_tag` a vrací druhy událostí ve formátu, jenž je vhodný pro formulářový prvek `select`.

Třída `JobRepository` zajišťuje práci s dávkami. Pomocí několika metod začínajících na slovo `set` lze manipulovat s požadavky o načtení dotazů programu `nfdump` a systému Warden. Metoda `getFilesByJobID` umožňuje získat pole obsahující cesty k souborům, které se dané dávky týkají. Výstup lze využít při generování seznamu parametrů pro program `nfdump_reader`, který je součástí systému Nemea.

Prezentační vrstva

Uživatel předává své příkazy tzv. *Presenteru*. Všechny `presentery` dědí od hlavního jménem `BasePresenter`, který zpřístupňuje modelovou vrstvu pomocí komentářového makra `@inject`. Využil jsem možnosti mít v každém `presenteru` instanci potřebného modelového objektu. `BasePresenter` dále vytváří instance objektů: `FileStorage`, `Warden`, `IPlookup` a `NetFlowAnalzyer`. Aplikace je dále sestavena z jednotlivých komponentů. Každá z nich může být nezávisle použita na různých místech. Všechny komponenty jsou spolu s podpůrnými třídami implementovány v adresáři `/app/lkadletz`. Diagram podpůrných tříd je zobrazen na obrázku 6.2.

`NetFlowAnalyzer`

Třída `NetFlowAnalyzer` je odpovědná za komunikaci se stejnojmenným skriptem napsaným v jazyce Python. Metoda `extractData()` spouští skript, který provede načtení statistik o provozu dle zachycených NetFlow záznamů a uloží výsledek do paměti. Ihned po načtení statistik je vytvořen kontrolní soubor, který obsahuje informace o průběhu operace. Při příš-

tím přístupu se přítomnost souboru kontroluje a pokud není v adresářové struktuře nalezen, provede se získání statistik znovu. O načtení dat z paměti se stará metoda `getJsonData()`, jež dle parametrů vrátí hodnoty ve správném formátu pro zobrazení. Zobrazení již provede knihovna Highcharts.

Rozhraní pro spouštění skriptů v jazyce Python je pro možnou budoucí rozšiřitelnost implementováno v nadřazené třídě `PythonInterface`. Metoda `execPython()` vrací naformátovaný výstup skriptu včetně návratové hodnoty. Využívám zde vestavěnou funkci `exec()`, která umožňuje provedení příkazu ve virtuálním terminálu přímo na serveru. Důležitou metodou je také `checkConnection()` ověřující přítomnost a spustitelnost skriptu.

`Warden`

Data ze systému Warden jsou uchována v souboru ve formátu CSV. Třída `Warden` zajišťuje získání dat co efektivní cestou. Ihned po instanciaci je nutné pomocí metody `setIStorage()` nastavit úložiště pro dočasné soubory. Metoda `parseDataInterval()` tuto paměť využívá pro uložení výsledku operace. Pokud není požadovaný výstup dostupný v paměti, je třeba jej získat. Nejprve pomocí nástroje `grep` výrazně zmenším velikost dat ke zpracování, poté provedu syntaktickou analýzu a získám potřebná data. Dočasná paměť pracuje na principu hašovací tabulky, kdy nastavuji pro jednotlivé druhy záznamů zvláštní jmenný prostor. Pro vyčištění paměti stačí odebrat adresář s názvem jmenného prostoru. Abych předešel problémům s časovými značkami nahlášených bezpečnostních událostí v systému Warden, ošetřuji vstupní požadavky metodou `aplySafeInterval()`. Metoda vrátí data v rozsahu o přibližně dvě hodiny širším z každé strany. Implementované opatření má negativní vliv na výkon aplikace, ale zajišťuje lepší informovanost uživatele o detekovaných událostech. Interval lze snížit až na několik minut, pokud budou časové značky v systému Warden lépe odpovídat zachyceným datům. Další metoda `getIPArrayByInterval()` slouží pro získání unikátních IP adres v rámci zvoleného intervalu. Seznam IP adres dále využívá třída `Nfdump`.

`Nfdump`

Třída `Nfdump` zajišťuje správné provedení příkazů programu `nfdump`. Podobně jako u třídy `Warden` využívám ukládání výstupu do dočasné paměti ve zvláštním jmenném prostoru. Každý provedený dotaz uložím do paměti a při příštím dotazu jej z paměti opět načtu. Existují dva základní typy dotazů: *TopN* a *ListFlows*. Dotaz typu TopN počítá statistiky a vrátí seřazené výstupy dle zvoleného parametru (např.: dle celkového počtu toků na jednu IP adresu, celkového množství přenesených dat, apod.). Dotaz typu ListFlows vrací libovolně agregované toky s možností řadit dle jejich časové značky. Třída zajišťuje i vytvoření souboru obsahující pouze vybraný síťový provoz. Tato funkcionalita má výrazný vliv na výkon při vytváření statistik u velkých zdrojových souborů.

Metoda `convertor()` slouží k analýze výstupu a doplnění HTML značek pro pozdější zpracování pomocí JavaScriptu. Pomocí regulárního výrazu vyhledávám IP adresy a porty a doplňuji meta informace. JavaScript na pohledové vrstvě poté způsobí označení IP adres a přidá možnost po kliknutí získat podrobné informace (DNS záznam, informace o lokalizaci, výskyt v systému Warden). Konverze je prováděna vždy a bylo by vhodné ukládat některá data do dočasné paměti včetně těchto informací.

`IPLookup`

Jedná se o třídu pro získání doplňujících údajů o IP adrese. Nadřazená třída `Database` je implementována v knihovně `IP2Location`. Zabezpečuje získání lokalizačních údajů z binární databáze uložené v souboru `IP2LOCATION-LITE-DB1.BIN`. Pomocí vestavěných funkcí `gethostbyname()` a `gethostbyaddr()` je možno získat záznamy ze systému DNS. Vše je ukládáno pod zvláštní jmenný prostor do dočasné paměti, aby nedocházelo k opakovaným dotazům.

`NemeaSupervisor`

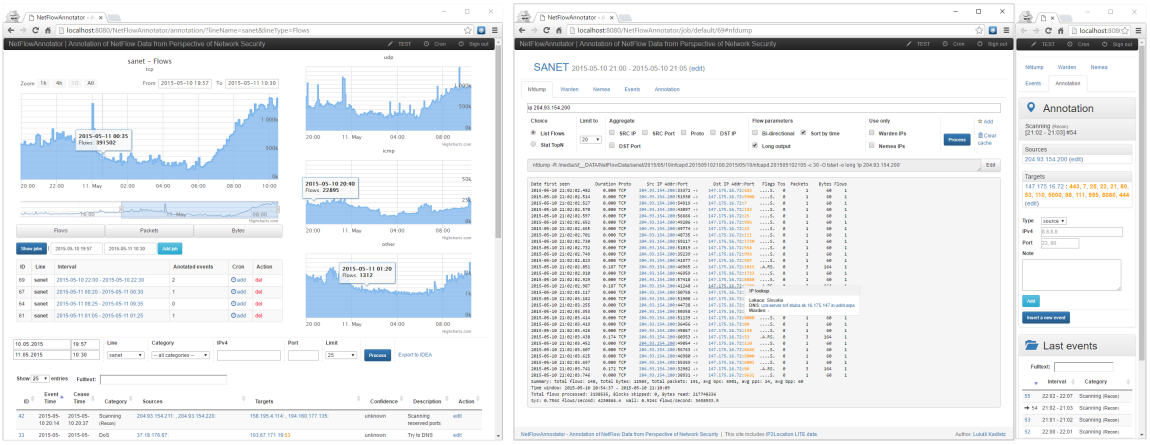
Systém Nemea spravuje třída `NemeaSupervisor`, která primárně pracuje s konfiguračním souborem programu `Supervisor`. Ze souboru je třída schopna získat informace o aktuálních modulech, jejich propojeních a aktivitě. Změnou souboru dojde po opětovném načtení ke změně parametrů spuštěných modulů. Metoda `getStats()` získává statistiky o modulech (vytížení procesoru, alokovaná paměť apod.) a ty zobrazuje uživateli. Aby bylo možné používat jakékoliv detektory, je třeba nejprve inicializovat konfigurační soubor a vložit do něj všechny dostupné moduly. Metoda `addNfdumpReader()` poté pro zvolený modul přidá program `nfdump_reader` a nastaví jeho parametry dle analyzované dávky. Po spuštění celého řetězce proběhne detekce a výstup se zapíše do souboru dle konfigurace. Metoda `getOutputFiles()` získá z konfiguračního souboru cestu k výstupním souborům. Soubory lze nejen číst, ale i mazat jejich obsah.

Nebylo možné provést opětovné načtení konfiguračního souboru programu `Supervisor` prostřednictvím spuštění skriptu přímo z aplikace. Pravděpodobně vinou používaného virtuálního terminálu vestavěnou funkcí `exec()`. Problém jsem vyřešil pravidelným spouštěním skriptu pomocí linuxového nástroje `Cron`. Každou minutu se znovu načte konfigurace, pokud se nic nezměnilo, operace proběhne velmi rychle a bude to pro procesor serveru zanedbatelná zátěž. Skript je součástí přílohy A.

Komponenty

Všechny komponenty jsou samostatnou jednotkou obsahující vrstvu prezentační i pohledovou. Každá komponenta má k sobě přiřazenou šablonu (případně více šablon). Výhodou je její použitelnost na libovolném místě v aplikaci. Mohu tedy snadno změnit grafický vzhled nebo logické uspořádání a stačí pouze vybrat vhodný `presenter` pro vložení komponenty. Přenosnost mezi aplikacemi není ve většině případů možná, protože komponenty jsou navázány na množství zde implementovaných tříd.

Formulářové komponenty obsahují vždy šablonu s vizuálním stylem formuláře, jeho definici a metodu pro zpracování požadavku. Součástí je většinou i zobrazení výsledků ve formě tabuly (Warden, IPLookup) nebo neformátovaného textu (Nfdump). Formuláře jsou definovány včetně základních validací, a proto jsem si při zpracování jist jejich správností. Bylo by vhodné na straně aplikace ještě více omezit vstupní hodnoty například pomocí regulárních výrazů. Některé moderní webové prohlížeče umožňují validovat formulářové pole dle regulárního výrazu a tím uživatele včas upozornit na chybu. Veškeré zpracování formulářů probíhá asynchronně pomocí přístupu `AJAJ` představeného spolu s Nette frameworkem v kapitole 4.2. Tento přístup výrazně urychlil práci s aplikací, protože nebylo nutné celý obsah znovu vykreslit, ale stačilo jen změnit určité HTML elementy na stránce. Během čekání na odpověď zobrazuji uživateli točící se kolečko umístěné buď vedle aktivního tlačítka



Obrázek 6.3: Příklad práce s aplikací. Vlevo je zobrazen postup výběru intervalu k anotaci. Vpravo se nachází ukázka práce s programem `nfdump` a anotace nalezeného útoku. (Ostatní snímky aplikace naleznete v příloze C)

nebo místo aktivního odkazu. Je to prvek grafického uživatelského prostředí, který zvyšuje přehled o probíhajících procesech na pozadí.

Pohledová vrstva

Pohledová vrstva obsahuje samotné kódování ve značkovacím jazyce HTML s podporou kaskádových stylů CSS. Součástí jsou i skripty v jazyce Javascript. Skripty jsou spolu s kaskádovými styly uloženy v adresáři `/www`. Většina skriptů je načítána v hlavním šablonovacím souboru `/app/template/@layout.latte`, který zároveň obsahuje kostru celé aplikace. Šablonovací systém Latte (součást Nette frameworku) umožňuje pohodlně kódovat jednotlivé stránky s důrazem na přehlednost zdrojového kódu. Šablony jsou před použitím přeloženy do tvaru obsahující PHP značky včetně důležitého escapování[6]

Většina implementovaných komponent obsahuje vlastní pohledovou vrstvu (šablonu). Nedochází proto k vícenásobnému použití stejného zdrojového kódu. Komponenta se stává univerzálnější a lze ji použít v libovolné části aplikace. Některé komponenty dokonce i v jiných aplikacích.

6.5 Grafické uživatelské prostředí

Implementace grafického uživatelského prostředí probíhala postupným skládáním funkčních prvků frameworku Bootstrap. Framework obsahuje mnoho předdefinovaných elementů a jelikož nejsem schopen vytvořit dostatečně kvalitní design, z větší části jsem na nich postavil grafický vzhled aplikace. Na obrázku 6.3 je zobrazen příklad práce s aplikací. Uživatel si vybírá z časových řad intervaly, které dále analyzuje a provádí anotace. V levé části obrázku jsou umístěny dvě okna. První slouží k zobrazování výstupů programu `nfdump` a druhé k provádění anotace.

Abych vyšel vstříc většině uživatelů, zvolil jsem vodorovné záložky oddělující funkcionalitu v rámci analýzy dávky. Každý panel však lze otevřít v novém okně a díky responzivnímu

[6]Escapování – Předřazení speciálního znaku použitého v daném jazyce speciálním symbolem. Obrana před útoky typu XSS, SQL Injection apod.

designu aplikace zobrazit v libovolné šířce kdekoliv na pracovní ploše. Zúžený tvar stránky zapříčiní skrytí nadbytečných elementů a zobrazení téměř čistého pracovního panelu.

Součástí přílohy C je několik snímků aplikace v různých fázích anotace. Zachytil jsem detaily uživatelského prostředí, které nemusejí být na první pohled zřejmé. Následuje výčet doplňkových funkcí grafického uživatelského prostředí:

- Každý formulář lze odeslat stisknutím klávesy ENTER a to vždy asynchronně s využitím přístupu AJAJ popsaného v kapitole 4.2.
- Většina funkčních odkazů opět využívá přístupu AJAJ a po kliknutí se transformují na rotující kolečko indikující provádění operace.
- Přepínání mezi panely u analýzy dávky je možné pomocí klávesové zkratky CTRL + šipky.
- Kliknutím na libovolnou IP adresu v rámci celé aplikace je vyvoláno malé okno, které zobrazuje lokaci, DNS překlad a seznam hlášení v systému Warden týkajících se dané IP adresy.
- Libovolný port zobrazený v aplikaci je zbarven oranžově, pokud se jedná o rezervovaný port nebo standardní port pro existující aplikaci.
- Pokud je IP adresa zobrazena ve správě anotací nebo výstupu programu `nfdump` a nachází se v hlášení systému Warden, je automaticky podbarvena žlutou barvou.
- Práce se systémem Nemea probíhá skrze kliknutí na názvy modulů. Kliknutím lze vyžádat zapnutí nebo vypnutí modulu, které zajistí program `Supervisor` pomocí opětovného načtení upravené konfigurace.
- Při prohlížení anotovaných událostí lze snadno některou z nich opravit kliknutím na odkaz s názvem `edit`. Kliknutí vyvolá zobrazení editačního režimu v rámci stávajícího okna aplikace. Není nutné opouštět stávající obrazovku.
- Každá zobrazená tabulka obsahuje prvky pro snadné vyhledání pomocí formulářového pole s názvem `fulltext`. Je možné vyhledat libovolný řetězec napříč všemi sloupci tabulky a to bez načítání celé stránky. Vyhledávání se děje na straně webového prohlížeče pomocí jazyka JavaScript.

6.6 Nasazení aplikace

Tato podkapitola je věnována popisu správného nasazení aplikace na stávající server. Aplikace má několik specifických požadavků na adresářovou strukturu a formát vstupních dat, které v následujících odstavcích vysvětlím. Použité nástroje a technologie jsou popsány v kapitole 4 a předpokládám, že server má všechny tyto nástroje správně nainstalovány a dále se o nich nebudu zmiňovat.

Uživatel má možnost přímo z hlavní obrazovky webové aplikace nastavit potřebné konstanty. Vždy se jedná o cesty v rámci serveru k různým objektům. Jde například o zdrojové NetFlow záznamy, export ze systému Warden, instalační adresář systému Nemea, adresář pro uložení dočasné paměti a další. Aplikace uživatele upozorní, pokud některá z konstant nebude obsahovat správnou cestu k danému objektu. Použití aplikace je uživateli umožněno až po vyplnění všech povinných konstant.

Vstupní NetFlow záznamy musí dodržovat jednotnou adresářovou strukturu a název souborů. Adresáře musí vyjadřovat datum uložených záznamů. Název souboru ve formátu `nfcapd.201504152350` určuje přesný datum a čas, kdy je pomocí číslic vyjádřen rok, měsíc, den, hodina a minuta vytvoření záznamu. Příklad adresářové struktury pro linku s názvem `aconet` a souborů zachycených v rozmezí od 15. dubna 23:50 do 16. dubna 13:30 roku 2015:

```
aconet/2015/04/15/nfcapd.201504152350
aconet/2015/04/15/nfcapd.201504152355
aconet/2015/04/15/...
aconet/2015/04/16/nfcapd.201504160000
aconet/2015/04/16/...
aconet/2015/04/16/nfcapd.201504161330
```

Pro správnou funkci programu `Supervisor` lze využít předpřipravený skript dostupný v přiloženém DVD (viz. příloha A) pod názvem `NetflowAnnotator-cron.sh`. Skript je nutné použít dle informací obsažených na jeho prvních řádcích. Funkci pravidelného načítání dotazů programu `nfdump` zajišťuje skript `NetFlowAnnotator-web.sh`, který lze spouštět pravidelně, například každých 10 minut nebo i méně.

Kapitola 7

Testování a tvorba datové sady

Cílem této kapitoly je představit průběh a výsledky testování aplikace. Tvorba samotné datové sady je velmi náročný proces, který závisí na znalostech uživatele a na výkonu použitého počítače. Dostupná data jsem nedokázal plně využít z mnoha důvodů (velikost, náročnost zpracování, požadavky na operační paměť a výkon procesoru atd.). Vybral jsem jen část dat, která obsahuje provoz různých dnů v týdnu a různých časových úseků.

7.1 Testování aplikace

Aplikaci jsem otestoval při vytváření anotované sady. Pracovním strojem byl můj osobní počítač osazený procesorem Intel® Core™ i5-4300U @ 1.90 GHz a SSD diskem. Aplikace byla nainstalována na virtuálním stroji se systémem Debian Linux (Ubuntu 15.10) a zpřístupněna pomocí předání portů do hostitelského operačním systému Windows®. Zvolil jsem testovací sadu o velikosti přibližně 55 GB obsahující data všech 8 dostupných linek. Tabulka 7.1 zobrazuje časové intervaly a velikost dat dle každé z linek. Celkovou velikost dat nebylo možné více omezit, jelikož bych poté neměl dostatečně široký interval pro analýzu. Snažil jsem se zvolit optimální velikost dat pro reálnou anotaci. Při volbě většího intervalu bych nebyl schopen v přiměřeném čase anotaci provést. Problémem je skutečnost, že některé pětiminutové intervaly mají velikost i více než 150 MB a provedení dotazů programu `nfdump` na širším intervalu je výpočetně velmi náročné.

Otestoval jsem grafickou stránku aplikace a návaznosti obrazovek dle stavového dia-

Linka	**Začátek**	**Konec**	**Velikost [MB]**
`Aconet`	24.4. 15:00	24.4. 21:00	6 264
`Amsix`	18.4. 00:00	18.4. 07:00	7 949
`Geant`	29.4. 20:00	30.4. 20:00	7 147
`Nix2`	25.4. 15:50	26.4. 12:00	2 440
`Nix3`	27.4. 20:00	27.4. 23:55	6 777
`Pioneer`	6.5. 08:00	6.5. 23:55	6 328
`Sanet`	10.5. 10:35	11.5. 10:30	9 483
`Telia`	18.4. 00:00	18.4. 06:00	9 541
Celkem	**4 dny, 10 hodin, 55 minut**		**55 929**

Tabulka 7.1: Zvolené intervaly zachyceného provozu pro účely testování a vytváření anotované datové sady.

gramu z kapitoly 5.3. Během procházení aplikace byly odezvy nízké. Práce s grafy zachyceného provozu se začala zpomalovat až při načtení dvou a více týdnů provozu. Časové řady jsou pro orientaci důležité. Nepodařilo se mi však do časových řad vložit označení hlášení ze systému Warden. Především z důvodu velkého množství hlášení, které nemusí v dostupných datech existovat. Systém Warden sbírá data o bezpečnostních událostech napříč celou sítí CESNET2 a dalšími.

Při vytváření anotované sady jsem nejvíce využíval systém Nemea. Označil jsem si interval provozu k anotaci a spustil detektory, které mi po delší době nahlásili seznam IP adres účastnících se nějakého útoku. Ručně jsem poté přešel do pokládání dotazů programu `nfdump` a každou IP adresu dohledal a přesně anotoval. Často se stávalo, že IP adresy opravdu útočili, ale jejich útok nebyl v systému Warden nahlášen. Důvodem je použití detektoru, který v době získání dat (květen 2015) nebyl implementován nebo nedokázal takový útok nalézt. Využití stávajících detektorů je jistě výhodou a dokázal jsem tak anotovat mnohem více útoků než za použití systému Warden. Například detektor `vportscan_detector` dokázal odhalit množství vertikálních skenování v síti za minimální čas a snadno jsem provoz dohledal v datech a provedl anotaci.

V průběhu testování jsem narazil na několik míst, kde má aplikace delší odezvy. Je to způsobeno tím, že virtuální stroj sdílí se systémem jedno fyzické jádro a 4 GB paměti RAM. Při větším množství dotazů není server schopen odbavovat dotazy paralelně a uživatel musí na odpověď serveru počkat. V kapitole 8.1 tento jev připomínám a uvádím budoucí možnosti řešení.

7.2 Anotovaná datová sada

Datová sada je složena z NetFlow záznamů, souboru obsahující exportované události ve formátu IDEA a souboru obsahující data relační databáze ve formátu SQL. NetFlow záznamy jsou uloženy v adresáři `NetFlow` ve formátu, který podporuje aplikace. Soubor `IDEA.json` obsahuje anotace událostí ve formátu IDEA a soubor `Netflow_annotator_database.sql` kompletní stav databáze v době exportu událostí. Pokud bude uživatel chtít pracovat s událostmi umístěnými v databázi, může je snadno z tohoto souboru extrahovat. Přiložil jsem i soubor `IDEA-all.json` obsahující všechny události, které byly v době vytváření datové sady uloženy v databázi.

Vytvořil jsem anotovanou datovou sadu, která obsahuje celkem **46** vzorových anotací napříč všemi linkami dle tabulky 7.1. U každého nově nalezeného typu útoku jsem provedl několik anotací, které mohou sloužit jako vzor pro další práci s aplikací. Součástí přiloženého DVD je anonymizovaná datová sada o velikosti **2.64 GB** a zahrnující 7 hodin provozu na lince **Sanet**. Linka **Sanet** obsahuje nejméně provozu, ale lze ji v této velikosti uložit na DVD a přiložit přímo k této práci. Obsah DVD je uveden v příloze A.

Kapitola 8

Závěr

Cílem této práce bylo vytvořit nástroj pro offline anotaci NetFlow datové sady. Nastudoval jsem protokol NetFlow včetně funkce kolektoru a exportéru. Analyzoval jsem nejčastější bezpečnostní události a způsoby jejich detekce v NetFlow záznamech. Provedl jsem analýzu manuální anotace a na jejím základě vytvořil koncepční návrh, který obsahuje dva formáty pro uložení anotací. Součástí návrhu je několik UML diagramů vyjadřujících chování aplikace. Využil jsem také stavový diagram pro vyjádření návaznosti obrazovek v grafickém uživatelském prostředí. Prostudoval jsem potřebné technologie pro implementaci moderních webových aplikací. Pro implementaci jsem zvolil jazyk PHP a Python. Aplikace je postavena na *Nette frameworku* s podporou knihovny *jQuery* a frameworkem *Bootstrap*, který zajišťuje responzivní design jednotlivých stránek.

Podařilo se mi vytvořit aplikaci splňující vytyčené cíle. Aplikace umožňuje uživateli provést anotaci nebo analýzu zachycených NetFlow záznamů. Vytvořené anotace lze snadno stáhnout ve formátu IDEA. Aplikace umožňuje zobrazení časových řad analyzovaného provozu dle celkového počtu toků, paketů nebo objemu přenesených dat. Na zvoleném vzorku dat jsem demonstroval funkčnost aplikace. Celkem jsem vytvořil 46 vzorových anotací zahrnujících několik záznamů od každého typu bezpečnostní události. Uživatel je schopen pomocí implementované aplikace vytvořit anotovanou datovou sadu, která může být využita pro testování nově vyvíjených detektorů a algoritmů.

Existující práce se často zabývají různými přístupy pro detekci bezpečnostních incidentů a jejich výstupem jsou funkční detektory. Aplikace může usnadnit práci programátorům těchto detektorů tím, že využijí anotovanou datovou sadu pro ověření správné detekce. Dle výsledků lze upravovat parametry detektorů a dosáhnout tak co nejpřesnější detekce.

8.1 Možnosti dalšího vývoje

Tato podkapitola se věnuje dalším možnostem vývoje aplikace. Během používání aplikace jsem nalezl několik možností pro další vývoj. V této kapitole bych chtěl uvést ty nejzajímavější a nastínit způsob jejich implementace. Popíši postup rozšíření implementace víceuživatelského rozhraní, pro které je tato aplikace navržena. Navrhnu i možnost lépe pracovat s dočasnými soubory uchovávající NetFlow data o omezeném provozu.

Víceuživatelské prostředí a přihlášení skrze EduID

Navrhl jsem aplikaci tak, aby ji mohlo využívat více uživatelů. Tato možnost ovšem není zcela implementována. Pro plnou funkčnost je třeba přidat registraci uživatelů, přihlašovací

formulář a uživatelský profil. Často se setkávám s webovými aplikacemi, které umožňují registraci, ale zrušení již vytvořeného účtu je těžko proveditelné (často jen skrze kontakt na správce). Zde bych chtěl umožnit uživateli jednoduchou registraci pomocí emailu a možnost zrušení účtu.

Jelikož předpokládám, že budou aplikaci využívat primárně studenti, nabízí se možnost využít přihlašovacího systému EduID. Jedná se o systém umožňující studentům a akademickým pracovníkům přistupovat do aplikací třetích stran pomocí svého uživatelského jména a hesla. Studenti by nemuseli provádět registraci, ale přímo by se do aplikace přihlásili. Systém spravuje Česká akademická federace identit. Existují však i další způsoby zjednodušení registrace. Například sociální sítě (Google, Twitter atd.) umožňují sdílení identity s libovolnou aplikací. Uživatel je vždy vyzván k odsouhlasení sdílení informací s konkrétní aplikací a poté je již automaticky přihlášen. Tento způsob je velmi populární, jelikož si uživatel nemusí vytvářet nový profil, ale použije svůj stávající. Pro rozšíření aplikace je implementace některého z výše zmíněných způsobů nutná.

Dočasné soubory s NetFlow daty

V rámci aplikace jsem implementoval vytvoření dočasného souboru obsahující jen část NetFlow dat. Použití tohoto souboru výrazně urychlilo dotazy programu `nfdump`. Při vytváření anotované sady jsem zjistil, že vytvoření souboru trvá dlouhou dobu. Je to z důvodu omezení provozu dle IP adres, které se v daném intervalu vyskytují buď v systému Warden nebo byly odhaleny systémem Nemea. Systém Warden bohužel obsahuje příliš mnoho IP adres, jež proces tvorby dočasného souboru nejvíce zpomalují. Chtěl bych vymyslet způsob, jak by bylo možné vytvořit dočasný soubor rychleji a neomezovat jej pouze na IP adresy ze systému Warden, ale i na adresy zadané uživatelem. Další možností by bylo do omezeného provozu zahrnout i IP adresy, které s danou IP adresou komunikovaly. Bylo by možné snadno zobrazit provoz, ve němž se vyskytují IP adresy do určitého zanoření. Popsané rozšíření by bylo vhodné implementovat především kvůli možnému zvýšení rychlosti filtrace.

Nasazení systému na výkonném serveru

Aplikaci jsem vyvíjel a testoval na svém osobním počítači, který nedisponuje velkým výpočetním výkonem. Chtěl bych ji nasadit na výkonný server, na němž by její odezvy byly výrazně nižší. Bylo by možné ve větší míře využít předem načtených dotazů a povolit větší množství paralelních spojení. Často se totiž stává, že aplikace čeká na provedení náročného dotazu a počítač již není schopen obsloužit další dotazy. Ovládání aplikace se tak stává méně komfortní. Zároveň by bylo třeba vyřešit přístup na server a možnost nahrát si na jeho pevný disk vstupní data, která mohou zabírat i desítky gigabytů.

Grafické uživatelské rozhraní

Bylo by vhodné zaměřit se na uživatelské prostředí a upravit jej dle dlouhodobějších zkušeností z práce s aplikací. Chtěl bych upravit především grafy zobrazující analyzovaný provoz. Existuje možnost, jak přímo do časových řad vložit tzv. vlajky, které by vhodným zbarvením označovaly anotované události. Zvýšil by se komfort při dohledávání událostí a bylo by možné v časových řadách snadno zobrazit určité typy událostí (shodné IP adresy, druhy událostí, porty atd.).

Jelikož se osobně grafikou webových aplikací nezabývám, nejsem schopen grafické prostředí vylepšit. Rád bych předal grafickou stránku aplikace grafikovi, který může aplikace pozměnit k lepšímu výsledku. Ačkoliv je nyní aplikace použitelná, zásah do grafického designu potřebuje.

Literatura

[1] ANONYM. *Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network*. Angel722 Computer Publishing, 1998. ISBN 978-1-575-21268-5.

[2] B. CLAISE, E. : Cisco Systems NetFlow Services Export Version 9 [online]. October 2004 [cit. 2015-11-14]. Dostupné z: <https://www.ietf.org/rfc/rfc3954.txt>

[3] BARTOŠ, V.; ŽÁDNÍK, M.; ČEJKA, T. : Nemea: Framework for stream-wise analysis of network traffic – CESNET Technical Report [online]. září 2013 [cit. 2016-04-08]. Dostupné z: <https://www.cesnet.cz/wp-content/uploads/2014/02/trapnemea.pdf>

[4] CESNET : Výroční zpráva sdružení CESNET 2014 [online]. 2014 [cit. 2016-01-09]. Dostupné z: <https://www.cesnet.cz/sdruzeni/vyrocni-zpravy-sdruzeni-cesnet/>

[5] CHEN, P. P. : The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, roč. 1, 1976: s 9–36.

[6] CISCO : NetFlow Services Solutions Guide [online]. leden 2007 [cit. 2015-11-27]. Dostupné z: <http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/netflow/nfwhite.pdf>

[7] CLAISE, B.; Cisco Systems, I. : Specification of the IP Flow Information Export (IPFIX) Protokol for the Exchange of IP Traffic Flow Information [online]. leden 2008 [cit. 2015-12-09]. Dostupné z: <https://www.ietf.org/rfc/rfc5101.txt>

[8] COLE, E. *Network Security Bible*. 2. Wiley Publishing, Inc., 2009. ISBN 978-0-470-50249-5.

[9] CONVERSE, T.; PARK, J.; MORGAN, C. *PHP5 and MySQL Bible*. 1. Wiley Publishing, Inc., 2004. ISBN 0-7645-5746-7.

[10] CROCKFORD, D. : The application/json Media Type for JavaScript Object Notation (JSON) [online]. červenec 2006 [cit. 2016-04-23]. Dostupné z: <https://www.ietf.org/rfc/rfc4627.txt>

[11] ČEJKA, T.; BARTOŠ, V.; TRUXA, L.; aj. *Intelligent Mechanisms for Network Configuration and Security: 9th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2015, Ghent, Belgium, June 22-25, 2015. Proceedings*. Cham: Springer International Publishing, 2015. Using Application-Aware Flow Monitoring for SIP Fraud Detection. Dostupné z: <http://dx.doi.org/10.1007/978-3-319-20034-7_10>. ISBN 978-3-319-20034-7, 10.1007/978-3-319-20034-7_10.

[12] ČESKO : Zákon č. 181 ze dne 23. července 2014 o kybernetické bezpečnosti a o změně souvisejících zákonů (zákon o kybernetické bezpečnosti). *Sbírka zákonů České republiky*, červenec 2014 [cit. 2015-11-21]: s 4–5. Dostupné z: <https://portal.gov.cz/app/zakony/download?idBiblio=82522&nr=181~2F2014~20Sb.&ft=pdf>

[13] ČESKO : Zákon č. 127 ze dne 22. února 2005 o elektronických komunikacích a o změně některých souvisejících zákonů (zákon o elektronických komunikacích). *Sbírka zákonů České republiky*, únor 2005 [cit. 2016-01-16]: s 110–113. Dostupné z: <https://portal.gov.cz/app/zakony/download?idBiblio=59921&nr=127~2F2005~20Sb.&ft=pdf>

[14] FLANAGAN, D. *jQuery - Pocket Reference: Read Less, Learn More*. O'Reilly, 2011. ISBN 978-1-449-39722-7.

[15] GRUDL, D. : Začínáme s Nette Framework [online]. *Zdrojak.cz*, 2009-03-10 [cit. 2013-12-25], ISSN 1803-5620. Dostupné z: <http://www.zdrojak.cz/serialy/zaciname-s-nette-framework/>

[16] HJELMVIK, E. : Sniffing Tutorial part 1 - Intercepting Network Traffic [online]. březen 2011 [cit. 2015-12-01]. Dostupné z: <http://netres.ec/?b=113EA66>

[17] KLYNE, G.; NEWMAN, C. : Date and Time on the Internet: Timestamps [online]. červenec 2002 [cit. 2016-01-18]. Dostupné z: <https://www.ietf.org/rfc/rfc3339.txt>

[18] KRETCHMAR, J. M.; DOSTÁLEK, L. *Administrace a diagnostika sítí pomocí OpenSource utilit a nástrojů*. 1. Computer Press Brno, 2004. ISBN 80-251-0345-5.

[19] POSTEL, J. : Transmission Control Protokol [online]. září 1981 [cit. 2015-12-01]. Dostupné z: <https://www.ietf.org/rfc/rfc793.txt>

[20] POSTEL, J. : ICMP [online]. září 1981 [cit. 2015-12-07]. Dostupné z: <https://www.ietf.org/rfc/rfc792.txt>

[21] POTEL, M. : MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java [online]. Taligent, Inc. 1996 [cit. 2014-04-12]. Dostupné z: <http://www.wildcrest.com/Potel/Portfolio/mvp.pdf>

[22] ROSENBERG, J.; SCHULZRINE, H.; CAMARILLO, G.; aj. : SIP: Session Initiation Protokol [online]. červen 2002 [cit. 2015-12-09]. Dostupné z: <https://www.ietf.org/rfc/rfc3261.txt>

[23] RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. *The Unified Modeling Language Reference Manual*. 2. Pearson Higher Education, 2004. ISBN 0-321-24562-8.

[24] SHAFRANOVICH, Y. : Common Format and MIME Type for Comma-Separeted Values (CSV) Files [online]. říjen 2005 [cit. 2015-12-08]. Dostupné z: <https://www.ietf.org/rfc/rfc4180.txt>

[25] SPURLOCK, J. *Bootstrap - Responsive Web Development*. O'Reilly, 2013. 128 s. Dostupné z: <http://it-ebooks.info/book/2331/>. ISBN 978-1-44934-391-0.

[26] TATROE, K.; MACINTYRE, P.; LERDORF, R. *Programing PHP*. 3. O'Reilly, 2013. ISBN 978-1-449-39277-2.

[27] WOLTER, R.; CLAISE, B. *Network Management: Accounting and Performance Strategies*. Cisco Press, 2007. ISBN 978-1-58705-198-2.

Přílohy

Seznam příloh

Příloha A

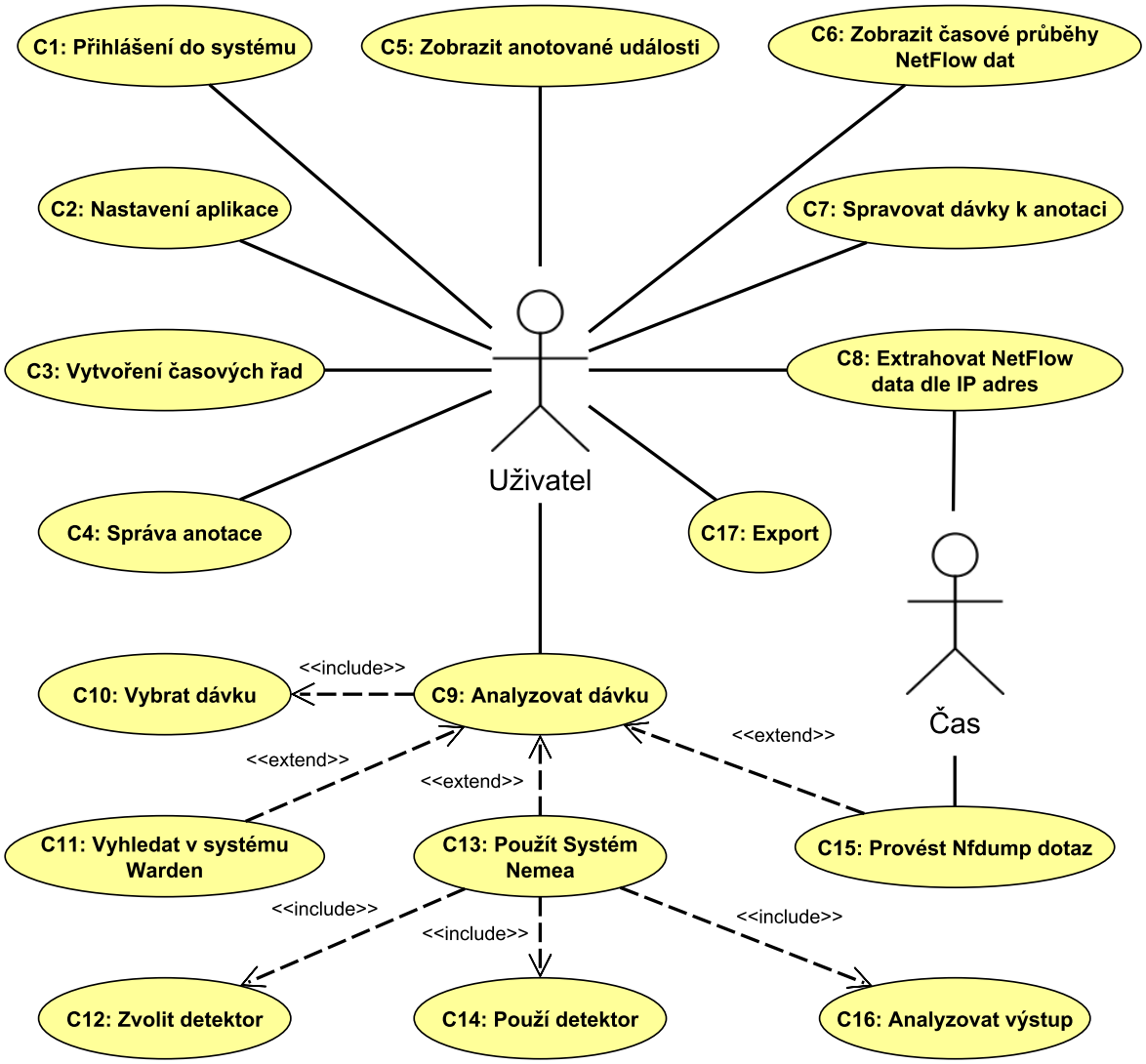
Obsah přiloženého DVD

- `/xkadle29_aplikace/` – zdrojové kódy aplikace, instalační SQL skript, knihovny
- `/xkadle29_dp_zdroj/` – zdrojové kódy práce v programu LaTeX, obrázky, návrhové diagramy a snímky aplikace
- `/xkadle29_NetFlow_Sada/`
  - `NetFlow/` – NetFlow záznamy linky `sanet`
  - `IDEA.json` – anotované události v NetFlow záznamech
  - `IDEA-all.json` – veškeré anotované události napříč všemi linkami
  - `Netflow_annotator_database.sql` – SQL skript zachycující databázi v době exportu dat do formátu IDEA
- `/xkadle29_aplikace.zip` – komprimovaný adresář ve formátu ZIP
- `/xkadle29_dp_zdroj.zip` – komprimovaný adresář ve formátu ZIP
- `/xkadle29_diplomova_prace.pdf` – diplomová práce ve formátu PDF
- `/xkadle29_diplomova_prace_print.pdf` – diplomové práce určená k tisku
- `/xkadle29_readme.txt` – podrobný popis adresářů a souborů

Příloha B

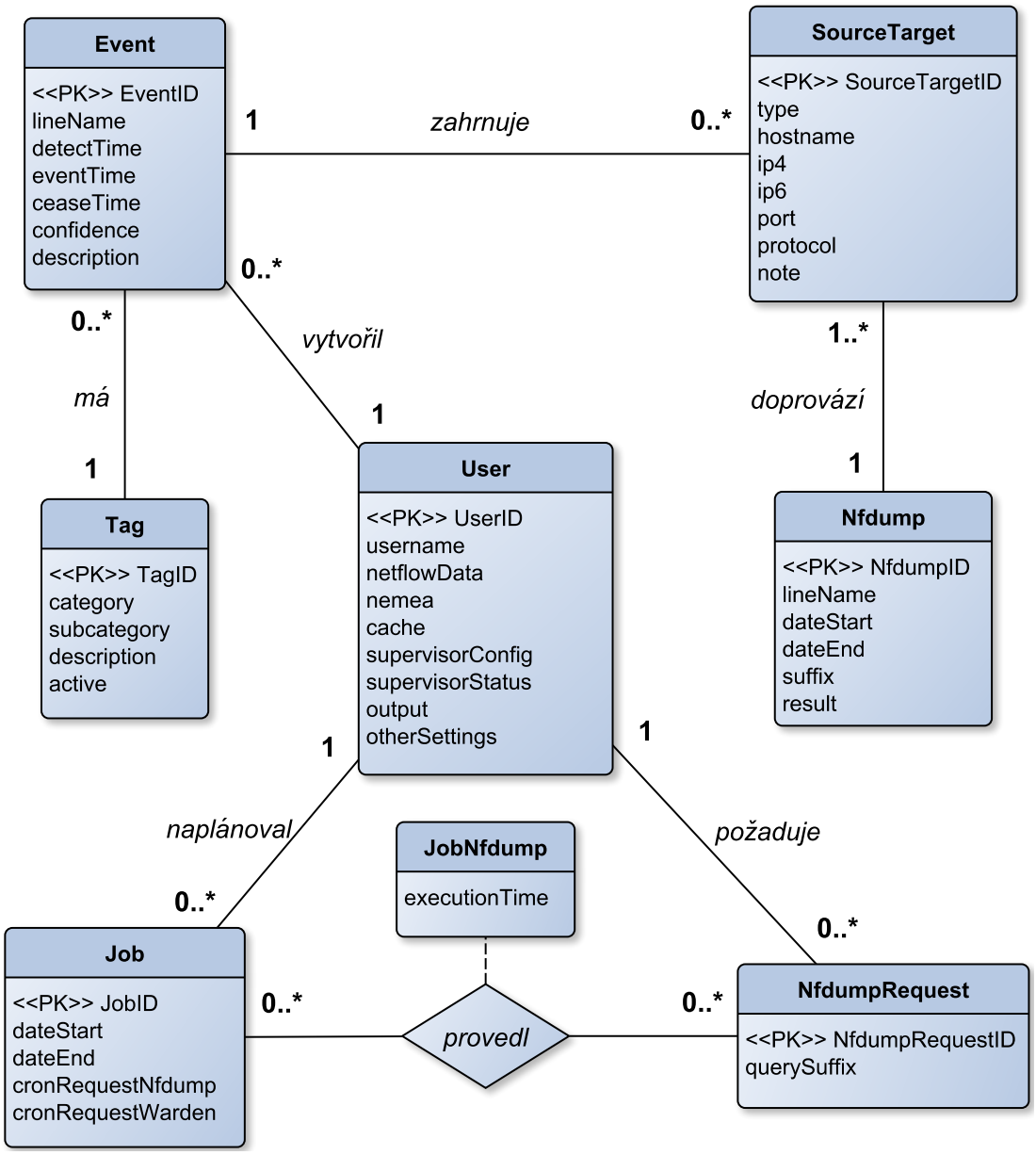
Návrhové diagramy

B.1 Diagram případů užití



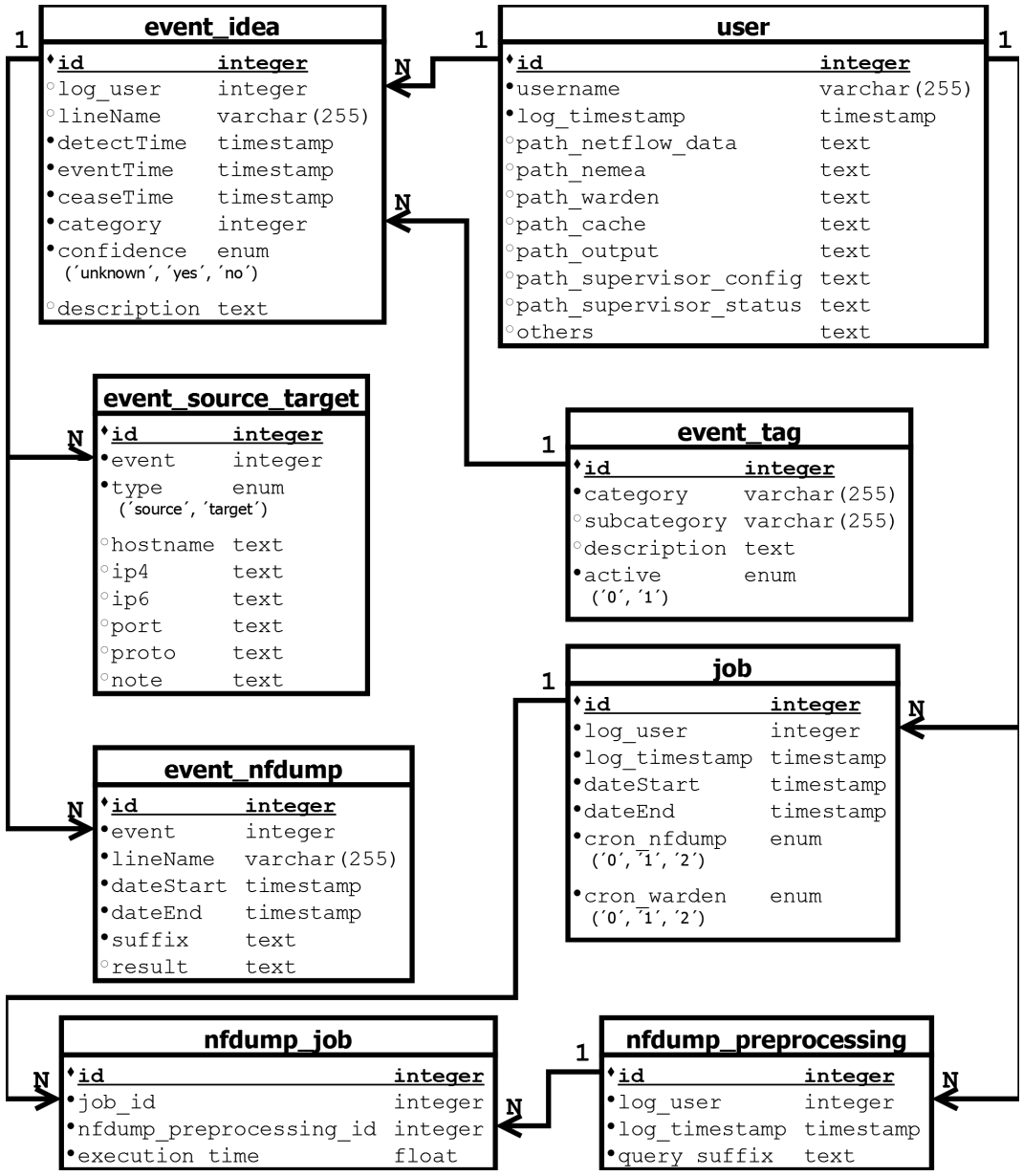
Obrázek B.1: Kompletní diagram případů užití aplikace pro anotaci NetFlow dat.

B.2 ER-diagram



Obrázek B.2: Entity-Relationship diagram zachycuje vztahy mezi entitními množinami. Popisky spojovacích čar vyjadřují typ vztahu.

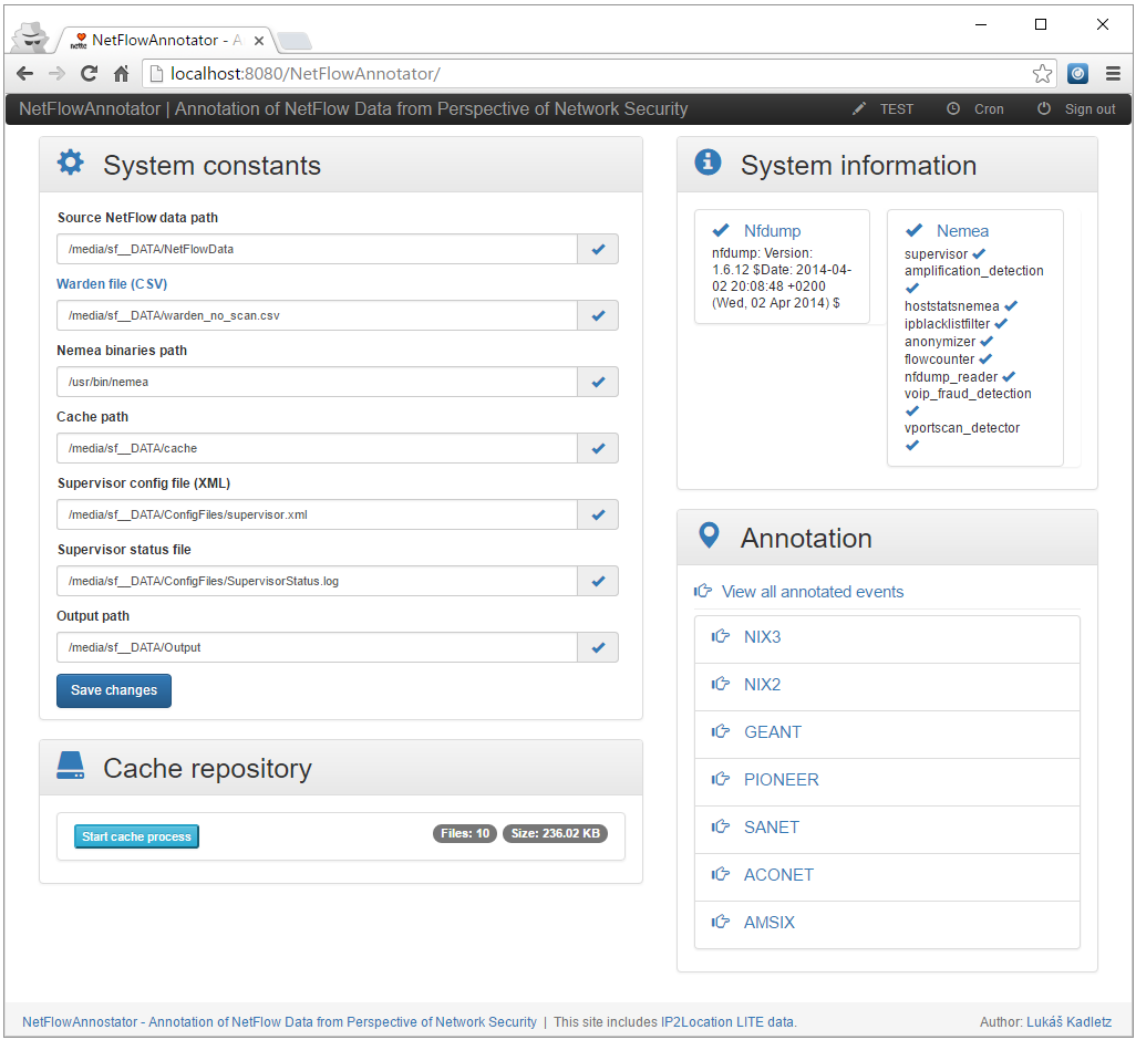
B.3 Schéma relační databáze



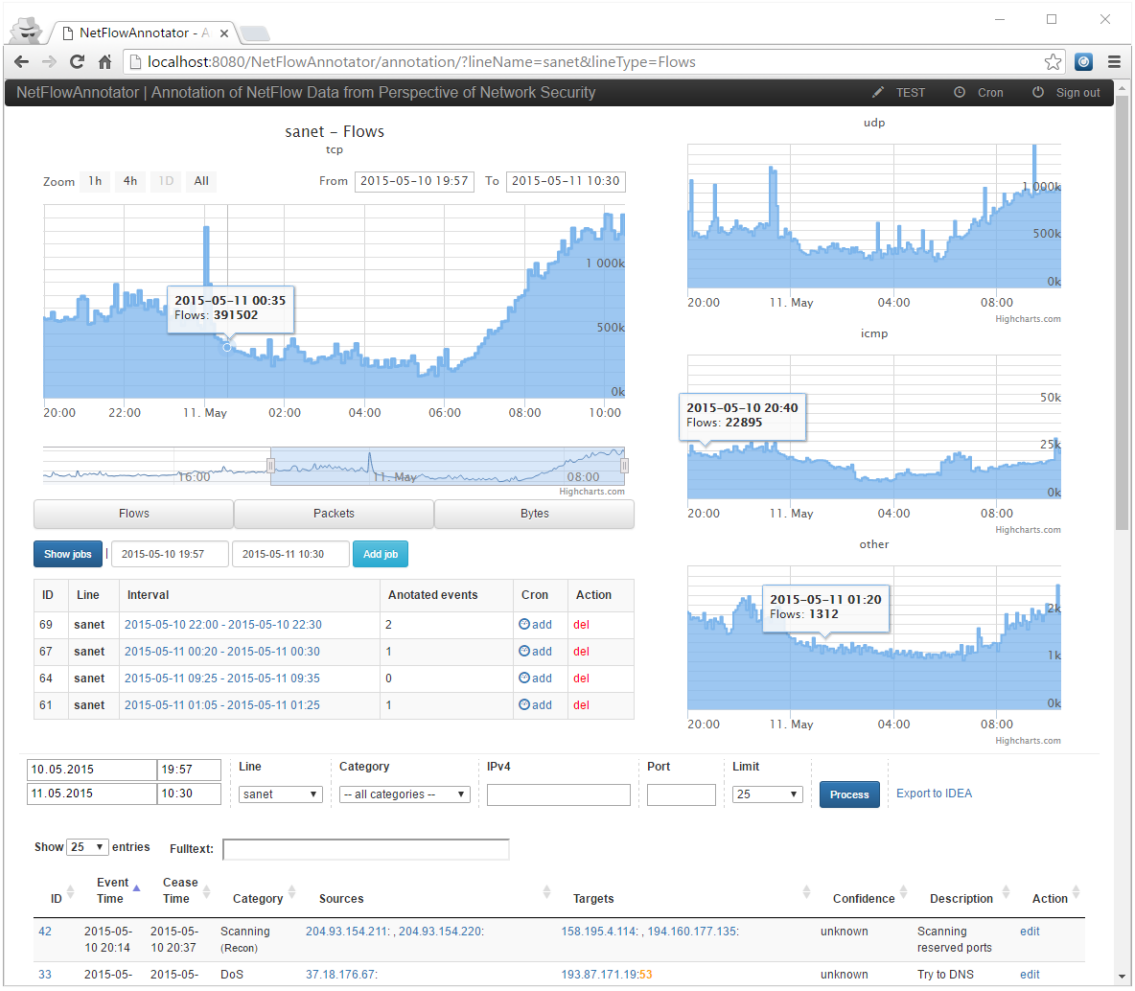
Obrázek B.3: Schéma relační databáze MySQL použité pro uchování konfigurace a data anotací.

Příloha C

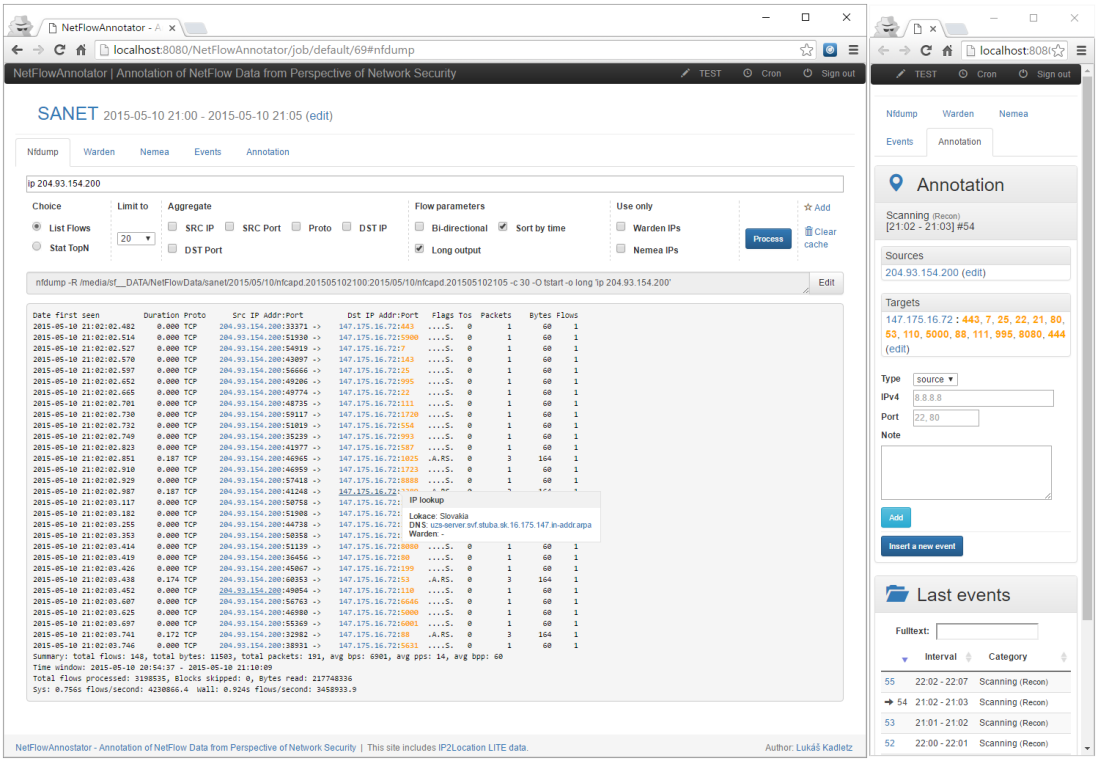
Snímky aplikace



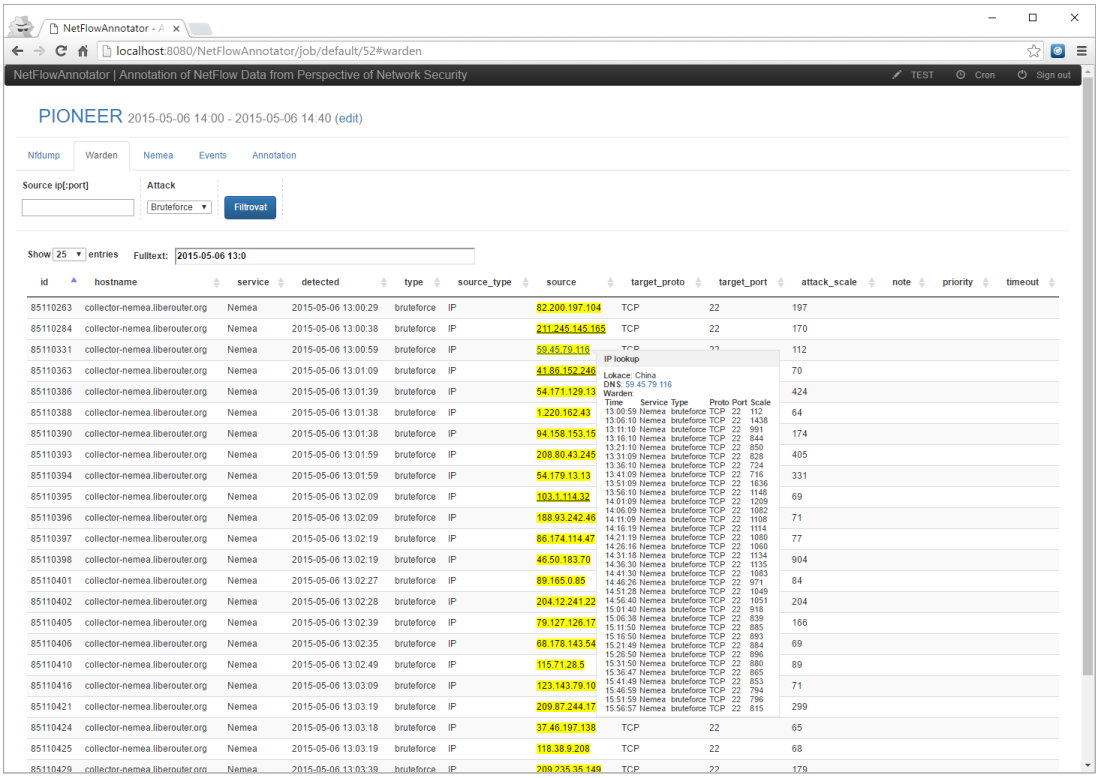
Obrázek C.1: Úvodní obrazovka obsahující práci s konstantami, pamětí a slouží jako vstupní bod k anotaci po volbě linky.



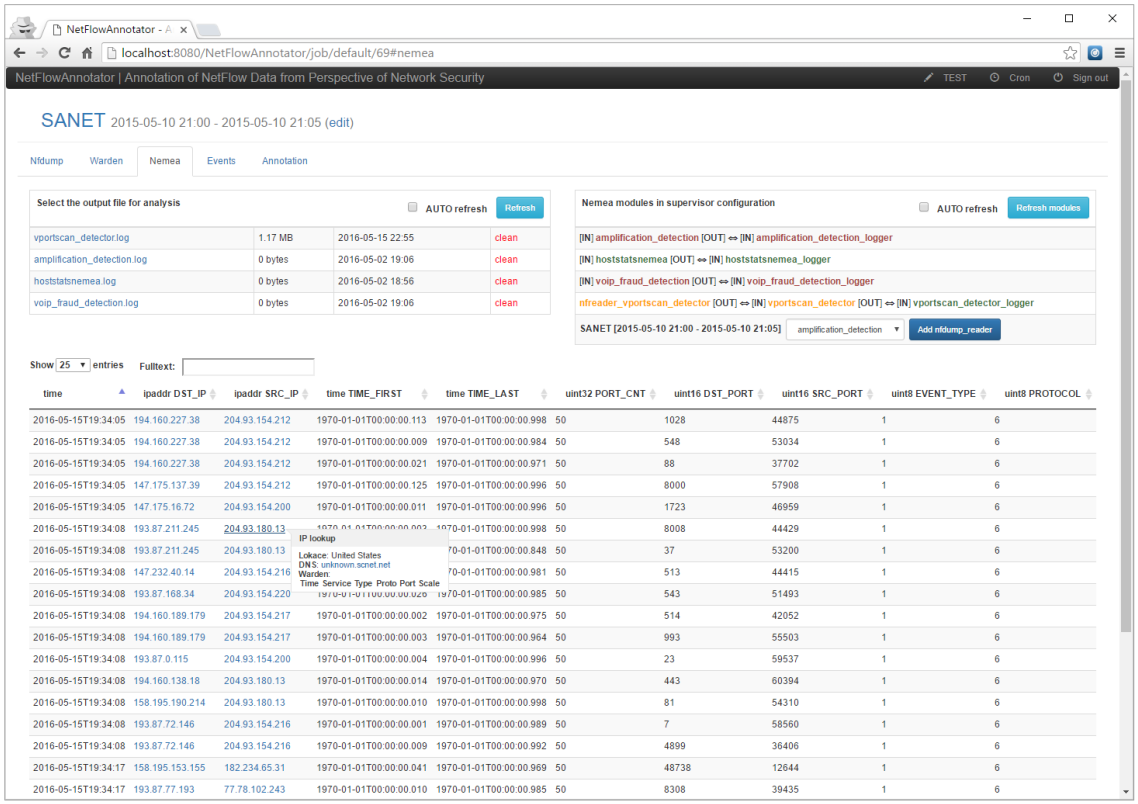
Obrázek C.2: Výběr časového intervalu pro analýzu provozu. Grafy zobrazují provoz dle počtu NetFlow toků, paketů nebo přenesených dat. Vždy jsou zobrazeny čtyři pro protokoly TCP, UDP, ICMP a ostatního provozu. Ve spodní části je zobrazeno filtrování již vytvořených anotací.



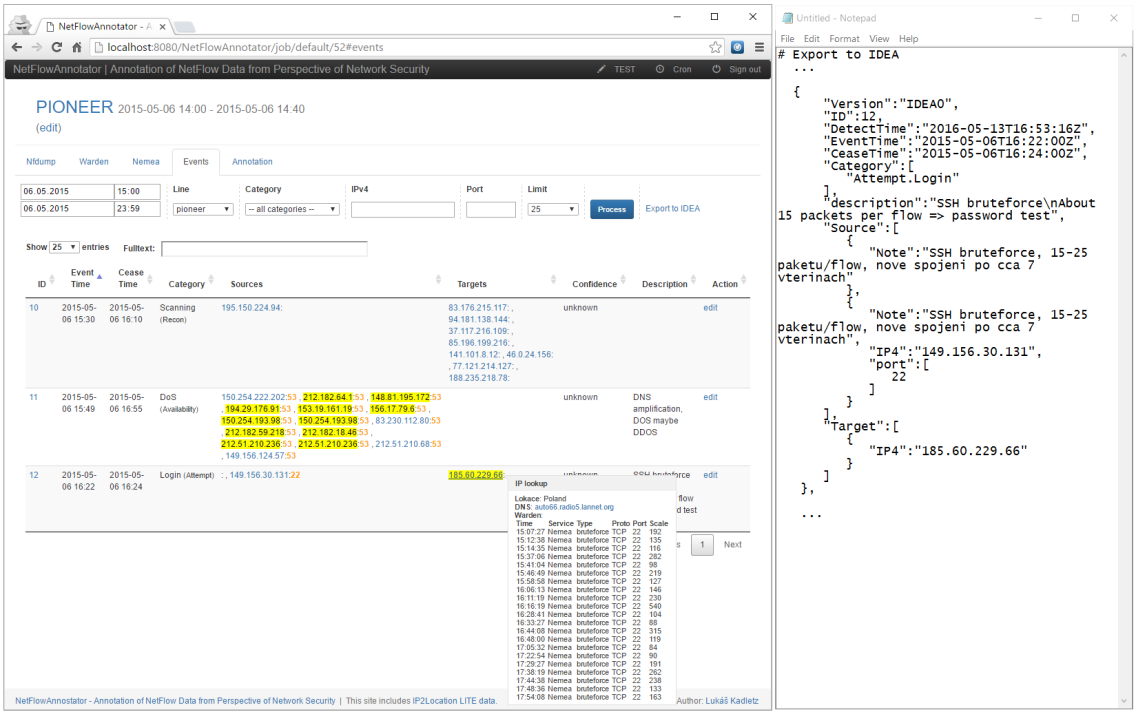
Obrázek C.3: Práce s programem **nfdump** a anotace nalezené události (skenování sítě).



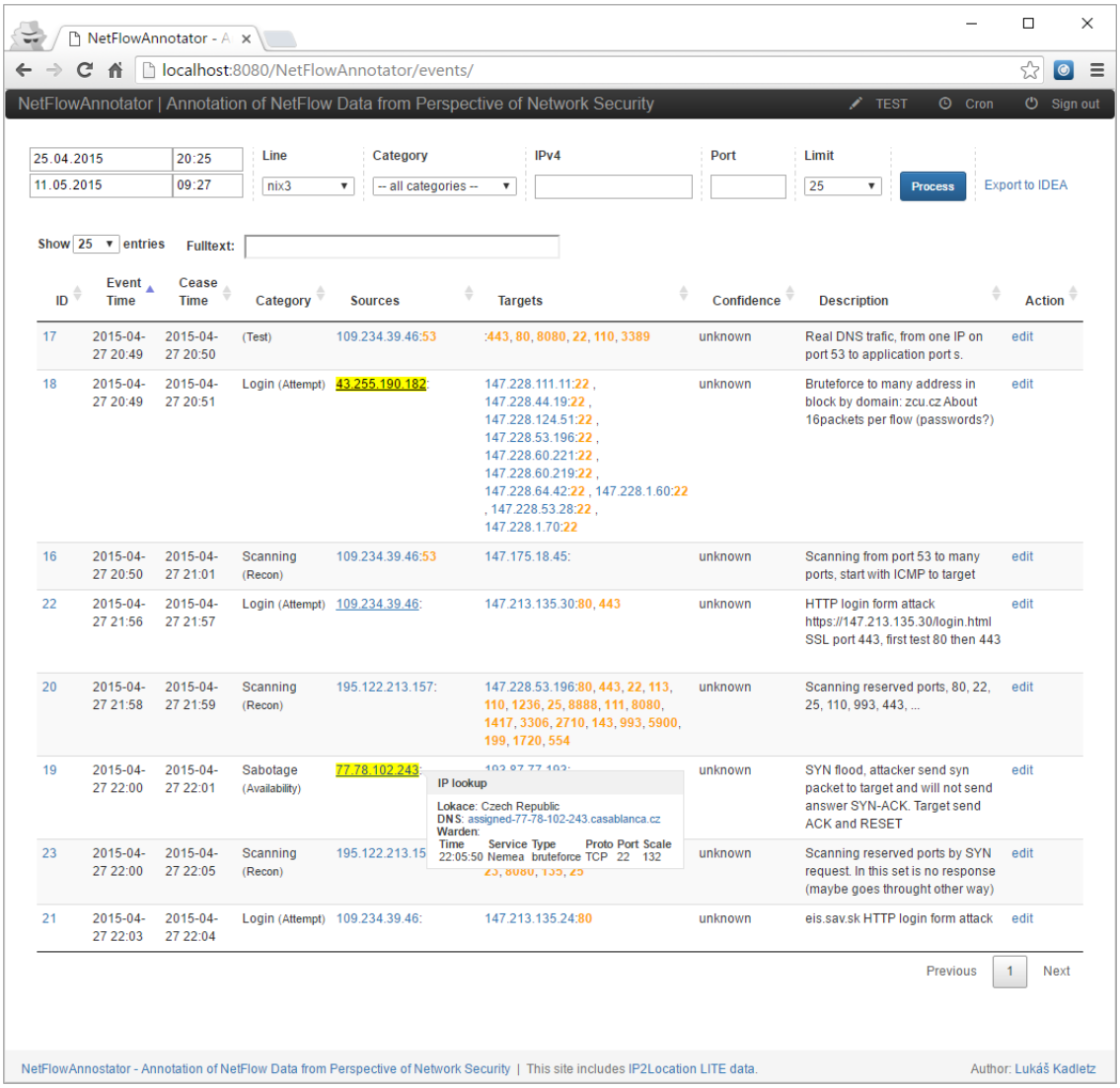
Obrázek C.4: Vyhledávání v systému Warden. Rychlý dotaz po kliknutí na IP adresu.



Obrázek C.5: Ovládání systému Nemea. Vpravo nahoře lze vyžádat zapnutí nebo vypnuti modulu nebo přidat modul `nfdump_reader` před vybraný detektor. Modul `nfdump_reader` bude po spuštění předávat data detektoru dle intervalu aktuální analyzované dávky. Vpravo nahoře jsou zobrazeny výstupní soubory, které lze přečíst a analyzovat jejich obsah. Vypsaná tabulka obsahuje vždy jinou kombinaci sloupců dle výstupu zvoleného detektoru.



Obrázek C.6: Zobrazení anotovaných událostí dle zvolené linky. Vpravo je zobrazen příklad exportovaných dat po kliknutí na odkaz `Export to IDEA`.



Obrázek C.7: Výpis všech anotací provedených na lince `nix3` včetně možnosti rychlého vyhledávání a exportu do formátu IDEA. Export lze provést po kliknutí na odkaz `Export to IDEA`.