VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

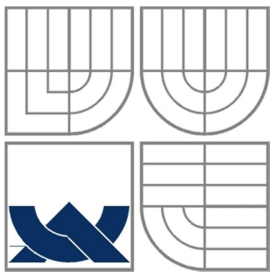
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

WEBOVÉ ROZHRANÍ PRO SYSTÉM DETEKCE SÍŤOVÝCH ANOMÁLIÍ

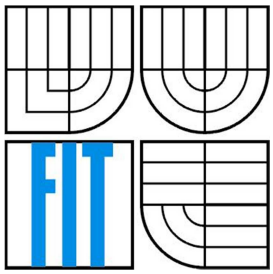
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PETR SLÁDEK
AUTHOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

WEBOVÉ ROZHRANÍ PRO SYSTÉM DETEKCE SÍŤOVÝCH ANOMÁLIÍ

WEB INTERFACE FOR NETWORK ANOMALY DETECTION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR SLÁDEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV BARTOŠ

BRNO 2013

Abstrakt

Úkolem této bakalářské práce je vytvořit webové rozhraní pro systém detekce síťových anomálií s názvem HostStats. Jeho úkolem je umožnit uživateli efektivně pracovat s daty a statistikami, které systém poskytuje. Webové rozhraní funguje jako plugin do NfSenu i jako zcela nezávislá webová aplikace. Implementace proběhla v PHP s využitím Nette Frameworku, HTML5, CSS3 a JavaScriptu s použitím jQuery knihovny.

Abstract

The goal of this work is to create a web interface for network anomaly detection system called HostStats. Its mission is to enable users to effectively work with data and statistics provided by the system. Web interface works as a plugin to NfSen as a completely independent web applications. Implementation took place in PHP using the Nette Framework, HTML5, CSS3, and JavaScript using the jQuery library.

Klíčová slova

HostStats, NfSen, NetFlow, timeslot, detekce útoků, síťové anomálie, Nette framework

Keywords

HostStats, NfSen, NetFlow, timeslot, attack detection, network anomaly, Nette framework

Citace

Sládek Petr: Webové rozhraní pro systém detekce síťových anomálií, bakalářská práce, Brno, FIT VUT v Brně, 2013

Webové rozhraní pro systém detekce síťových anomálií

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Václava Bartoše. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Sládek
15.května 2013

Poděkování

Děkuji svému vedoucímu práce Ing. Václavu Bartošovi za odborné vedení a všechny podněty a podklady, které mi k vypracování práce poskytl.

Obsah

1 Úvod

Cílem této práce je popsat a zdokumentovat webové rozhraní pro systém detekce síťových anomálií HostStats, vyvíjeného v rámci výzkumné skupiny ANT. HostStats je název systému, který využívá data v NetFlow záznamech a generuje statistiky o jednotlivých hostech na síti. V těchto statistikách poté vyhledává anomálie, na základě kterých detekuje jednotlivé síťové útoky jako DOS, horizontální a vertikální skenování portů a další.

Webové rozhraní (neboli také frontend) systému HostStats je prostředek pro zobrazování statistik ve formě tabulek, filtrů a grafů uživateli tak, aby bylo vše přehledně vidět a s daty se dalo bez problémů pracovat. Webové rozhraní je plugin do monitrovacího systému NfSen, který slouží pro práci s NetFlow záznamy. Důraz je ale taktéž kladen na to aby frontend mohl fungovat samostatně jako nezávislá webová aplikace.

Tato práce se zabývá analyzováním a určením nejlepšího vhodného technologického řešení pro implementaci webového rozhraní systému HostStats, a také popsáním implementace samotné. A to především architekturou pluginů do monitorovacího systému NfSen, fungováním aplikace ve vybraném Nette Frameworku, dále pak popsáním některých důležitých tříd objektového modelu, popsáním MVC struktury aplikace a v neposlední řadě komunikací webového rozhraní s backendovou (serverovou) částí systému.

V poslední kapitole je popsáno testování funkčnosti a použitelnosti webu. Vzhledem k tomu, že tato aplikace slouží pro administrátora či správce sítě, testování proběhlo na skupince lidí z IT oboru. Dále jsou uvedeny možná budoucí užitečná rozšíření systému.
Na přiloženém CD jsou kompletní zdrojové kódy webového rozhraní a elektronická verze tohoto dokumentu.

2 Analýza

Tato kapitola se zabývá teoretickou stránkou funkčnosti systému. Dozvíte se kde a jak se berou data pro statistiky ze kterých HostStats vychází a na základě kterých odhaluje síťové útoky. Jsou zde popsány technologie, na kterých je systém detekce síťových anomálií založen a ze kterých je proto nutné vycházet. Na závěr jsou zde uvedeny možné technologie a prostředky na kterých je vhodné stavět webové rozhraní.

2.1 Proces získávání dat pro HostStats

Stručně funguje systém takto: HostStats zpracovává NetFlow záznamy o tocích na síti. NetFlow záznamy jsou ukládány do souborů, přičemž každý soubor obsahuje záznamy z jednoho časového intervalu o délce 5 minut (tzv. **timeslot**). Pomocí opensource nástroje *nfdump* se dají data z těchto souborů číst. Pro lepší vizualizaci a práci s NetFlow záznamy se ovšem používá opensource nástroj NfSen. Jeho webový frontend zobrazuje tyto údaje uživateli ve formě grafů a tabulek s různými filtry a vyhledáváním. NfSen také nabízí podporu pro pluginy. Vždy po 5 minutách, když dojde k vytvoření nového souboru s flow záznamy, pošle NfSen pomocí pluginu zprávu s názvem souboru timeslotu na HostStats a ten ho zpracuje a vytvoří statistiku o jednotlivých hostech na síti.
Základním článkem získání dat ke statistice je tedy NetFlow.

2.1.1 NetFlow

NetFlow je protokol vytvořený, jednou z nejvíce dominujících firem na poli síťových prvků, společností Cisco Systems. Tento otevřený protokol slouží k monitorování síťového provozu na základě IP toků a to v téměř reálném čase. Toto monitorování je důležité zejména k zabezpečování počítačové sítě, ale i například pro ISP k zpoplatňování svých služeb na základě přenesených dat.

IP tok (neboli flow) je definovaný jako ukončená **sekvence paketů** se shodnými pěti údaji

- Cílová IP adresa
- Zdrojová IP adresa
- Cílový port
- Zdrojový port
- Číslo protokolu

U každého toku je evidován čas jeho začátku, délka jeho trvání, počet přenesených paketů, počet přenesených bytů a případně další údaje. Z každého toku tedy můžeme určit kdo s kým kdy komunikoval, na jakém protokolu a kolik přitom přenesl dat.

Architektura NetFlow je složena z několika sond (tzv. **exportérů**) a jednoho **kolektoru**. Exportéry jsou umístěny v různých místech sítě, na kterých chceme schraňovat statistiky a analyzují procházející pakety. Ukládají informace o toku a ve chvíli kdy tok skončí, odešlou ho na kolektor. A to buď po síti na které jsou připojené nebo po vlastní dedikované lince, která nezatěžuje monitorovanou síť. [1]

Kolektor je zařízení, které schraňuje a ukládá všechny statistiky o tocích, které přijme od exportérů. Proto musí mít velký úložný prostor. Většinou na tomto zařízení běží také software vhodný pro vizualizaci těchto NetFlow záznamů. Jedním z nich je například NfSen.

2.1.2 NfSen – Netflow Senzor

Tento oblíbený a často používaný nástroj slouží nejen jako **NetFlow kolektor**, ale ve své podstatě jako grafická nadstavba nástroje *nfdump*. Pochází také od stejného autora a je stejně tak vydávaný jako opensource. NfSen je rozdělen na backend a webový frontend. Backend slouží jako kolektor, který shromažduje NetFlow záznamy a ukládá je do RRD[1] databáze. Frontend pomocí RRD grafů vykresluje využití sítě (v bytech, paketech i tocích) a další statistiky. Tyto data se dají samozřejmě filtrovat podle portů, ip adres, jednotlivých senzorů atp…

Jak backend, tak i frontentd mají své pluginy. Backendové jsou volané při různých událostech, jako je například nový timeslot, nějaké definované upozornění a podobně. Frontendové pluginy slouží k vlastní vizualizaci a filtraci dat. A také k volání funkcí v backendových pluginech. [2]

Tyto pluginy jsou využity i pro HostStats. Backendový plugin posílá nová data do HostStats ke zpracování a Frontendový plugin zajišťuje vykreslení HostStats frontendu v layoutu NfSenu.

2.1.3 HostStats

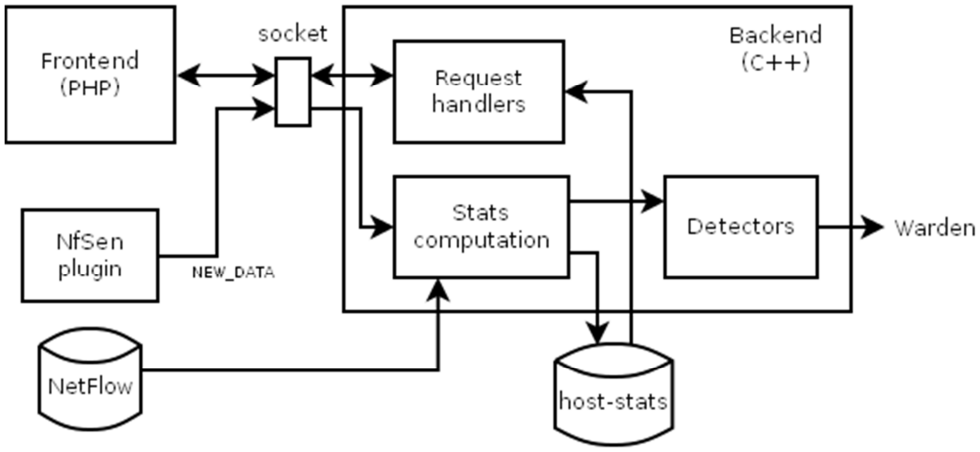
HostStats je název systému pro analýzu dat a detekci síťových anomálií vyvíjený pod výzkumnou skupinou ANT, jehož základem je výpočet statistik o jednotlivých hostech (IP adresách) v síti.
Systém je vyvíjen jako plug-in pro NfSen. Každých 5 minut jsou z NetFlow dat vypočítány informace o provozu jednotlivých IP adres. V těchto informacích pak jsou vyhledávány adresy vykazující podezřelé chování.

Komunikace jednotlivých prvků systému je znázorněna na schématu (obrázek 1). Backendový plugin NfSenu, napsaný v jazyce perl, posílá jednou za 5 minut přes socket zprávu s názvem nového timeslotu. HostStats server (backend) timeslot analyzuje a vypočítá statistiky pro

[1] RRD – Databázový nástroj, který se zaměřuje na zpracování a ukládání časově závislých dat

jednotlivé hosty (IP adresy) a statistiku uloží na disk. Ze vzniklé statistiky detekuje hrozby, jejichž seznam taktéž uloží na disk k dalšímu použití, případně odešle informace do systému Warden [2].

Na všechny uložená data se opět přes socket dotazuje webový frontend. Ten je poté zobrazuje v uživatelsky přijatelné formě uživateli. Podrobně je komunikace HostStats backendu s frontendem popsána v kapitole 2.3.



Obrázek 1: Schéma datových toků HostStats

2.2 Architektura pluginů NfSen

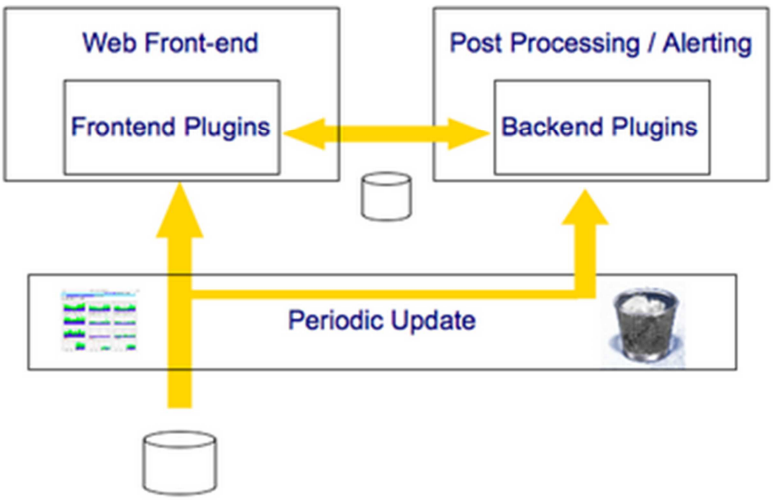
Webové rozhraní (frontend) systému HostStats musí fungovat jako plugin do NfSenu. A to především z toho důvodu, aby byly všechny nástroje používané k analýze NetFlow pěkně na jednom místě. Na druhou stanu je ovšem vhodné aby webový frontend mohl být funkční i samostatně bez použití NfSenu.

Jak už bylo řečeno NfSen nabízí podporu pro frontendové pluginy psané v PHP , které jsou vidět na webovém frontendu NfSenu a backendové pluginy, které se spouští jako perl skript a mohou komunikovat s frontendovými pomocí aplikačního rozhraní.

Z důvodu potřeby samostatnosti našeho webového rozhraní, není možné použít nabízené rozhraní pro frontendové pluginy. Proto je použit jen drobný můstek, kterým je php soubor *hoststats.php* v adresáře NfSen pluginů. Ten vypne automatické obnovování stránky v nastaveném intervalu, vloží iframe s adresou našeho webového rozhraní a pomocí JavaScriptu ho roztáhne na celou možnou plochu uživatelova prohlížeče.

[2] WARDEN - systém pro efektivní sdílení informací o detekovaných událostech (hrozbách). Systém je vyvíjen v rámci projektu Velká Infrastruktura CESNET, který řeší sdružení CESNET. https://csirt.cesnet.cz/Warden

Jakákoli jiná možnost velmi komplikovala využít kterékoli framework, ajaxové požadavky, vlastní CSS styly atd. NfSen rozhraní pro PHP frontendové pluginy není pro funkčnost HostStats webového rozhraní nutností, protože to komunikuje přes sockety přímo s HostStats backendem a nevyžaduje jakékoliv propojení s NfSenem.



Obrázek 2: Schéma architektury pluginů v NfSen [2]

2.3 Komunikace Frontendu s Backendem

Frontend posílá požadavky na socket otevřený backendem (standardně *localhost* na portu *3333*) ve formátu:

- první byte - kód požadavku
- řetězec libovolné délky - parametry
- poslední byte - ukončení zprávy nulovým znakem ('\0')

Např:

```
\x1Eall;201302171200;flows > 100;50;0\0
```

Počet a význam parametrů závisí na typu požadavku, parametry jsou obvykle oddělovány pomocí středníku.

Backend odešle zpět požadovaná data (formát závisí na typu požadavku) a uzavře spojení. Pro každý požadavek je tedy nutné vytvořit nové spojení.

Pokud není specifikováno jinak:

- všechny časy (timesloty) se posílají ve formátu: yyyymmddHHMM.
- parametr "profil" určuje profil, ze kterého se získávají data, je to buď "all", nebo některý z řetězců získaných voláním GET_PROFILES viz dále.

Pokud dojde k chybě, je místo očekávané odpovědi odeslána chybová hláška začínající řetězcem "ERROR". Ta je potom vyvolaná jako Exception a dále odchycená a zpracovaná v aplikaci.
Dále jsou uvedeny jednotlivé **typy požadavků**. V závorce je vždy uveden kód požadavku v desítkové soustavě.

2.3.1 NEW_DATA (1)

Speciální požadavek, který nepřichází z webového rozhraní, ale posílá ho NfSen backend plugin. Říká tím, že jsou k dispozici nová data. Na tento požadavek jako jediný není odesílána žádná odpověď.

Parametry:

```
timeslot
```

2.3.2 GET_STATUS (10)

Vrátí základní stavové informace o stavu backendu. Požadavek nemá žádné parametry.

Formát odpovědi

```
promenna=hodnota;promenna=hodnota;...;promenna=hodnota
```

V současnosti se vracejí tyto proměnné:

- `processing` - 1 pokud plugin právě zpracovává nová data, jinak 0
- `timeslot` - aktuálně zpracovávaný, nebo poslední zpracovaný timeslot
- `flows` - počet načtených toků v posledním timeslotu
- `hosts` - počet načtených hostů v posledním timeslotu

2.3.3 GET_HOST_CNT_HISTORY (11)

Vrací historii počtu hostů v minulých timeslotech

Paramery:

```
profil;starttimeslot,endtimeslot
```

Formát odpovědi

```
timeslot=hodnota;timeslot=hodnota;...;timeslot=hodnota
```

Hodnota pro každý timeslot mezi `starttimeslot` a `endtimeslot` (včetně), ke kterému jsou data k dispozici.

2.3.4 GET_FLOW_CNT_HISTORY(12)

Vrací historii počtu toků v minulých timeslotech. Parametry a formát odpovědi stejné jako u `GET_HOST_CNT_HISTORY`.

2.3.5 GET_PROFILES(20)

Vrátí seznam dostupných profilů. Nemá žádné parametry.

Formát odpovědi:

```
nazev_profilu_1;nazev_profilu_2;...;nazev_profilu_n
```

2.3.6 GET_FIELD_LIST(30)

Vrátí seznam položek, ze kterých se skládá záznam o hostu.

Parametry:

```
profile
```

Formát odpovědi:

```
polozka;polozka;...;položka
```

Příklad:

```
address;in_flows;in_packets;in_bytes;out_flows;out_packets;out_bytes
```

2.3.7 GET_TIMESLOT_DATA(31)

Na tento požadavek přijde jako odpověď tabulka dat z jednoho timeslotu.

Parametry:

```
profile;timeslot;filter;limit;sort_by;ascending
```

- `filter` - filtrovací pravidlo, formát kontroluje backend, z hlediska frontendu jde o libovolný řetězec). Může být prázdné.
- `limit` - maximální počet vrácených záznamů (kladné číslo)
- `sort_by` - název položky, podle jejíž hodnoty budou data seřazena (jeden z názvů získaných voláním `GET_FIELD_LIST`), může být prázdné, pak data nejsou řazena
- `ascending` - 1 pokud mají být data řazena vzestupně, jinak jsou řazena sestupně (default)

Formát odpovědi:

Tabulka, řádky odděleny pomocí '`\n`', hodnoty na řádku pomocí '`;`'.

- První řádek - popisky sloupců tabulky (názvy položek)
- Každý další řádek jeden záznam (tj. adresa a hodnoty položek)

Příklad:

```
address;in_flows;in_packets;in_bytes;out_flows;out_packets;out_bytes
10.0.0.1;2;2;120;1;4;2405
192.168.1.2;54;510;14028;27;98;841
2001:db8::1428:57ab;0,0,0;250;250;15000
```

2.3.8 GET_TIMESLOT_IPMAP (32)

Na tento požadavek přijde jako odpověď tabulka dat z jednoho timeslotu agregované podle /16 prefixu IPv4 adresy. Tyto data se používají pro vykreslení IP mapy (viz kapitola 253.3.3).

Parametry:

```
profile;timeslot;base_address;prefix_len
```

- `base_address` – Adresa specifické podsítě
- `prefix_len` – Délka prefixu předchozí adresy

Formát odpovědi:

`Stejný jako v předchozím požadavku`

2.3.9 GET_HOST_HISTORY(35)

Vrací statistiky o jednom hostu z daného časového intervalu, tj. ze všech existujících souborů s časovou značkou mezi `starttimeslot` a `endtimeslot` (včetně).

Parametry:

```
profile;address;starttimeslot;endtimeslot
```

- `address` - adresa hosta, jehož statistiky jsou požadovány (IPv4 nebo IPv6 v běžném textovém formátu)
- `starttimeslot` - začátek časového rozsahu (první požadovaný timeslot)
- `endtimeslot` - konec časového rozsahu (poslední požadovaný timeslot)

Formát odpovědi:

Tabulka, řádky odděleny pomocí '`\n`', hodnoty na řádku pomocí '`;`'.

- První řádek - popisky sloupců tabulky (názvy položek)
- Každý další řádek jeden záznam (tj. timeslot a hodnoty položek)
- Pokud se požadovaná adresa v datech z určitého timeslotu nevyskytuje, jsou všechny hodnoty záznamu nulové.
- Pokud nějaký timeslot není vůbec k dispozici (soubor neexistuje), je řádek s tímto timeslotem vynechán.

Příklad:

```
timeslot;in_flows;in_packets;in_bytes;out_flows;out_packets;out_bytes
201301021200;2;2;120;1;4;2405
201301021205;0;0;0;0;0;0
201301021210;212;215;4320;250;250;15000
```

2.3.10 GET_DETECTION_LOG_LIST(40)

Vrací seznam dnů, pro které je k dispozici log detekovaných událostí. Tento požadavek nemá žádné parametry.

Formát odpovědi:

Seznam dní ve formátu `yyyymmdd` oddělených pomocí '`\n`'

Příklad:

```
20130102
20130103
20130104
```

2.3.11 GET_DETECTION_LOG(41)

Vrací obsah logu detekovaných událostí pro zadaný den

Parametry:

`day` (den ve formátu `yymmdd`)

Formát odpovědi:

Tabulka, řádky odděleny pomocí '`\n`', hodnoty na řádku pomocí '`;`'.

Příklad:

```
201202151115;portscan_h;6;43.99.57.181;;;;1516;horizontal SYN scan
201202151115;portscan_h;6;138.243.71.66;;;;1413;horizontal SYN scan
201202151115;dos;6;;20fd:6cfa:e321:ff:0:ffff:93fa:e3fe;;;124112;
```

Význam jednotlivých sloupců:

1. Timeslot, ve kterém byla událost detekována
2. Typ události, možnosti:
   - `portscan` (skenování portu bez bližšího určení)
   - `portscan_h` (horizontální skenování portu, tj jeden port na více adresách)
   - `portscan_v` (vertikální skenování portu, tj různé porty na jedné adrese)
   - `bruteforce` (hádání hesel hrubou silou)
   - `dos` (denial od service útok)
   - `other` (ostatní)
3. protokol, přes který byl útok veden (1=ICMP, 6=TCP, 17=UDP)
4. zdroj události (IPv4 nebo IPv6 adresa)
5. cíl události (IPv4 nebo IPv6 adresa)
6. zdrojový port
7. cílový port
8. intenzita události (např. počet proskenovaných adres/portů, počet toků DoS útoku apod.)
9. poznámka – jakýkoli řetězec

Protokolů, adres a portů může byt i vice, v tom případě jsou jednotlivé hodnoty odděleny čárkou ',' .

2.4 Potencionální technologie Frontendu

Vzhledem k zadání je nutné použít pro webové rozhraní PHP na straně serveru a Javascript na straně klienta. To ulehčuje výběr a odpadají tím možnosti jako Ruby, ASP, Perl či Python. Ovšem používat v dnešní době PHP bez nějakého frameworku je velmi nevhodné a časově neekonomické. PHP frameworků je široká škála a proto je nutné vybrat ten nejvhodnější.

K nejvíce používaným a nejlépe dokumentovaným patří Zend Framework, Nette Framework, CodeIgnitier, CakePHP a Sympony. Všechny tyto Frameworky samozřejmě fungují na PHP5, které obsahuje vylepšenou podporou pro objektově orientované programování.

Zend Framework pochází od stejné společnosti jako PHP samotné, je stále vyvíjený a má podrobnou a udržovanou dokumentaci. Oproti tomu je ale relativně pomalý, obsahuje mnoho souborů a tříd, které v normální aplikaci programátor vůbec nevyužije a oproti ostatním frameworkům není moc uživatelsky příjemný. Skoro vše se konfiguruje a předává přes pole, místo toho aby se použil objektový přístup.

Nette Framework vyvíjí malá česká komunita. Na začátku roku vyšla jeho 2 verze, která je stabilní a kvalitně česky i anglicky zdokumentovaná. Obsahuje skvělý šablonovací systém Latte a vynikající systém vytváření formulářů a komponent. Do Nette existují také různé pluginy, knihovny a rozšíření od uživatelské komunity. Tento Framework se stává velmi populární a pro své webové aplikace ho použili například servery Ulozto.cz, Denik.cz nebo kancelář prezidenta ČR.

CodeIgnitier, CakePHP a Sympony jsou menší frameworky. Podporují, stejně jako předchozí dva, základní nástroje jako MVC strukturu, formuláře, emaily atp. Využívají se ovšem méně a nejdou úplně s dobou.

Na základě nezávislého testu [3] a prohlédnutí dokumentace a příkladů jednotlivých frameworků jsem se rozhodl použít pro Webové rozhraní HostStats framework Nette. A to především kvůli šablonovacímu systému a skvělé podpoře AJAXu.

Pro vývoj aplikace tohoto typu se hodí taktéž CSS Framework. Ten slouží k vytvoření základního rozložení layoutu webu a také obsahuje typografii a základní styly formulářových prvků, tabulek, různých zpráv, stavů apod.

Takovýchto volně dostupných frameworků se dá najít také mnoho. Některé, jako například 360.gs, jsou jen na vykreslení tzv „mřížky[3]“ webu. Jiné mají i grafické styly a JavaScript pro záložky, vysouvací menu apod. Mezi nejznámější patří YUI od Yahoo, YAML CSS Framework a Twitter Bootstrap.

Já jsem pro implementaci zvolil ten poslední. Twitter Bootstrap má široké využití ve všech „administračních“ prostředích, má hezký design, který se dá jednoduše přizpůsobit a nabízí spoustu užitečných prvků.

3 Implementace

Po výběru optimálních technologií a prostudování dokumentace technologií, na které je potřeba navázat bylo množné začít implementovat. V této kapitole jsou popsány technologie, frameworky a knihovny použité pro implementaci.

[3] Grid neboli mřížka webu je rozmístění jednotlivých celků na stránce

Dále je uveden objektový návrh MVP struktury, životní cyklus aplikace a podrobněji popsány důležité třídy aplikace.

3.1 Použité technologie

3.1.1 HTML5

HTML neboli hypertextový značkovací jazyk je jedním ze základních jazyků webu. Tedy publikace dokumentů na internetu. Dnes je HTML spolu s JavaScriptem, stylovacím jazykem CSS a serverovým jazykem PHP základem pro většinu internetových prezentací.

Další vývoj HTML přímo ovlivňuje rozvoj internetových aplikací a umožňuje vývojářům realizovat čím dál tím více nápadů. Proto se s každou novou verzí internet posune o neuvěřitelný krok kupředu. Stejně tak se stalo i v případě vydání poslední verze HTML5.

HTML5 se začala vyvíjet kolem roku 2007, a její konečné schválení je plánováno až na rok 2014. Proto je verze 5 stále velmi živá a její podpora je ze strany prohlížečů přijata různě. Velká část HTML 5 je v současné době podporována všemi velkými prohlížeči a je tedy možné mnoho z nové specifikace již používat.

HTML verze 5 se od verze 4 liší novými, zkrácenými a rychlejšími zápisy značek (tagů). Autoři dávají důraz na jednoduchost a zároveň účinnost. V HTML5 je též možné vytvořit aplikaci, která funguje v prohlížeči i tehdy, když uživatel nemá internetové připojení, a která ukládá data do lokálního úložiště na uživatelově počítači. Je-li internetové připojení k dispozici, může aplikace synchronizovat data se vzdáleným serverem.

```
<!DOCTYPE html>
<html lang="cs" dir="ltr">
  <head>
    <meta charset="UTF-8">
    <title>Titulek stránky</title>
  </head>
  <body>
    Tělo stránky
  </body>
</html>
```

Obrázek 3: Ukázka HTML5 kódu

Příklad těla dokumentu ukazuje obrázek 3. Na první pohled je zřejmé, že je velmi zjednodušený *DOCTYPE* a také meta tagy. Vzhledem k tomu že HTML5 vychází stejně jako starší HTML4 ze SGML nemusí být všechny tagy párové jako tomu bylo u XHTML vycházejícího z XML.

Jazyk dále obsahuje mnoho nový značek (tagů). Například <*header*>, <*footer*>, <*section*> a <*article*> pro jasnější členění obsahu nebo <*video*>, <*audio*>, <*source*> a <*track*> pro přidávání multimediálního obsahu. Za zmínku také stojí nové typy formulářových prvků jako „number“ pro číselné hodonty, „search“ pro vyhledávání, „range“ pro číselý rozsah nebo například „date“ pro datum, které rozšiřují práci s webovými formuláři bez nutnosti použití externích javascriptových knihoven.

Dále HTML5 obsahuje novinky jako geolokaci, vykreslovací *canvas* pro kreslení grafů, schémat nebo obrázků a již zmiňované *filestorage* pro ukládání dat na disk. HTML5 je se všemi těmto novinkami zpětně kompatibilní a web v něm psaný je částečně funkční i ve starších prohlížečích které „pětku“ nepodporují. [4]

3.1.2 CSS3

Kaskádové styly (CSS) jsou dalším základním kamenem tvorby webů. Hlavním smyslem CSS je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí a proto konsorcium W3C začalo pracovat na standardu CSS. Pro moji aplikaci jsem zvolil nejnovější verzi tohoto jazyka a to CSS3.

Kaskádové styly ve své třetí verzi nabízejí mnoho novinek ulehčující kodérům práci a umožňujících psát čistší přenositelnější kód a které odstraňují rozdíly mezi jednotlivým prohlížeči. Za zmínku stojí například CSS vlastnost **border-radius**, díky které lze u blokových i řádkových elementů nastavit zaoblení rohů. Další takovou novinkou jsou vlastnosti **box-shadow** a **text-shadow** pro vytvoření stínu blokům nebo textu. Velké využití mají také tzv. *transformace*, ty slouží k jednoduchým animacím například při *hover* efektu (najetí myší) nebo při focus (zaktivnění) nějakého formulářového prvku. Další velkou výhodou je možnost zadávat barvy ve formátu RGBA podporující alfa-kanál pro průhlednost.

Mnohé tyto vlastnosti měli prohlížeče už dříve, ovšem každý na to měl své vlastnosti. Například Mozzila používala *–moz-border-radius* a chrome *–webkit-border-radius.* Specifikace CSS3 tyto rozdíly eliminuje a popsuje přesné chování ve všech prohlížečích podporujících CSS3. [5]

3.1.3 PHP Framework Nette

Nette je populární nástroj pro vytváření webových aplikací v PHP 5. Tento Framework bez speciálního zásahu programátora elimuje výskyt bezpečnostních děr jako např. XSS, CSRF, session

hijacking, session fixation atd. Disponuje skvělými ladícími nástroji, jak pro vypisování a odhalování chyb, tak pro ladění SQL dotazů, optimalizaci rychlosti apod. [6]

Programování v Nette Framework Vás také vede k čistému návrhu aplikace s důrazem na budoucí rozšiřitelnost. Díky velké a aktivní komunitě nabízí velké množství doplňků, jako například různé DataGridy, formulářové prvky, užitečné knihovny a podobně.

3.1.4 Konfigurace přes Neon

NEON je textový soubor s příponou *.neon, který se používá pro definici konfigurace. Je jednoduší a pro člověka čitelnější než např. INI, JSON nebo PHP pole. NEON používá velmi podobnou syntaxi jako YAML[4]. Hlavní rozdíl je v tom, že NEON podporuje tzv. „entity“, pro odsazení používá tabulátory, syntaxe je o něco jednodušší a jeho analýza je rychlejší. Syntaxe je blíže popsaná v dokumentaci [7].

Aplikace používá dva konfigurační soubory. První, který je uložen *v app/config/application.neon* a nastavuje prostředí aplikace, framwork, php, služby a podobně. A druhý (*app/config/config.neon*), ve kterém se nastavuje přizpůsobení HostStats frontendu. Dají se zde změnit formáty datumu, email na který chodí případné chyby v aplikaci, parametry (adresa a port) pro připojení k HostStats, interval obnovování přehledu a adresu serveru na který se mají posílat whois dotazy.

3.1.5 JavaScriptová knihovna jQuery

Tato velmi populární knihovna velmi usnadňuje práci s JavaScriptem na webu. Její filozofie je úplně oddělit „chování“ od struktury HTML dokumentu, stejně jako CSS odděluje grafické prvky a formátování. jQuery také využívá podobné selektory jako CSS. Například místo běžného `document.getElementById('login')` stačí napsat `$('#login')`.

Jak je vidět i jQuery se drží hesla „write less, do more“ (piš málo, udělej hodně). Knihovna nabízí mnoho animací, efektů, událostí a nesčetné množství rozšíření a pluginů, které si psát sám by byla v dnešní době velká ztráta času.

Ve webovém rozhraní HostStats jsou použity jQuery pluginy pro generování grafů pro HTML5 canvas (*jquery.flot.js*), pro práci s cookies (*jquery.cookiee.js*), pro jednotný vzhled zaškrtávacích políček, přepínačů a dalších formulářových prvků (*jquery.uniform.js*), pro práci s AJAXem a Nette (*jquery.livequery.js* a *jquery.nette.js*) a další.

[4] YAML je formát pro serializaci strukturovaných dat. Výhodou tohoto formátu je, že je čitelný nejen strojem, ale i člověkem. Jednou ze zásadních věcí tohoto formátu je, že nepodporuje znaky tabulátorů. Ty musí být nahrazeny mezerami, jinak vše vyústí v chybu.

3.1.6 CSS Framework Bootstrap

Pokud se jedná o administrační rozhraní, ovládání nebo nastavování nějakého systému je zbytečné vymýšlet nějakou speciální grafiku. Většinou poslouží kvalitní CSS Framework, který za použití správných CSS tříd vytvoří celý vzhled. Jedním z takovýchto frameworků je Bootstrap od tvůrců známé sociální sítě Twitter. Autoři o něm tvrdí že je elegantní, intuitivní a výkonný front-end framework pro rychlejší a snadnější vývoj webových aplikací. [8]

Bootstrap definuje fluidní i statické layouty. Pro frontend hoststats byl zvolen fluidní, který se přizpůsobí šířce monitoru a tím pádem zvýhodní lidi s velkými monitory, kterým se vejde na jednu obrazovku více informací.

Dále jsou využity styly pro tabulky, formuláře, chybové informace a stavy, boxy, menu atd.

3.1.7 AJAX & Snippety

Moderní webové aplikace dnes běží napůl na serveru a napůl v prohlížeči. AJAX je tím klíčovým spojovacím prvkem. AJAX umožňuje změny na webové stránce, bez toho aniž by se musela celá znovu načítat. To celé provádí pomocí JavaScriptu, který na pozadí posílá běžné HTTP požadavky na server. Jejich odpověď (většinou ve formě JSONu, XML, ale klidně i HTML či „plain text“) zpracovává a vykresluje pomocí orientace v DOMu zpět do webové stránky.

Nette podporuje AJAX hned několika způsoby. Umí pomocí hlavičky HTTP požadavku rozpoznat, zda se jedná o AJAXový požadavek nebo o „normální“. Na základě toho pak vykreslí buď celou šablonu, nebo pošle jen odpověď ve formátu JSON (tzv. payload), který zpracovává JavaScript.

```
public function actionDelete($id)
{
    if ($this->isAjax()) {
        $this->payload->message = 'Success';
    }
    ...
}
```

Obrázek 4: AJAXový payload v Nette

Do tohoto payloadu je možné zapsat svoje vlastní proměnné, jak je vidět na příkladu (obrázek 4). Daleko silnější nástroj ovšem představuje vestavěná podpora AJAXových snippetů. Díky ní lze udělat z obyčejné aplikace AJAXovou prakticky několika řádky kódu. A přitom všem aplikace poběží správně, i když webový prohlížeč AJAX nebo Javascript nepodporuje nebo ho má vypnutý.

Snippety fungují tak, že při prvotním (tedy neAJAXovém) požadavku se přenese celá stránka a poté se při každém již AJAXovém požadavku na stejnou stránku přenáší pouze kód změněných částí ve zmíněném úložišti `payload`. Tím se poté pomocí JavaScriptu pouze v příslušném bloku nahradí.[9]

Na každý odkaz v HTML kódu šablony, které má třádu *ajax* je navěšena akce *onClick*, která AJAXovým GET požadavkem zavolá, jeho cílovou adresu, a očekává payload v JSONu, ze kterého přečte obsah snippetů a těm poté aktualizuje obsah. Obsah snippetu se do payloadu dostane tak, že je v šabloně tento blok kódu speciálně označen (Latte makrem {snippet}) a kdykoliv, v průběhu životního cyklu aplikace, ještě před vykreslením (viz kapitola 3.2.2) je označen jako změněný.

V naší aplikaci se AJAXem odesílají i formuláře. Pomocí jQuery doplňku jquery.ajaxSubmit.js se z formuláře posbírají hodnoty a AJAXovým POST požadavkem se pošlou ke zpracování na presenter. Ten vrátí stejný payload jako v případě GETu v předchozím odstavci.

3.2 Objektový návrh aplikace

3.2.1 MVP struktura

Model-View-Controller je softwarová architektura, která vznikla z potřeby oddělit u aplikací s grafickým rozhraním kód obsluhy (controller) od kódu aplikační logiky (model) a od kódu zobrazujícího data (view). Tím jednak aplikaci zpřehledňuje, usnadňuje budoucí vývoj a umožňuje testování jednotlivých části zvlášť.

Model je datový a zejména funkční základ celé aplikace. Je v něm obsažena aplikační logika. Jakákoliv akce uživatele (přihlášení, vložení zboží do košíku, změna hodnoty v databázi) představuje akci modelu. Model si spravuje svůj vnitřní stav a ven nabízí pevně dané rozhraní. Voláním funkcí tohoto rozhraní můžeme zjišťovat či měnit jeho stav. Model o existenci view nebo kontroleru neví. Pojem Model představuje celou vrstvu, nikoliv samostatnou třídu.

View, tedy pohled, je vrstva aplikace, která má na starost zobrazení výsledku požadavku. Obvykle používá šablonovací systém a ví, jak se má zobrazit ta která komponenta nebo výsledek získaný z modelu.

Controller je řadič, který zpracovává požadavky uživatele a na jejich základě pak volá patřičnou aplikační logiku (tj. model) a poté požádá view o vykreslení dat. Obdobou kontrolerů v Nette Framework jsou **presentery**.

Každý požadavek na aplikaci se dostane přes soubory index.php a bootstap.php do objektu `$application`. Objekt aplikace za pomocí *routeru* zpracuje HTTP požadavky (URL adresu a její GET parametry) a zavolá správný presenter a jeho akci kterou má vykonat. Například uživatel chce zobrazit na e-shopu produkt s id=123. V aplikaci je tento požadavek v presenteru Product, akce show s parametrem id (v Nette se to běžně zapisuje *Product:show id=>123*). Router je ten, který na

základě tohoto požadavku vygeneruje správnou url adresu (např. www.example.com/product/show/produkt-123.html) a naopak z této adresy dokáže zpátky zjistit, že se má zavolat akce Product:show id=>123. [10]

Presenter je objekt, který vezme požadavek přeložený routerem a vytvoří odpověď. Odpovědí může být HTML stránka, obrázek, XML dokument, soubor na disku, JSON, přesměrování nebo cokoliv potřebujete. Konkrétně *ProductPresenter* požádá model o data a ty poté předá do šablony k vykreslení. Tohle se zpravidla odehraje v metodě renderShow, kde slovo Show odpovídá názvu akce a parametr požadavku id bude předán jako parametr této metodě:

```
class ProductPresenter extends Nette\Application\UI\Presenter
{
    public function renderShow($id)
    {
        // získáme data z modelu a předáme do šablony
        $this->template->product = $this->model->getProduct($id);
    }
}
```

Obrázek 5: Kód presenteru a jeho akce

Po zavolání metody `render<view>(<params>)` přichází na řadu šablony. Šablona není nic jiného než samostatný php soubor, který má za úkol pouze pomocí základních *php* konstrukcí (*foreach, for, switch, ...*) vykreslovat HTML kód webu. Jak je vidět na příkladu viz. obrázek 5, proměnné presenter šabloně předává přes objekt `$this->template`. V tomto případě se k proměnné product přistupuje v šabloně už jen jako `$product`. [11]

Nette pro šablony nabízí svůj LatteFilter. Latte má podobnou syntaxi známějšímu systému smarty[5]. Latte se používá pro ulehčení práce a správné zabezpečení výstupu aplikace, pomocí tzv. maker. Například pro vygenerování odkazu na produkt e-shopu se použije do html kódu makro *{link…}* viz. obrázek 6, kde *$idProduct* je proměnná obsahující číslo produktu.

```
<a href="{link Product:detail $idProduct}">Detail produktu</a>
```

Obrázek 6: HTML kód s Latte makrem

Normální vykreslování proměnné se provádí pouze jeho zapsáním mezi složené závorky, takto: *{$variable}*. To ale neudělá pouze vypsání proměnné, ale ještě ji escapuje, čímž samovolně chrání proti hrozbě *Cross-site scripting* (XSS). Latte filter poskytuje programátorovi spoustu

[5] Smarty je šablonovací systém pro PHP oddělující vykreslovací šablony od logiky aplikace.

užitečných funkcí a nástrojů, jako například další makra, helpery (drobné užitečné funkce, přes které se výsledek přepíše), snippety apod. Více je v dokumentaci latte v [12].

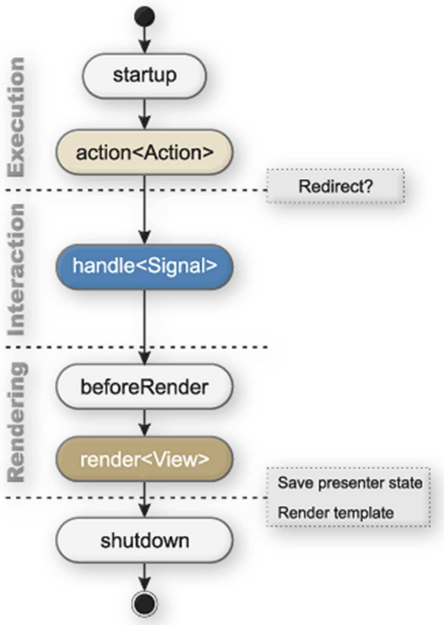
Celá naše aplikace je rozdělena do 6 presenterů, pro každou sekci jeden, které dědi od jednoho BasePresenteru. Ten má za úkol předávat do šablon i presenterů data, komponenty a metody, které jsou pro všechny presentery společné. Všechny tyto presentery jsou uloženy ve složce *app/presenters/*.

Každý presenter má také svou defaultní šablonu v Latte souboru (pro šablonovací systém Nette frameworku) */app/templates/<Název_presenteru>.default.latte* a případně ještě další (místo *default* je zde potom uveden název akce presenteru).

Pro celou aplikaci vystačil jeden jediný model. Jeho třída se jmenuje stručně Model a je uložena v *app/models/Model.php*. Ten zajišťuje převod dat z *HSConnection* (knihovny pro komunikaci s Backendem) do správného formátu a správných objektů. Např. instancuje třídy Timeslot, DateTime či Timewindow a stejně tak očekává vstupy v instancích různých tříd a třídě HSConnection pak předává jen surová data (název Timeslotu, datum ve správném formátu atp.)

3.2.2 Životní cyklus aplikace

Jak bylo psané dříve, na presenteru se volají metody render<View> (např. renderShow), které způsobí akci vykreslení šablony show s předanými proměnnými od modelu přes presenter až do šablony. Render metody však nejsou zdaleka jediné které je možné na presenteru zavolat.[10]



Obrázek 7: Životní cyklus Nette aplikace [10]

Metoda `startup()`, se zavolá ihned po vytvoření presenteru. Slouží například k inicializování proměnných, nebo k ověření uživatelského oprávnění.

Po ní se volá metoda `action<Action>` (s názvem akce, předané routerem podle URL adresy). Je to obdoba metod s prefixem `render`, akorát se neočekává, že bude vykreslovat šablonu. Může vykonat nějaký úkon, načíst data z databáze či ověřit uživatelské oprávnění a poté buďto přesměrovat uživatele jinam, vykreslit např. JSON a skončit, a nebo nic a životní cyklus pokračuje dál podle diagramu viz. obrázek 7.

Pokud nás *action* nikam nepřesměruje, nebo neukončí běh aplikace následuje vyřizování tzv. signálů. To mají za úkol metody s prefixem `handle<Signal>`. Používají se především pro ajax, či změnu stavu komponenty (presenteru). Volají se z odkazu v šabloně pomocí *{link signal! arg1,arg2…}*, kde signal je název signálu. Po jejich zavolání se očekává přesměrování nebo v případě ajaxu invalidace snippetů.

Metoda `beforeRender`, jak už název napovídá, se volá před už několikrát zmiňovanou metodou `render<View>()` a může obsahovat například nastavení šablony, předání proměnných společných pro více view a podobně.

Známé `render<View>()` obvykle nastaví do šablony potřebná data. Hned po ní se vyrenderuje příslušná šablona.

Jako poslední v řadě je `shutdown()`. Ta se obvykle používá pro odpojení od vzdálených prostředků (sockety, databáze, soubor, ...) a nebo například pro označení příspěvků za přečtené apod.

V průběhu životního cyklu se samozřejmě dají využívat standartní objektové vlastnosti třídy Presenter. Není problém vytvořit si atribut například *private $product* v action do něj načíst z databáze objekt, v handle na něm zavolat nějakou metodu, a v render ho předat šabloně k vykreslení.

Specialitou Nette je ovšem tzv. **persistentní parametr**. To je běžný atribut třídy, před který stačí napsat anotaci[6] *@persistent*. Tento attribut se poté automaticky přenáší do všech odkazů vygenerovaných přes Latte makro *{link …}*. To se dá skvěle použít například pro stránkování či řazení, ale i jiné věci.

V presenterech se ještě objevují metody začínající prefixem createComponent<name>(). Ty vytváří tzv. komponenty. Takovouto komponentou je například každý formulář. V této metodě se formulář vytvoří, definuje a tato metoda ho vrátí přes *return*. V latte šabloně se pak jen použije makro {control <name>} a na komponentě (v našem případě formuláři) se zavolá metoda render(), která zajistí jeho vykreslení do šablony. Dalšími komponentami můžou být například různé DataGridy, stránkovače, kalendáře apod. Používání komponent velmi zvyšuje přenositelnost kódu na další aplikace, což by v dnešní době měla být, především z ekonomických důvodů, velká priorita.

[6] Anotace je běžný blokový PHP komentář začínající dvěma hvězdičkami. Stejný zápis se používá například pro DoxyGen komentáře.

3.2.3 Třída HSConnection

Tato třída zajišťuje komunikaci Webového rozhraní z backendem HostStats pomocí socketů. Přes její konstruktor se předá adresa hosta a port na který se má připojit a volají se na ní metody pro získávání dat z HostStats serveru. Výčet metod koresponduje s komunikačním protokolem z kapitoly 2.3. jen jsou jeho názvy převedeny do camelCase a parametry se předávají jako jednotlivé parametry metody.

Například příkaz GET_TIMESLOT_DATA se zavolá metodou na objektu typu HSConnection takto:

```
$obj->getTimeslotData($timeslot, $profile, $filter, $limit, $sort, $asc);
```

Metoda zajistí připojení k serveru přes sockety, zaslání správného kódu typu požadavku a zaslání parametrů. Poté počká na odpověď od serveru a přijatá data vrátí v asociativním poli .

Tato třída se dá bez jakýchkoli problémů vzít a použít v jiné aplikaci. Do frontendu HostStats je připojená jako služba (service) přes konfigurační soubor Nette. Ten jí přímo z configu předá parametry do konstruktoru (host a port) a nabídne ji pod názvem *conn*. Ve všech presenterech se k ní poté dá přistupovat přes `$this->context->getService('conn')` nebo zkráceně `$this->context->conn`.

3.2.4 Třída Timeslot

Tato třída slouží k reprezentaci pětiminutových časových úseků používaných v HostStats, tzv. timeslotů .V aplikaci potřebujeme pracovat jejich časovým názvem a posouvat se o timesot dozadu či dopředu. Pro vyjádření Timeslotu ale nevystačí běžný DateTime z PHP, takže tato třída od něj dědí, a nabízí další potřebné metody.

Třída má konstruktor, který zvládne vzít jakýkoliv formát času (stejný jako DateTime) a navíc formát který používá HostStats (dvanáctimístného číslo RRRRMMDDHHMMSS, kde R je rok, M je měsíc atp.) Čas, který je do konstruktoru předán je také potřeba srovnat na 5-ti minutové úseky. Timeslot končící například 8 minut a 20 sekund samozřejmě neexistujte, proto se zaokrohlí na 5 minut 0 sekund.

Oproti DateTime byly také definovány nové metody. Nejdůležitější jsou `getName()`, která vrací název timeslotu ve formátu dvanáctimístného čísla, které se používá pro komunikaci s HostStats. Dále pak metody `getNext()` a `getPrev()`, které posouvají timeslot na další / předchozí (tzn. o 5minut dopředu / dozadu).

Tato třída se dá samozřejmě také použít v jakékoliv jiné aplikaci pracující s timesloty. V případě že je timeslot jinak dlouhý než 5 minut, stačí změnit statickou konstantu *LENGTH* na jiný počet vteřin.

3.2.5 Třída Timewindow

Pro historii komunikace jednoho konkrétního hosta je potřeba uvádět od kdy do kdy se mají data vypisovat. Tento časový úsek se nazývá časové okno – timewindow. Stejnojmenné třídě, která jej reprezentuje se v konstruktoru předávají dvě instance třídy *Timeslot* (timeslot od a timeslot do), mezi kterými chceme definovat časové okno. Z nich se automaticky dopočítá délka časového okna, ze které vycházejí v podstatě všechny důležité metody.

Metoda `getTitle()` zajišťuje slovní vyjádření délky okna ve *stringu*. Například „1 day“, „2 hours“, „last week“ nebo „last 2 hour“ pro minulé dvě hodiny do teď. Tyto informace se vypisují na webovém frontendu uživateli, pro lepší přehled než kdyby tam bylo pouze dvakrát datum.

Třída *Timewindow* opět disponuje mimo jiné metodami `getNext()` a `getPrev(),` které vrací další časové okno o stejné délce.

3.2.6 Třída Nettools

V aplikaci byly zapotřebí také různé nástroje pro práci s IP adresami a zjišťování informací o nich, Tato třída obsahuje statické metody pro zjištění WHOIS záznamu, pro zjištění názvu hosta podle adresy a také ověření validity formátu IPv4 a IPv6 adresy.

Metoda `Nettool:whois($ip, $host=null)` ke zjištění WHOIS záznamu používá standartní unixový příkaz *whois* s volitelným parametrem *-h <host>* , který slouží k určení whois serveru na kterém se má hledat. V našem případě se hledá na serveru whois.ripe.net (ip 193.0.6.135), který je nastaven v configu.

3.2.7 Zpracování chyb

Chyby jsou v celé aplikaci ošetřovány za pomocí vyjímek (tzv. exceptions) a try-catch bloků. Jediné odchytávané výjimky v presenterech jsou potomky třídy `HSCException` a TimewindowException, jejichž zprávy jsou poté zobrazovány uživateli ve formě flash message ve webovém rozhraní.

Potomky vyjímky `HSCException` generuje třída `HSConnection` v případě, že dojde k chybě při komunikaci ze serverem (`HSCConnectException`) nebo v případě že HostStats server vrátí zprávu začínající slovem „*ERROR:*“ (`HSCBadCommandException`). To se může stát, když pošleme špatný kód typu požadavku, špatné parametry požadavku nebo když server potřebuje ohlásit nějakou chybu (např. že nemá příslušná data).

`TimewindowException` vyvolává třída Timewindow v případě špatně zadaných parametrů konstruktoru.

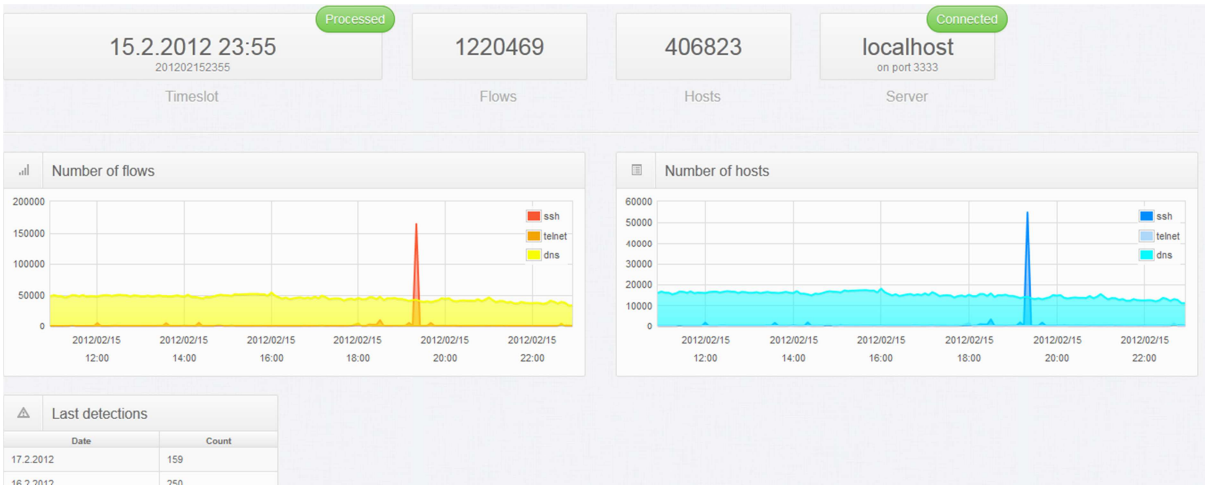
Všechny ostatní neodchycené vyjímky jsou chyby aplikace a v případě, že by se někde objevily, logují se do souboru */log/errors.log*.

3.3 Sekce Frontendu

Celý webový frontend je rozdělen do 5 sekcí. Přehled, detail timeslotu, IP mapa timeslotu, historie hosta a detektor útoků. Všechny tyto sekce jsou navzájem propojené odkazy a nebo se k nim dá přistoupit z hlavního menu.

3.3.1 Sekce Overview

Sekce „Overview“ neboli „Přehled“ je hlavní stranou aplikace. Je otevřená hned po zadání adresy, případně otevření pluginu v NfSenu.



Obrázek 8 Výřez sekce Overview

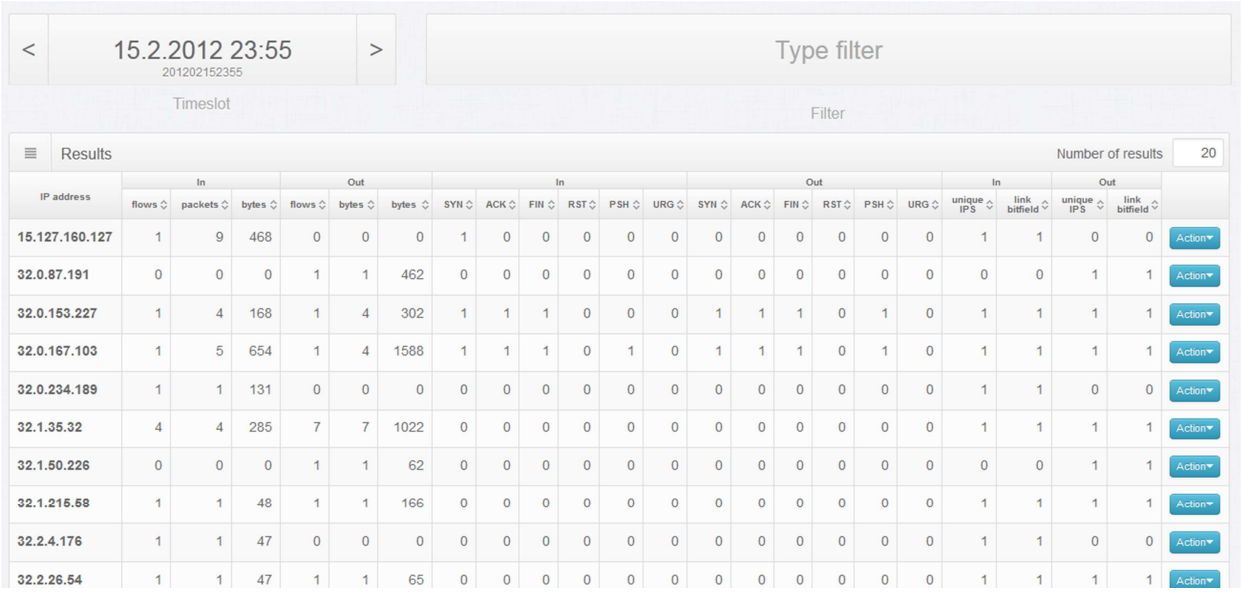
Jak je vidět na obrázku 8 je zde zobrazen aktuální načtený timeslot. Případně se u něj píše, jestli už je zpracovaný (zelené „Proccessed“) nebo jestli se právě zpracovává (modré „Processing“). Tzn. jestli se právě vytváří statistika. Vedle je poté počet toků a počet hostů v aktuálním timeslotu. Dále je ve vrchní části ještě informace ke kterému HostStats serveru (backendu) je frontend připojen či se k němu snaží připojovat.

Pod tímto „řádkem“ se stavem HostStats jsou dva grafy. Červeno-oranžový uvádí historii počtu toků v jednotlivých timeslotech za posledních 12 hodin, a modrý historii počtu hostů v těchto timeslotech.

Na stránce přehledu je ještě historie detekcí útoků. Je to ta drobná tabulka vlevo dole. Uvádí několik posledních dní, ve kterých byly detekovány nějaké útoky a jejich počet. Po kliknutí na příslušný řádek je uživatel samozřejmě přesměrován na stránku s výpisem oněch útoků.

3.3.2 Sekce Detail

V této sekci je vidět statistika kterou počítá HostStats. Ze všech toků v jednom timeslotu (tzn. 5ti minutovém časovém úseku) sečte příchozí/odchozí packety, jednotlivé typy packetů (SYN,ACK,FIN..) atd. Podle těchto čísel se poté detekují jednotlivé hrozby.



IP address	In			Out			In						Out						In		Out	
	flows	packets	bytes	flows	bytes	bytes	SYN	ACK	FIN	RST	PSH	URG	SYN	ACK	FIN	RST	PSH	URG	unique IPS	link bitfield	unique IPS	link bitfield
15.127.160.127	1	9	468	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
32.0.87.191	0	0	0	1	1	462	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
32.0.153.227	1	4	168	1	4	302	1	1	1	0	0	0	1	1	1	0	1	0	1	1	1	1
32.0.167.103	1	5	654	1	4	1588	1	1	1	0	1	0	1	1	1	0	1	0	1	1	1	1
32.0.234.189	1	1	131	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
32.1.35.32	4	4	285	7	7	1022	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
32.1.50.226	0	0	0	1	1	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
32.1.215.58	1	1	48	1	1	166	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
32.2.4.176	1	1	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
32.2.26.54	1	1	47	1	1	65	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Obrázek 9: Výřez sekce Detail

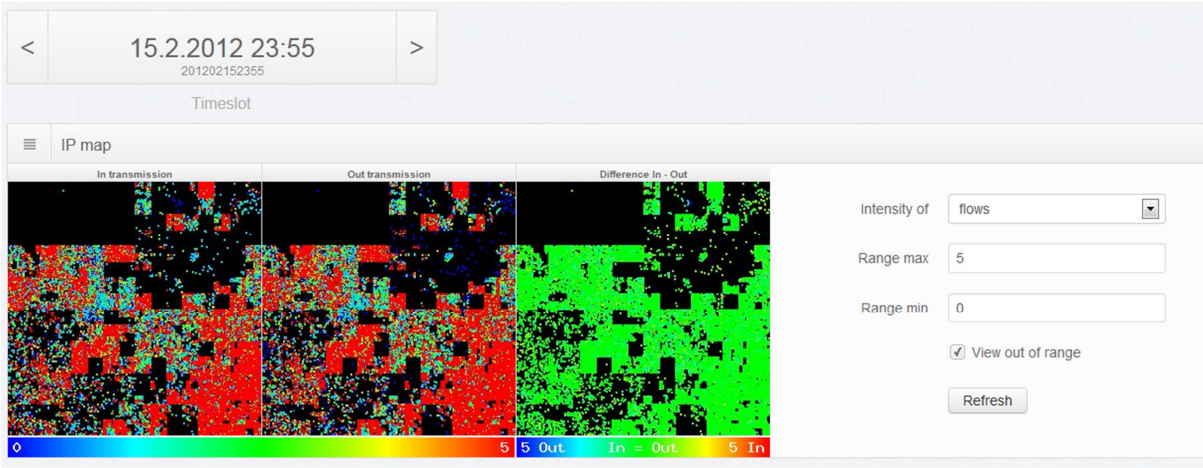
Zde, jak je vidět na obrázku 9, je možné vybrat příslušný timeslot (po kliknutí k dispozici kalendář s výběrem konkrétního času). Šipkami doleva a doprava je možné se pohybovat na další či předchozí timeslot (tzn o 5 minut dozadu či dopředu) a v tabulce uvidíte vždy seznam hostů (IP adres), kteří v tomto časovém úseku spolu na síti komunikovali.

Vzhledem k tomu, že takovýchto záznamů můžou být na síti desetitisíce, statisíce i víc (záleží na velikosti sítě), vypisuje se pouze N výsledku (počet se dá nastavit změnou čísla v pravém horním rohu tabulky). Těchto N výsledků samozřejmě můžeme seřadit podle příslušného sloupce, buďto od největšího nebo nejmenšího čísla. Dále se dají výsledky filtrovat zapsáním filtrovacího pravidla do kolonky „Filter“ v pravé horní části obrazovky. Tyto filtrovací pravidla mají formát např. `IN_ACK > 100 && OUT_ACK = 0.` Podrobněji jsou popsány v dokumentaci k HostStats serveru, který má také za úkol je rozpoznávat a validovat. V případě chyby se do frontendu vrátí chybová hláška která je poté zobrazena uživateli.

3.3.3 Sekce IP map

Zde je vidět vizualizace komunikujících hostů shromážděných podle prefixu IP adresy v tzv IP mapě. Je to obrázek 256x256 pixelů, kde každý pixel odpovídá jednomu /16 prefixu (prvních 16bitů z každé IP adresy hosta. Na obrázku 10 je vidět, že IP mapy jsou zde tři. Jedna pro příchozí přenosy, druhá

pro odchozí a třetí pro jejich rozdíl. Na té je sytě modře vidět, když některý rozsah IP adres data jen vysílal a sytě červeně když jen přijímal. Formulář vpravo nastavuje, jestli chceme intenzitu přenosu zobrazovat podle počtu toků, paketů nebo bytů. Dále se nastavuje rozsah zobrazení, který je vidět zároveň pod mapami. Pokud je intenzita menší/větší než rozsah, použije se minimální/maximální hodnota. Pokud pixely s intenzitou mimo rozsah vůbec nechceme vidět, stačí odškrtnout „View out of range“.



Obrázek 10: Výřez sekce IP map

V této vizualizaci můžou být vidět některé typy útoků. Například při DDOS útoků s podvrženými (náhodnými) zdrojovými adresami se rozsvítí velká část bitmapy.

Tyto mapy jsou inspirovány projektem Internet Census 2012[7]. Stejně jako tam jsou zde prefixy IPv4 adres vykreslovány do bitmapy podle Hilbertovy křivky, takže po sobě jdoucí čísla (prefixy adres) se seskupují do čtvercových oblastí.

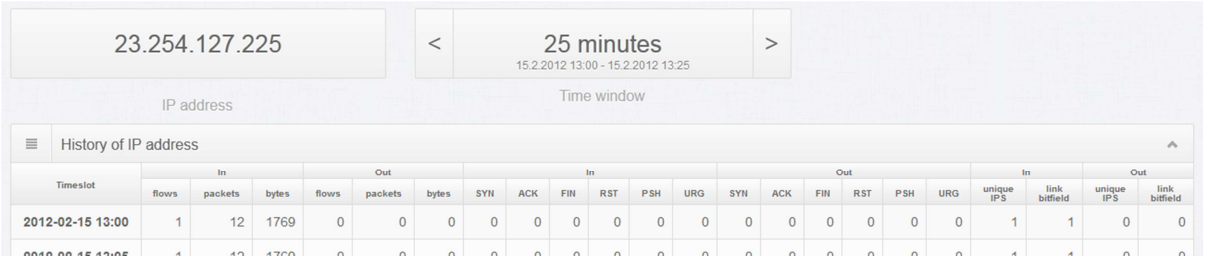
3.3.4 Sekce History

Pokud potřebujeme naopak vědět, jak na síti komunikoval jeden host v průřezu času, slouží k tomu sekce History.

V horní části okna je možné zadat IP adresu hosta, o kterém chceme vědět informace a vedle toho vpravo je políčko pro zadání časového okna. Pole pro výběr časového okna (timewindow) poskytuje možnost použít předdefinované hodnoty (minulá hodina, 6 hodin, 12 hodin ..) nebo zadat pomocí kalendáře přesný čas od kdy do kdy chceme vidět výsledky. Dále je pak možné časovým oknem hýbat dopředu a dozadu.

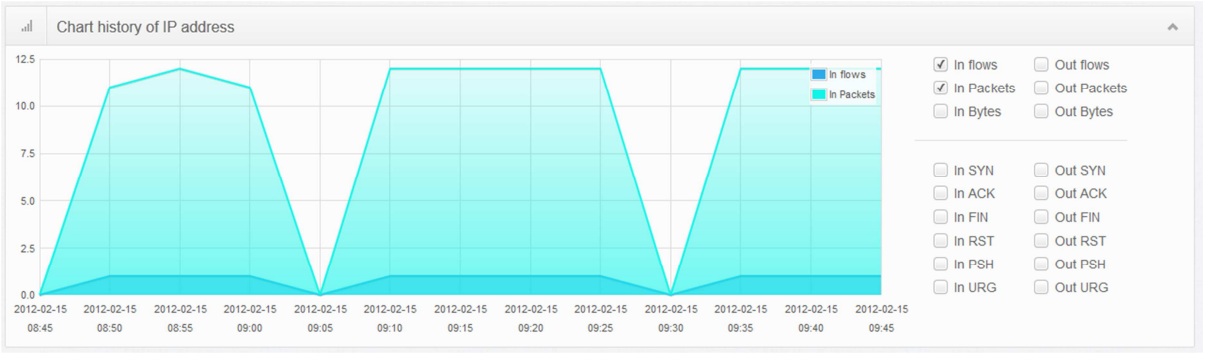
Pod těmito boxy na zadání „filtru“ se zobrazují samotná data v tabulce. Tabulka obsahuje, kromě prvního, stejné sloupce jako v sekci Details. V prvním sloupci se zobrazuje timeslot, ze kterého jsou data v tomto řádku. Timeslot je samozřejmě prolinkovaný na jeho detail.

[7] http://internetcensus2012.bitbucket.org/



Obrázek 11 Výřez sekce History - filtr + tabulka

Pod tabulkou následuje box s grafem. Do tohoto grafu je možné pomocí zaškrtávacích políček napravo vynést libovolné sloupce z tabulky v závislosti na čase. Samozřejmě má smysl porovnávat pouze vstupní a výstupní hodnoty stejné veličiny, protože jinak jsou čísla diametrálně rozdílná (např. počet toků a počet přenesených bytů v nich je několikanásobný).



Obrázek 12 Výřez sekce History - grafy

Úplně na konci stránky se nachází výpis WHOIS[8] záznamu zkoumané IP adresy. Tento výpis se získává z unixového konzolového nástroje „whois“ a jeho syntaxe je zvýrazněna pomocí volně dostupné knihovny GeSHi.

3.3.5 Sekce Detectors

V této sekci se zobrazují útoky, které HostStats detekoval. V horní části stránky je opět možnost zadat datum, tentokrát celý den, ve kterém chceme zobrazit odhalené útoky. Pomocí šipek doleva a doprava se dá opět pohybovat po dnech zpět a vpřed.

[8] WHOIS – databáze evidující údaje o majitelích internetových domén a IP adres

Obrázek 13 Výřez sekce Detectors

Tabulka obsahuje následující sloupce:

- Timeslot – časový úsek ve kterém byl útok detekován
- Type – typ útoku (Horizontální či vertikální skenování portů, Dos útoky atd..)
- Protocol – Protokol na kterém byl útok veden
- Source IP – adresa hosta, ze kterého byl útok veden
- Destination IP – adresa hosta na kterého byl útok veden
- Source port – Zdrojový port
- Destination port – Cílový port
- Intensity – Intenzita útoku (Význam se liší se podle typu útoku. U portscan je to např. počet proskenovaných adres/portům u DoS útoku počet toků, apod).
- Note – Poznámka, kterou posílá detektor v HostStats

4 Testování

Testování systému před jeho nasazením do provozu je velmi důležitá součást vývoje. Pro testování webového rozhraní HostStats jsem vybral několik aspektů, které je vhodné testovat. Testování těchto oblastí a jejich výsledky jsou uvedeny v této kapitole.

4.1 Oblasti testování

4.1.1 Testování funkčnosti

Nejdříve je potřeba otestovat funkčnost. Jestli aplikace dělá to co má. Zobrazuje správně všechna očekávaná data. A také jestli ve všech případech provede to co má a neskončí s nějakou chybou či neodchycenou vyjímkou.

4.1.2 Testování použitelnosti

Úkolem tohoto typu testování je odhalit jestli je aplikace uživatelsky přívětivá. V případě, že se uživatel ztrácí v navigaci či neví jak udělat libovolnou akci, je něco špatně.

Testování proběhlo na malé skupině lidí orientující se v oblasti IT, pro které je tento systém také zamýšlený. Pozoroval jsem především, jakým problémům uživatel čelí a jestli jsou všechny prvky rozhraní uživateli srozumitelné.

4.1.3 Testování bezpečnosti

Testování bezpečnosti systému je jedna z nejdůležitějších oblastí. I systém HostStats pracuje s velmi citlivými daty. Webové rozhraní ovšem nedisponuje žádnými prvky ochrany jako je autorizaci či autentizace. Počítá se s tím, že nebude na veřejně přístupném serveru a tím pádem svěřuje tuto autorizační službu operačnímu systému serveru, na kterém běží.

4.2 Na čem bylo testováno

Testovaný systém běžel na školním serveru ANT-STUD (ant-stud.fit.vutbr.cz). Na stejném stroji byl nainstalován NfSen, HostStats server (backend) i naše testované webové rozhraní.

Serverové zázemí použité k testování:

- Operační systém: *Linux ant-stud 2.6.18-194.26.1.el5 #1 SMP Tue Nov 9 12:54:20 EST 2010 x86_64 x86_64 x86_64 GNU/Linux*
- Webový server: *Apache/2.2.3 (CentOS) Server*
- PHP: *PHP 5.3.3 (cli) (built: Jun 27 2012 12:25:48)*

Testované webové prohlížeče

- Mozzilla Firefox 20.0.1
- Google Chrome Verze 26.0.1410.64 m
- Microsoft Internet Explorer v.9 a 10.

4.3 Výsledky testování

Funkčnost systému byla ověřována a kontrolována již během implementace a samozřejmě i po dokončení. Za pomocí diagnostických nástrojů poskytovaných frameworkem byly všechny chyby odstraněny a vyladěny. V krajním případě, že by se ještě nějaká objevila, zapíše se do logu */app/log/error.log* a odešle se na email nastavený v konfiguraci.

Na základě pozorování a reakcí uživatelů při testování použitelnosti byly lehce upraveny některé funkční prvky rozhraní. V této fázi se rozhraní jeví uživatelům jako velmi příjemné a použitelné.

Bezpečnost systému zajišťuje to, že je umístěný na serveru na školní síti, kam mají přístup jen oprávnění uživatelé.

5 Možnosti dalšího rozšíření

Možností dalšího rozšíření, které by obohatily tento systémem, se objevilo hned několik.

Pro širší využití systému by se určitě hodilo implementovat **autorizaci uživatelů**. V případě, že by každý uživatel systému měl svůj login a heslo mohla by být aplikace nasazena i na veřejně přístupném webovém serveru (datová komunikace s backendem by samozřejmě musela být stále důkladně zabezpečena.).

Dále by se dalo uvažovat nad **jazykovými mutacemi**. Aplikace by poté byla příjemnější uživatelům z různých zemí. Nette Framework poskytuje pro jazykové překlady spoustu užitečných nástrojů.

Pokračovat lze také samozřejmě v různé **vizualizaci dat**. Ta by v případě tohoto systému měla uživateli pomáhat odhalovat bezpečnostní hrozby pouhým okem.

V případě širšího nasazení by šlo aplikaci rozšířit o **přepínání mezi více servery**. Nyní se připojuje jen k jednomu serveru, který je nastavený v konfiguračním souboru.

6 Závěr

Tato bakalářská práce se zabývá návrhem, implementací a testováním webového rozhraní pro systém detekce síťových anomálií s názvem HostStats. Webové rozhraní je možné provozovat jako plugin do monitorovacího systému NfSen, ale i jako samostatnou webovou aplikaci, závislou pouze na serverovém zázemí (webový server + PHP5).

Webové rozhraní systému komunikuje s HostStats serverem pomocí socketů. To v případě potřeby umožňuje mít webový server na jiném počítači než HostStats server. Komunikace probíhá princpem požadavek – opověd. Webové rozhraní pošle pomocí PHP přes sockety typ požadavku a jeho parametry, a ze socketu přečte odpověď, kterou zpracuje a ve webovém prohlížeči zobrazí uživateli v uživatelsky přívětivé formě.

Celá aplikace byla na straně webového serveru naprogramována ve skriptovacím jazyce PHP za použití nejnovější verze Nette Frameworku. Na straně klienta se počítá s moderním prohlížečem podporujícím HTML5, CSS3 a JavaScript. Při progamování javascriptu byla použita volně dostupná knihovna jQuery a některé její pluginy.

Systém splňuje požadavky na něj kladené a při uživatelském testování nebyl nalezen žádný problém při jejím užívání. Aplikace je psána způsobem, aby nebylo složité ji rozšířit či upravit. V závěrečné kapitole jsou rozvedeny některé možné způsoby dalšího rozšíření. V této fázi aplikace ovšem plně implementuje všechny funkce, které HostStats server poskytuje.

Literatura

[1] B. Claise: *Cisco Systems Netflow Services Export Version 9*. RFC 3954, 2004.

[2] *NfSen - Netflow Sensor*. [online], [cit 2013-04-20]. Dostupné z: http://nfsen.sourceforge.net/

[3] DANĚK, Petr. Velký test PHP frameworků. In: *Root.cz* [online]. 2008 [cit. 2013-04-20]. Dostupné z: http://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/

[4] PILGRIM, Mark. *Dive Into HTML5*. O'Reilly Media, 2010. ISBN 0596806027.

[5] *Kaskádové styly* [online],[cit. 2013-04-20]. Dostupné z: http://www.w3.org/Style/CSS/Overview.cs.html.

[6] *Nette framework* [online]. 2008 [cit. 2013-04-20]. Dostupné z: http://nette.org/

[7] *NEON* [online]. 2010 [cit. 2013-04-20]. Dostupné z: http://ne-on.org/

[8] *Bootstrap* [online]. [cit. 2013-04-20]. Dostupné z: http://twitter.github.io/bootstrap/

[9] AJAX & snippety. *Nette Framework* [online]. [cit. 2013-04-20]. Dostupné z: http://doc.nette.org/cs/ajax

[10] MVC aplikace & presentery. *Nette Framework* [online]. [cit. 2013-04-20]. Dostupné z: http://doc.nette.org/cs/presenters

[11] Šablony. *Nette Framework* [online]. [cit. 2013-04-20]. Dostupné z: http://doc.nette.org/cs/templating

[12] Výchozí Latte makra. *Nette Framework* [online]. [cit. 2013-04-20]. Dostupné z: http://doc.nette.org/cs/default-macros

Seznam použitých zkratek

HTML	HyperText Markup Language
SGML	Standart Generalized Markup Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
CSS	Cascade Style Sheets
PHP	Hypertext Preprocesor
WWW	Word Wide Web
IP	Internet Protokol
NfSen	Netflow sensor
FW	Framwork
AJAX	Asynchronous JavaScript and XML
DOM	Document Object Model
RRD	Round-Robin Database
URL	Uniform Resource Locator

Seznam příloh

Příloha A. CD s kompletními zdrojovými kódy a dokumentací

Příloha A - CD s kompletními zdrojovými kódy a dokumentací

Na přiloženém CD jsou k dispozici následující soubory a adresáře

`src/`	Adresář se zdrojovými kódy aplikace
`readme.txt`	Základní informace o aplikaci
`install.txt`	Soubor s popisem instalace aplikace
`bachelor_thesis.pdf`	Elektronická verze tohoto dokumentu
`bachelor_thesis.docx`	Elektronická verze tohoto dokumentu