VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALYTICKÉ WEBOVÉ PROSTŘEDÍ PRO ZPRACOVÁNÍ ZACHYCENÉ SÍŤOVÉ KOMUNIKACE

NETWORK FORENSIC ANALYTICAL WEB-BASED PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. TOMÁŠ AMBROŽ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAN PLUSKAL**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Ambrož Tomáš, Bc.**
Program: Informační technologie Obor: Počítačové sítě a komunikace
Název: **Analytické webové prostředí pro zpracování síťové komunikace**
Network Forensic Analytical Web-Based Platform
Kategorie: Web

Zadání:

1. Seznamte se s aktuálním řešením zpracování síťové komunikace nástrojem Netfox Detective.
2. Proveďte návrh webového analytického prostředí pro zobrazení detailů síťové komunikace a extrahovaných informací z aplikačních protokolů.
3. Implementujte s využitím Dockeru a DotVVM.
4. Proveďte zhodnocení výsledků z pohledu uživatelské přívětivosti, ergonomie práce a srovnejte s alternativními nástroji.

Literatura:

- Strauss, D. (2017). *C# 7 and .NET Core Cookbook*. Birmingham: Packt Publishing.
- Price, M. (2017). *C# 7 and .NET Core modern cross-platform development: create powerful cross-platform applications using C# 7, .NET Core, and Visual Studio 2017 or Visual Studio Code*. Birmingham, UK: Packt Publishing.
- Unger, R., & Chandler, C. (2012). *A Project Guide to UX Design: For user experience designers in the field or in the making*. New Riders.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Pluskal Jan, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 28. října 2018

Abstrakt

V současnosti probíhá velká část komunikace skrze počítačové sítě. Objem této komunikace se každým rokem zvětšuje. To má za následek, že se zvyšují i nároky na výpočetní výkon. Za účelem zvýšení výpočetního výkonu se vyplatí proces zpracování komunikace pro forenzní účely paralelizovat. Výzkumná skupina NES@FIT v jednom ze svých projektů vytvořila nástroj Netfox Detective. Z tohoto nástroje je v plánu vytvořit distribuovaný systém pro zpracování zachycené komunikace za účelem forenzní analýzy. Jednou z částí je rozhraní, pomocí kterého by se distribuovaný systém ovládal. V této diplomové práci se budu zabývat tvorbou webového rozhraní pro nástroj Netfox Detective, který je v současné době desktopovou aplikací. Webové rozhraní by se poté dalo po drobnějších úpravách použít pro paralelizovanou variantu aplikace. Webové rozhraní bude zprostředkovávat stejné informace jako desktopová varianta. Pro získání informací pro forenzní analýzu bude využívat stejně jako desktopová varianta framework Netfox Framework. Výhoda webového rozhraní oproti desktopové variantě uživatelského rozhraní je v tom, že uživateli k tomu, aby mohl přistoupit k webovému rozhraní, postačuje zařízení s webovým prohlížečem. Což znamená, že uživatel může pracovat na jakémkoliv operačním systému.

Abstract

At present a big part of communication passes through computer network. Amount of the communication increases every year. That is why the claims on computing power raise. The procedure of the communication processing for forensic purposes is worth paralleling to increase the computing power. Research group NES@FIT created an instrument Netfox Detective in one of its projects. From this instrument it is planned to create a distributed system for the processing of intercepted communication for the purpose of forensic analysis. Interface is one of the parts, with its help the distributed system will be operated. In my theses I will concern with the creation of the web interface for the instrument Netfox Detective which is a desktop application presently. Web interface, after little modifications, will be used for paralleled version of application. Web interface will mediate the same informations as the desktop version. To obtain information for forensic analysis it will use framework Netfox Framework identically as the desktop version. Advantage of web interface compared to the desktop version is that a user who approaches web interface will need a device with web browser. It means that a user can work with any operation system.

Klíčová slova

Netfox Detective, DotVVM, HTML, Bootstrap, HangFire

Keywords

Netfox Detective, DotVVM, HTML, Bootstrap, HangFire

Citace

AMBROŽ, Tomáš. *Analytické webové prostředí pro zpracování zachycené síťové komunikace*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal

Analytické webové prostředí pro zpracování zachycené síťové komunikace

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jana Pluskala. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Ambrož
21. května 2019

Poděkování

Rád bych poděkoval Ing. Janu Pluskalovi za vedení mojí diplomové práce.

Obsah

Kapitola 1

Úvod

Každým rokem objem komunikace na internetu neustále roste, ať už se jedná například o komunikaci přes chatové služby, prohlížení webový stránek či emailovou komunikaci. Větší objem dat více zaměstnává vyšetřovatele, kteří analyzují zachycený síťový provoz za účelem získání forenzních informací o případné trestné činnosti. Tím pádem vzniká poptávka po nástrojích, jenž by usnadnily zpracování a analýzu zachycené síťové komunikace.

Problému se zpracováním a zobrazením extrahovaných dat se už nějakou dobu věnuje výzkumná skupina NES@FIT na Fakultě informačních technologií Vysokého učení technického v Brně v jednom ze svých projektů. Jedná se o projekt Netfox, ve kterém výzkumná skupina spolupracuje s Ministerstvem Vnitra České republiky. Projekt se zabývá vývojem nástroje pro získávání forenzních dat pro vyšetřování trestné činnosti orgány činnými v trestním řízení. Nástroj k tomuto určený se nazývá Netfox Detective. Jedná se o desktopovou aplikaci. Aplikace umí ze vstupních souborů se zachycenou komunikací extrahovat informace pro forenzní analýzu.

Cílem diplomové práce je vytvořit webové rozhraní, které by bylo alternativou pro grafické uživatelské rozhraní desktopové aplikace. Aby webové rozhraní poskytovalo stejná data jako desktopová varianta, tak bude používat pro extrahování dat k forenzním účelům framework Netfox Framework. Na framework je napojena i desktopová aplikace. Jednou z výhod webového rozhraní oproti desktopové verzi bude možnost spolupracovat na vyšetřování. Dále uživatelé nemusí řešit na jakém zařízení k webovému rozhraní přistupují, protože jim k tomu bude postačovat webový prohlížeč.

Prvním krokem řešení bude prozkoumání stávajících nástrojů, čímž se práce zabývá v kapitole 2. Zjistíme, jak dané nástroje vypadají, co uživateli poskytují za užitečné informace a s jakou komunikací si dokážou poradit.

Kapitola 3 diplomové práce se věnuje popsání současného stavu implementace aplikace Netfox Detective. Budou zde zmíněny technologie, které aplikace využívá pro implementaci. Dále budou vysvětleny návrhové vzory, které se v implementaci budou vyskytovat a je nutné je vysvětit pro pochopení, jak celá aplikace funguje. Nakonec se budeme v kapitole zabývat funkcemi některých důležitých komponent a postupem zpracování vstupního souboru se zachycenou komunikací na extrahovaná data užitečná pro orgány činné v trestním řízení.

Na první dvě kapitoly naváže kapitola 4, která pojednává o specifikaci požadavků webového rozhraní. Jinými slovy, co budou moci uživatelé v jednotlivých rolích provádět v rámci systému. Dále se také zmíní postup, jakým by uživatel měl postupovat při vyšetřování a návrh databáze pro správu uživatelů a vyšetřování.

Po návrhu se směle pustíme do popisu implementace webového rozhraní. V popisu implementace budou zmíněny použité frameworky a knihovny pro použití v implementaci. Hlavní částí kapitoly 5 je propojení webového rozhraní s Netfox Framework.

Ke konci práce bude zmíněno, jak probíhá testování jednotlivých částí vytvořeného webového rozhraní a výsledné řešení porovnáme s již existujícími nástroji. K porovnání nástrojů bude použit cvičný případ vyšetřování.

Kapitola 2

Nástroje pro forenzní analýzu síťového provozu

Nejprve definuji, co to vlastně forenzní analýza síťového provozu je. Lze ji definovat tak, že je to proces zachytávání, sběru a analýzy dat síťového provozu za účelem shromažďovaní informací, důkazů pro soudní řízení či detekci narušení systému. Pro zpracování síťového provozu právě za účelem forenzní analýzy existuje spousta nástrojů. Umožňují zachytávat síťový provoz, analyzovat zachycenou komunikaci.[15, 6, 20]

Například z VoIP provozu nám tyto nástroje jsou schopny říct, kdo komu volal, jaký kodek byl použit pro kódování hovoru či nám dokáží daný hovor přehrát. V kapitole budou některé příklady těchto nástrojů zmíněny. Ze zmíněných nástrojů poté budeme vycházet u návrhu webového rozhraní a porovnáme s nimi výsledné řešení.

2.1 Wireshark

Wireshark je open source desktopová aplikace. Slouží jako nástroj pro zachytávání, filtrování a analýzu síťového provozu. Aplikaci lze spustit na operačních systémech Windows, Linux i MacOS. Díky jednoduše použitelnému grafickému uživatelskému rozhraní, rozšířeným možnostem filtrování a podpoře PDML (Packet Details Markup Language) je Wireshark velmi užitečný nástroj pro vyšetřovatele v oblasti analýzy zachycené síťové komunikace.[3]

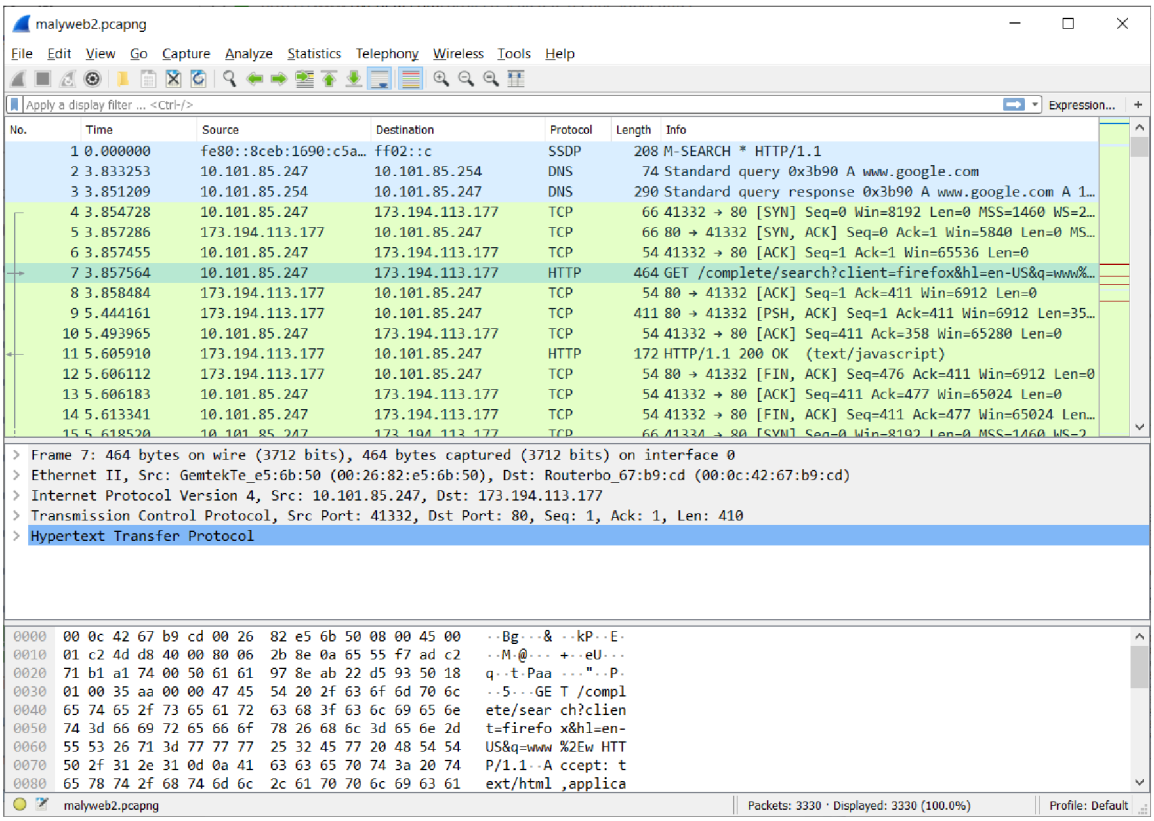
Wireshark umožňuje zachytávat síťový provoz na jakémkoliv systémovém síťovém rozhraní, jen pro to musí mít přiřazena dostatečná systémová oprávnění. Při zachycení komunikace se poté zobrazuje v reálném čase. Po ukončení zachytávání komunikace lze zachycený provoz uložit do souboru. Wireshark, jak již bylo výše řečeno, umožňuje analýzu síťového provozu. Dokáže interpretovat a zobrazit detailní informace o daném protokolu u jednotlivých paketů.

Toto zpracování jednotlivých paketů zajišťuje disektor. Pro každý podporovaný protokol existuje disektor. Také je možné si tento disektor naimplementovat vlastní pro protokol, který například není podporovaný v základu. Dále Wireshark umožňuje například analýzu VoIP provozu a případné přehrání hovoru, výpočet statistik pro DNS, HTTP atd.[3]

Pokud se podíváme na obrázek 2.1 s ukázkou aplikace Wireshark, tak si můžeme všimnout, že se skládá ze tří panelů [3, 19]:

- Seznam se zachycenými pakety — v tomto panelu se zobrazují pakety, které byly zachyceny. U každého paketu je uvedeno, kdy byl paket zachycen, zdrojová a cílová IP adresa, protokol, velikost a stručné shrnutí o přenášených datech.

- Detail paketu — tady pak lze očekávat detailní informace o všech obsažených protokolech v paketu, který je označený v prvním panelu.
- Bajty paketu — zde se zobrazují bajty paketu v hexadecimální a ASCII podobě.

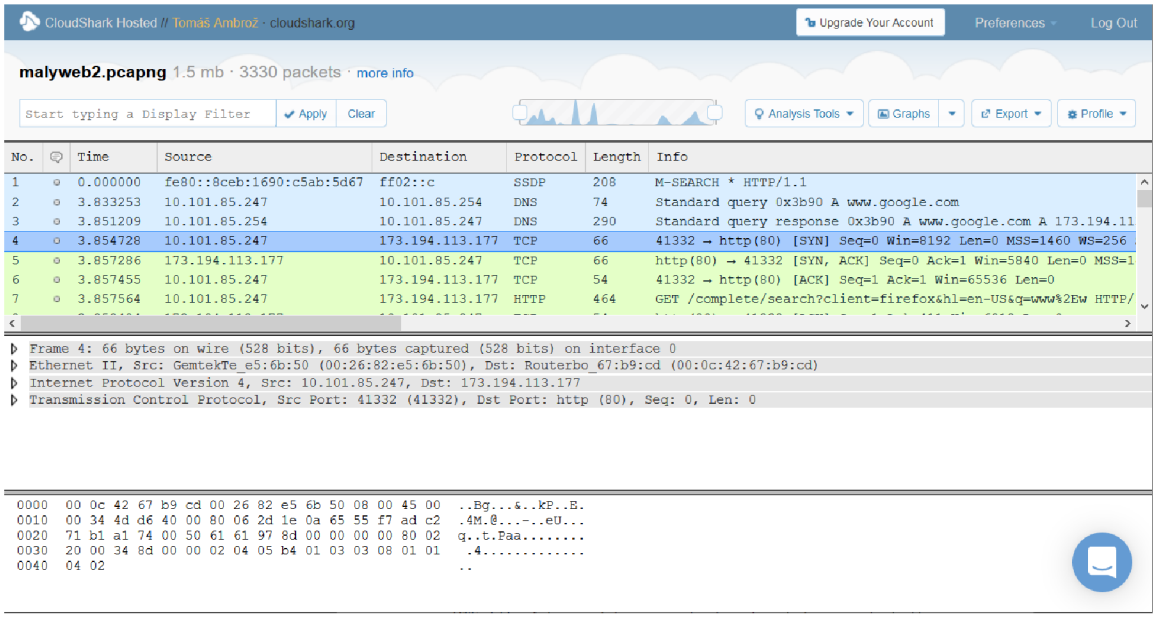


Obrázek 2.1: Ukázka desktopové aplikace Wireshark.

2.2 Cloudshark

Cloudshark je webová služba, která zobrazuje zachycenou síťovou komunikaci ze souboru podobně jako Wireshark viz obrázek 2.2. Mimo jiné patří mezi oblíbené nástroje pro sdílení zachycené komunikace s ostatními uživateli. Po nahrání souboru s komunikací na server se může sdílet pomocí odkazu, který rozešle kolegům, s nimiž pak uživatel může na analýze provozu spolupracovat.[19]

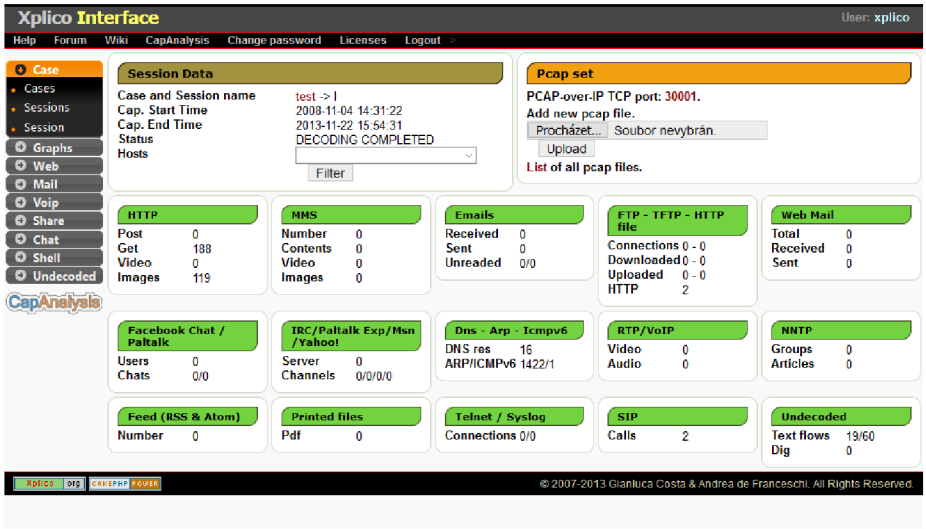
Jak Wireshark, tak i Cloudshark má pár nástrojů pro analýzu síťového provozu. Nabízí možnost vypsat jednotlivé koncové body v komunikaci a konverzace mezi nimi. Ze síťových protokolů podporuje analýzu například HTTP, DNS, SIP a RTP. Z analýzy HTTP protokolu dokážeme zjistit, jaké požadavky a odpovědi na ně se v komunikaci nacházejí. Dále jsme schopni zobrazit obsah přenášeného souboru například obrázek. U SIP protokolu dokáže určit všechny důležité informace o hovoru (volajícího, volaného, dobu trvání hovoru atd.). Analýza RTP nám pak dovoluje si obsah proběhlého hovoru poslechnout.



Obrázek 2.2: Ukázka webové služby Cloudshark.

2.3 Xplico

Xplico je open source nástroj pro analýzu síťového provozu, jenž dovoluje vyšetřovateli extrahovat data ze zachycené komunikace. Xplico dokáže extrahovat informace běžně používaných protokolů jako jsou například HTTP, SIP, IMAP, SMTP a DNS. Jelikož Xplico se ovládá pomocí webového rozhraní, bude popsáno trochu blíže. Bude se nám to hodit pro porovnání s naimplementovaným webovým rozhraním v této práci.[5]



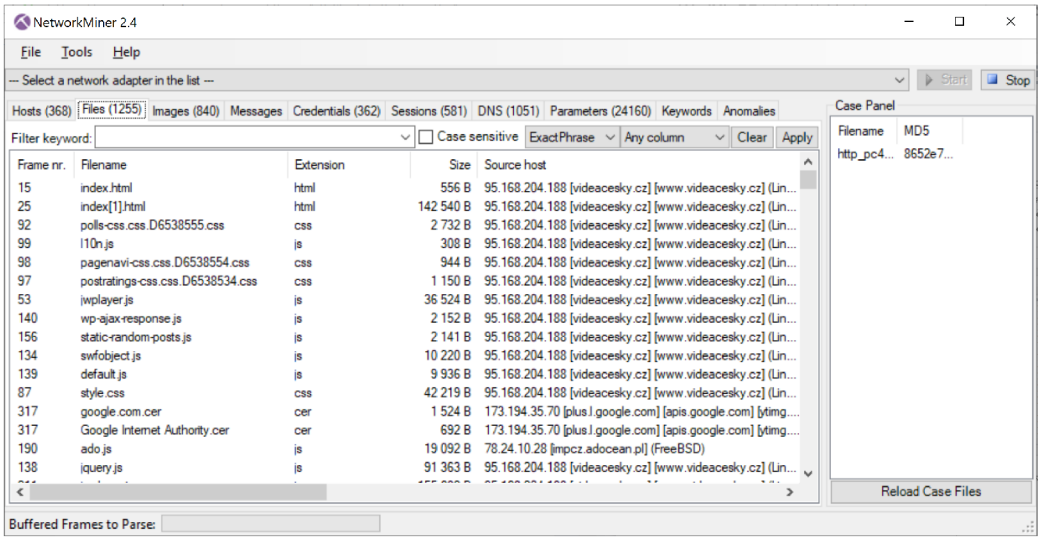
Obrázek 2.3: Ukázka webového rozhraní Xplico.

Aby mohl vyšetřovatel začít vyšetřovat, musí se nejprve přihlásit. Výchozí uživatelské jméno po instalaci je *xplico* s heslem taktéž *xplico*. Poté si vyšetřovatel musí založit nový případ, pokud případ již existuje stačí kliknout na název případu. Následně se musí v případu vytvořit nové sezení. Po vytvoření a otevření sezení se zobrazí úvodní stránka viz obrázek 2.3. V horní levé části se nacházejí obecné informace o sezení a případu. V pravé horní části se pak nachází seznam s nahranými soubory se zachycenou komunikací a formulářové pole pro přidání vstupních souborů s komunikací. Ve spodní půlce se pak zobrazují informace o extrahovaných datech u konkrétních protokolů. Vyšetřovatel tedy musí nahrát soubor se zachycenou komunikací, který se po nahrání začne zpracovávat. Pro zobrazení aktuálních dat po zpracování vstupního souboru je potřeba stránku obnovit. Neděje se tak automaticky. Pro zobrazení extrahovaných dat dané kategorie vyšetřovatel přistoupí přes menu, které se nachází na levé straně. Zde si pak může zvolit například zobrazení emailů, kde se mu zobrazí odesilatel, příjemci, obsah zprávy a případné přílohy. Nebo si může zvolit VoIP, který se dělí na exportovanou RTP komunikaci, kde si vyšetřovatel může hovor přehrát, a na signalizaci hovoru pomocí SIP protokolu.

2.4 NetworkMiner

NetworkMiner je desktopová aplikace primárně určená pro operační systém Windows. Lze jej ale spustit i na operačním systému Linux za použití Mono Frameworku. Využívá se pro analýzu síťového provozu. Dostupné jsou dvě verze. Jedna verze je dostupná zdarma s omezenou funkcionalitou oproti druhé verzi nazvané Professional. Data pro analýzu se buď dají vložit formou souboru se zachycenou komunikací nebo přímo z aktuálně probíhající komunikace na daném rozhraní.

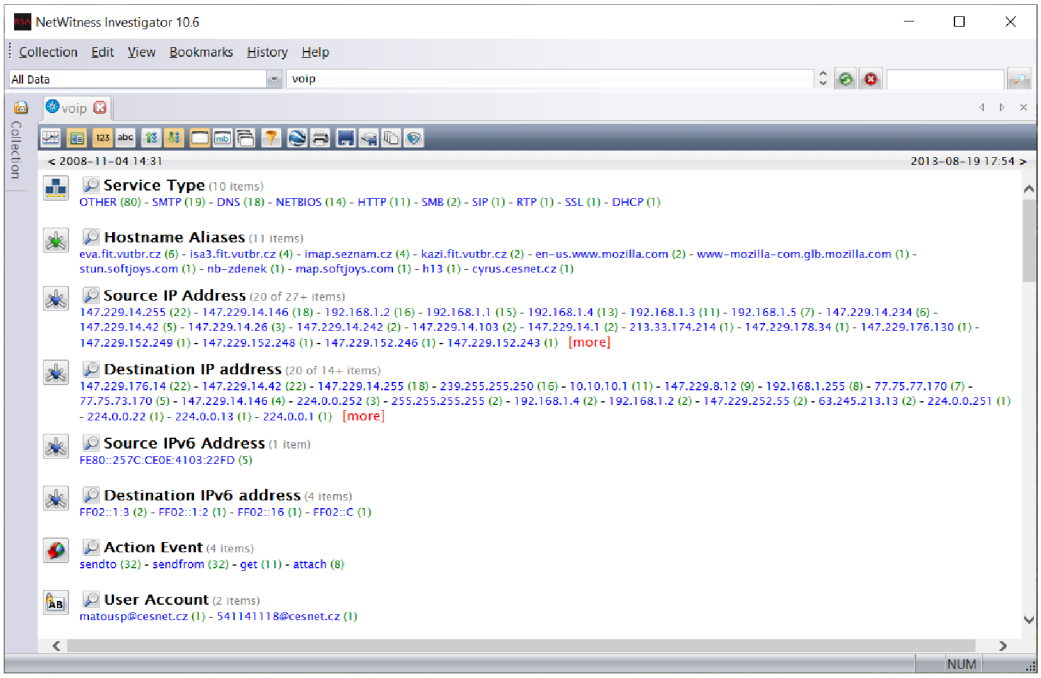
NetworkMiner podporuje analýzu protokolů HTTP, FTP, TFTP, POP3, IMAP, SMTP, SMB, SMB2. Z těchto protokolů dokáže vyexportovat přenášené soubory, obrázky, zprávy či přístupové údaje. Ve verzi Professional navíc umí extrahovat VoIP hovor, u kterého nám pak je schopen zobrazit kdo komu volal a přehrát obsah proběhlého hovoru. Jak si můžeme všimnout na obrázku 2.4, jednotlivé kategorie exportů jsou rozděleny pomocí záložek.



Obrázek 2.4: Ukázka aplikace NetworkMiner.

2.5 NetWitness Investigator

NetWitness Investigator je desktopová aplikace určená pro operační systém Windows. Stejně jako Wireshark nebo NetworkMiner umožňuje vložit soubor se zachycenou komunikací nebo zachytávat komunikaci přímo ze síťového rozhraní. Oproti ostatním nástrojům v této kapitole, probíhá analýza trochu jiným způsobem. Po vložení vstupního souboru se uživateli nezobrazí jednotlivé rámce či exportovaná data, ale zobrazí se seznam identifikátorů, které se vyskytují v zachycené komunikaci. Jako identifikátory zde můžeme brát IP adresy koncových uzlů nebo názvy protokolů či emailové adresy, pokud se v zachycených datech vyskytuje nějaká emailová komunikace. Pomocí těchto identifikátorů může uživatel filtrovat komunikaci, jenž ho zajímá. Jak filtrování v aplikaci vypadá můžeme vidět ze snímku 2.5. Vyfiltrovanou komunikaci můžeme uložit do nové kolekce nebo vyexportovat z ní soubory. Mezi vyexportovanými soubory mohou být obrázky, dokumenty, zvukové nahrávky například z VoIP hovoru, videa atd.



Obrázek 2.5: Ukázka aplikace NetWitness Investigator.

Kapitola 3

Netfox Detective

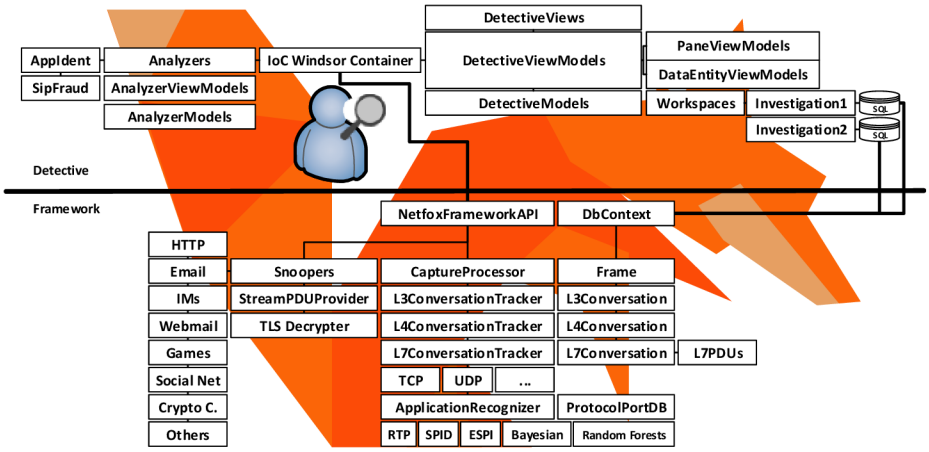
Nástroje pro forenzní účely v počítačových sítích mohou být rozděleny do dvou hlavních skupin. První skupinou nástrojů je NFAT (Network Forensic Analysis Tool). Tato skupina nástrojů dovoluje správci sítě monitorovat počítačovou síť, umožňuje shromažďovat informace o probíhajícím provozu na počítačové síti a pomáhá při vyšetřování nelegální činnosti na počítačových sítích. Druhá skupina NSM (Network Security and Monitoring) se spíše zaměřuje na monitorování a správu počítačové sítě. Mezi zástupce této skupiny se řadí například Wireshark, TCPDump či Microsoft Network Monitor.

Netfox Detective lze zařadit do skupiny NFAT. Je to tedy nástroj určený pro extrahování obsahu z aplikačních protokolů za účelem forenzní analýzy síťového provozu. Nástroj Netfox Detective je navrhnut pro pokročilou rekonstrukci a analýzu zachycených dat síťového provozu se zaměřením na emaily (včetně webových emailů), rekonstrukci HTTP (Hyper Text Transport Protocol), detekci a rekonstrukci hovorů přes VoIP (Voice over IP). Framework používá pokročilých technik a heuristik pro seskupení odposlechnutých dat, identifikaci provozu na aplikační vrstvě, rekonstrukce původních konverzací a zobrazení získaných dat z aplikační vrstvy pro vyšetřovatele.

Netfox Detective je v současné chvíli vyvíjen na platformě .NET Framework 4.7, konkrétně v programovacím jazyce C#. Jedná se o technologii vyvíjenou společností Microsoft. Netfox Detective je spustitelný na počítačích s operačními systémy Windows 7 a novějšími verzemi tohoto operačního systému [14, 15]. Mezi důležité návrhové vzory při vývoji aplikace patří návrhový vzor MVVM a DI (Dependency Injection). DI je návrhový vzor, který umožňuje snížit závislosti mezi jednotlivými komponentami aplikace. Jednou z výhod snížení závislostí mezi komponentami aplikace je ta, že jednotlivé komponenty aplikace se lépe testují [11]. Návrhový vzor MVVM bude podrobněji popsán v kapitole věnované implementaci webového rozhraní, protože je nesmírně důležitý pro pochopení, jak celá aplikace funguje a bude se podle něj implementovat i webové rozhraní.

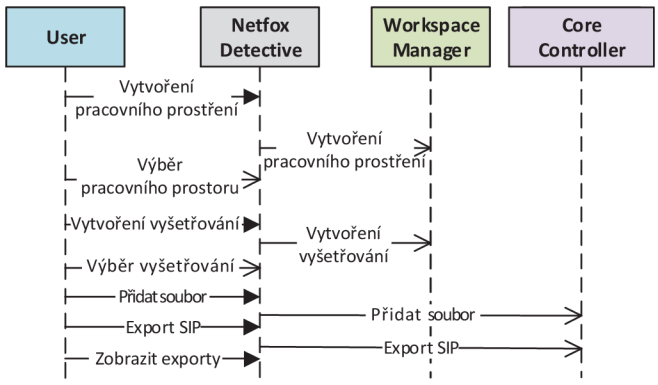
Nyní si popíšeme, jak aplikace a její důležité části fungují. Když se aplikace spouští, tak se musí vytvořit nový pracovní prostor či načíst již vytvořený. Pracovní prostor reprezentuje adresářovou strukturu. V pracovním prostoru se ukládají jednotlivá vyšetřování. Pracovní prostor může mít jedno či více vyšetřování. V rámci vyšetřování se ukládají soubory s odposlechnutou síťovou komunikací. Tato skutečnost je patrná i z obrázku 3.2. Nástroj můžeme rozdělit na dvě základní části. Těmito částmi jsou Framework a Detective. Rozdělení je naznačeno na obrázku 3.1. Ve spodní části obrázku se nachází množina komponent představující část Netfox Framework, která je zodpovědná za zpracování vstupních souborů se zachycenou komunikací. Ve vrchní části obrázku jsou pak komponenty patřící do grafického uživatelského rozhraní aplikace Netfox Detective. Grafické rozhraní pak zajišťuje

interakci s uživatelem a zobrazení výsledků zpracovaných částí Netfox Framework [14]. Řízení zpracování dat má na starost komponenta NetfoxFrameworkAPI. Řídící komponenta komunikuje s komponentami CaptureProcessor a Snoopers. Architektura propojení řídící komponenty NetfoxFrameworkAPI je naznačená na obrázku 3.1. CaptureProcessor rozdělí



Obrázek 3.1: Architektura nástroje Netfox Detective. Převzato z [14].

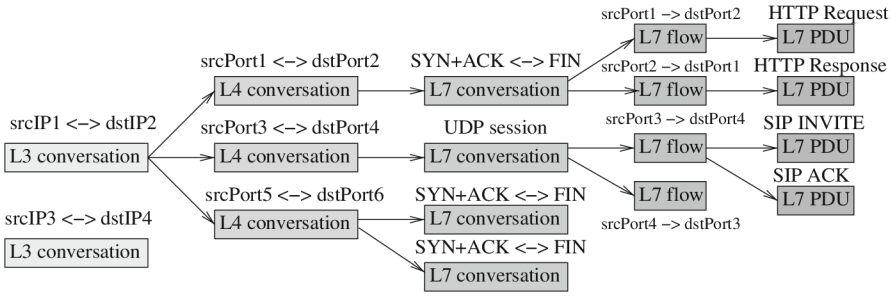
jednotlivé rámce v komunikaci podle jejich příslušnosti k příslušné konverzaci. Konverzace lze chápat jako pár kolekcí shromážděných rámců v odchozím i v příchozím směru síťového provozu seřazených podle časového razítka. Zpracování rámců na konverzace začíná zpracováním L3 konverzací. Určení, do které L3 konverzace rámec patří, probíhá na základě IP adres odesilatele a příjemce. Následně se zpracují L4 konverzace. Ty jsou definovány pomocí použitých čísel portů a použitého transportního protokolu. Posledním typem konverzací, které se vytváří, jsou L7 konverzace. V případě, že se v zachycených datech vyskytnou dvě



Obrázek 3.2: Schéma založení pracovního prostoru a přidání vstupního souboru. Převzato z [15].

komunikace, které využívají transportní protokol UDP a komunikují na stejných zdrojových a cílových portech, tak není možné tyto komunikace rozlišit. Příkladem této skutečnosti mohou být dvě aplikace využívající protokol SIP. Obě aplikace budou komunikovat na stejném

zdrojovém i cílovém portu, a to například na portu 5060 pro všechny SIP konverzace. Proto může být L4 UDP konverzace považována za L7 konverzaci. Nyní popsané zjištění můžeme vidět i na obrázku 3.3, kde je naznačeno rozdělení rámců zachyceného provozu do jednotlivých typů konverzací.



Obrázek 3.3: Zpracovaní rámců z komunikace na konverzace. Převzato z [7].

3.1 Pmlib

Jedná se o knihovnu napsanou v jazyce C#. Knihovna poskytuje rozhraní pro manipulaci se soubory se zachycenou komunikací. Knihovna podporuje různé typy formátů souborů, do kterých lze ukládat zachycenou síťovou komunikaci. Jsou to například Wireshark/ TCPDump LibPcap, Microsoft Network Monitor či Pcap-ng formát. Dále knihovna PmLib zpracovává vstupní soubory, aby poskytla možnost přistoupit k jednotlivým síťovým vrstvám (L2, L3 a L4) rámců uložených v souboru se zachycenou komunikací. Pro zpracování rámců využívá knihovnu PacketDotNet 0.13. Tato knihovna podporuje protokoly Ethernet, LinuxSSL, PPP, PPPoE, IP, TCP a další.[13]

3.2 ApplicationRecognizer

Za bližší zmínění stojí rozhodně jedna z nejdůležitějších komponent nástroje Netfox Detective nazývající se ApplicationRecognizer. Komponenta má na starost identifikaci aplikačního protokolu, který se v komunikaci vyskytuje. Existuje několik způsobů, jak aplikační protokol určit. Nejjednodušší metodou je rozhodovat na základě čísel portů, která jsou přiřazena organizací IANA (Internet Assigned Numbers Authority). Bohužel tato metoda selhává na konverzacích, kde aplikace používají dynamicky přidělené porty, peer-to-peer komunikaci, video streaming atd.

Pokročilé metody zkoumají obsah nesených dat a pro určení aplikačního protokolu se snaží nalézt nějaký charakteristický vzor, který se může vyskytovat ať už v hlavičce nebo v nesených datech. Netfox Detective používá dvě pokročilé metody:

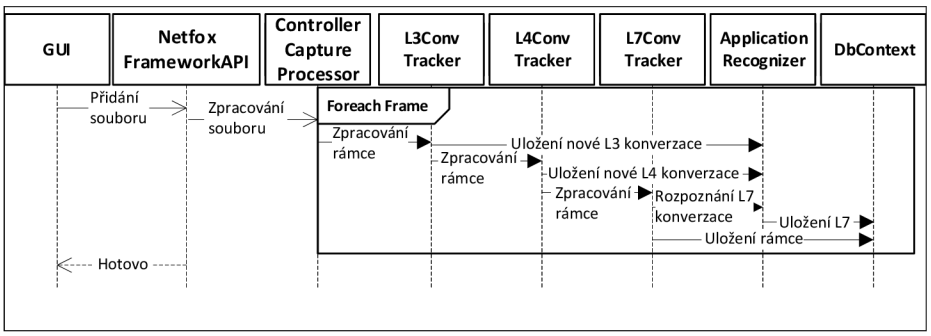
- **RTP fingerprinting**
  Používá se, pokud nelze určit protokol na základě portu. Metoda používá vícestupňový klasifikátor, který sleduje minimální délku RTP hlavičky, číslo verze RTP a typ dat nesených v RTP.[7]

- **SPID (Statistical Protocol Identification)**
  S touto metodou jako první přišel Erik Hjemvik. Zakládá se na metodě učení s učitelem, jedná se o jednu z metod strojového učení. Tato metoda strojového učení potřebuje trénovací sadu, v níž bude ke konkrétním vzorkům provozu přiřazena správná klasifikace. Poté si algoritmus vytvoří databázi modelů protokolů, do které si uloží otisky aplikačních protokolů.[7]

3.3 Zpracování vstupního souboru

Nyní bude vysvětleno, jak nástroj Netfox Detective funguje jako celek při získávání forenzních důkazů pro orgány činné v trestním řízení. Prvně uživatel pomocí grafického uživatelského rozhraní vloží soubor se zachycenou komunikací. Formát souboru by měl být takový, který podporuje knihovna PmLib, jak bylo uvedeno výše. Po vložení souboru se zavolá knihovna PmLib, jenž z rámců uložených ve vstupním souboru vytvoří kolekci.

Následně se pomocí ConversationTrackeru asynchronně zpracují všechny typy konverzací pro daný rámec z kolekce. Výsledky jednotlivých typů konverzací se uloží do SQL databáze pomocí komponenty DbContext. Typy konverzací mohou být například L3, L4 a L7, kde čísla odpovídají číslům vrstev ISO/OSI síťového modelu. L7ConversationTraker vytváří konverzace, které probíhají na aplikační vrstvě nad transportními protokoly TCP nebo UDP a vytváří objekty L7PDU, které reprezentují zprávy aplikačního protokolu. Objekty L7PDU jsou tvořeny bez jakékoliv znalosti daného aplikačního protokolu. Při vytváření konverzací a rekonstrukci se využívají čísla portů a segmentová čísla v protokolu TCP pro odhalení chybějících dat či neukončeného TCP spojení. Proces zpracování vloženého a vstupního souboru se zachycenými daty můžeme sledovat na obrázku 3.4.



Obrázek 3.4: Zpracování vstupního souboru na konverzace. Převzato z [15].

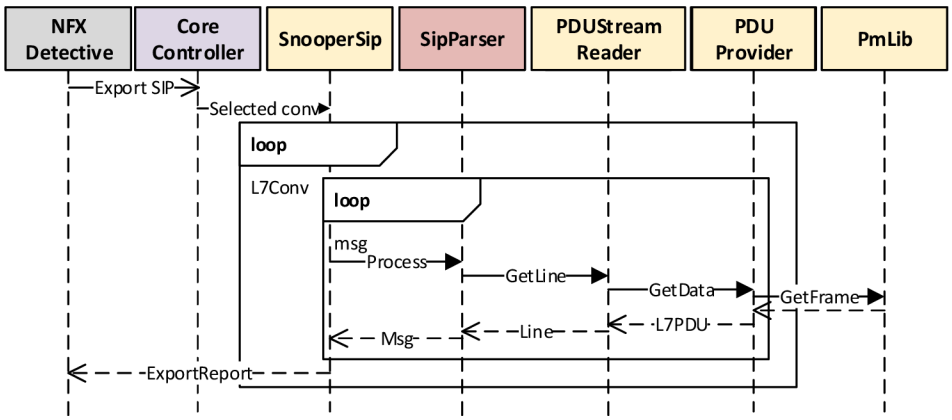
Krok, který nyní následuje, je pro úspěšnou forenzní analýzu stěžejní. V této fázi zpracování se musí určit, jakým aplikačním protokolem se v L7 konverzaci komunikuje. O identifikaci aplikačního protokolu, jak už bylo výše řečeno, se stará ApplicationRecognizer. Identifikace aplikačního protokolu je zásadní z toho důvodu, že na základě identifikace se vybere modul, který z příslušných zpráv aplikačního protokolu reprezentovaných L7PDU extrahuje informace důležité pro forenzní analýzu. Když bychom tedy určili aplikační protokol špatně, tak bychom v lepším případě nedostali žádné informace. V tom horším případě bychom mohli extrahovat informace, které neodpovídají realitě.

3.4 Snooper

Modul zajišťující extrahování informací pro forenzní analýzu se nazývá Snooper. Samozřejmě Netfox Detective obsahuje Snooper modulů více, protože každý Snooper zpracovává určitý aplikační protokol. Snooper moduly si Netfox Detective načítá dynamicky, z čehož plyne, že když budeme chtít přidat nový Snooper, tak nemusíme kompilovat celou aplikaci znova, ale postačuje zkompilovat Snooper a jeho binární soubory nakopírovat do složky s binárními soubory Netfox Detective. Výstup jednoho modulu Snooper může být napojený na vstup druhého modulu za účelem další rekonstrukce provozu.

Uveďme si příklad, kdy se výstup zpracování protokolu HTTP předá na vstup modulu, který provádí rekonstrukci web mailu. Snooper po dokončení analýzy exportuje data obsažená v konverzacích spolu s metadaty získanými při analýze aplikačního protokolu v daném vyšetřování.

Každý Snooper obsahuje vlastní modely, view-modely a pohledy pro detailní zobrazení dat získaných během rekonstrukce. Například HTTP snooper zobrazuje rekonstruovaný web, email snooper vypíše rekonstruované emaily, VOIP snooper sestavování hovoru s RTP streamy pro možné přehrání atd. Na obrázku 3.5 si můžeme všimnout příkladu znázorňujícího extrahování dat z protokolu SIP. Prvně přijde zpráva řídící komponentě od uživatelského rozhraní, aby extrahovala data pro forenzní účely z protokolu SIP. Řídící komponenta spustí snooper učený pro SIP protokol a předá mu konverzace jež byly identifikovány, že v nich probíhá komunikace tímto protokolem. Snooper pak projde všechny přiřazené L7 konverzace, ze kterých si extrahuje SIP zprávy. Po extrahování SIP zpráv se vytvoří ExportReport, kde se uvedou kupříkladu čísla volajícího a volaného, délka ukončeného hovoru, stav hovoru, zda se navazuje spojení hovoru, hovor probíhá či hovor byl již ukončen atd.[15]



Obrázek 3.5: Příklad extrahovaní forenzních informací z protokolu SIP. Převzato z [15].

Kapitola 4

Návrh implementace webového rozhraní

Návrh implementace webového rozhraní se bude dělit do několika částí. V první části budou konkretizovány požadavky, které budou pro webové rozhraní požadovány. V rámci požadavků bude vymezeno, co by mělo webové rozhraní umožňovat. Dále bude potřeba popsat, jak má vyšetřovatel postupovat pro exportování dat ze zachycené komunikace. Do poslední části bude zahrnut návrh databáze, do které se budou ukládat informace o uživatelích, vyšetřování, statistikách souborů se zachycenou komunikací a statistikách o exportech v daném vyšetřování.

4.1 Požadavky na webové rozhraní

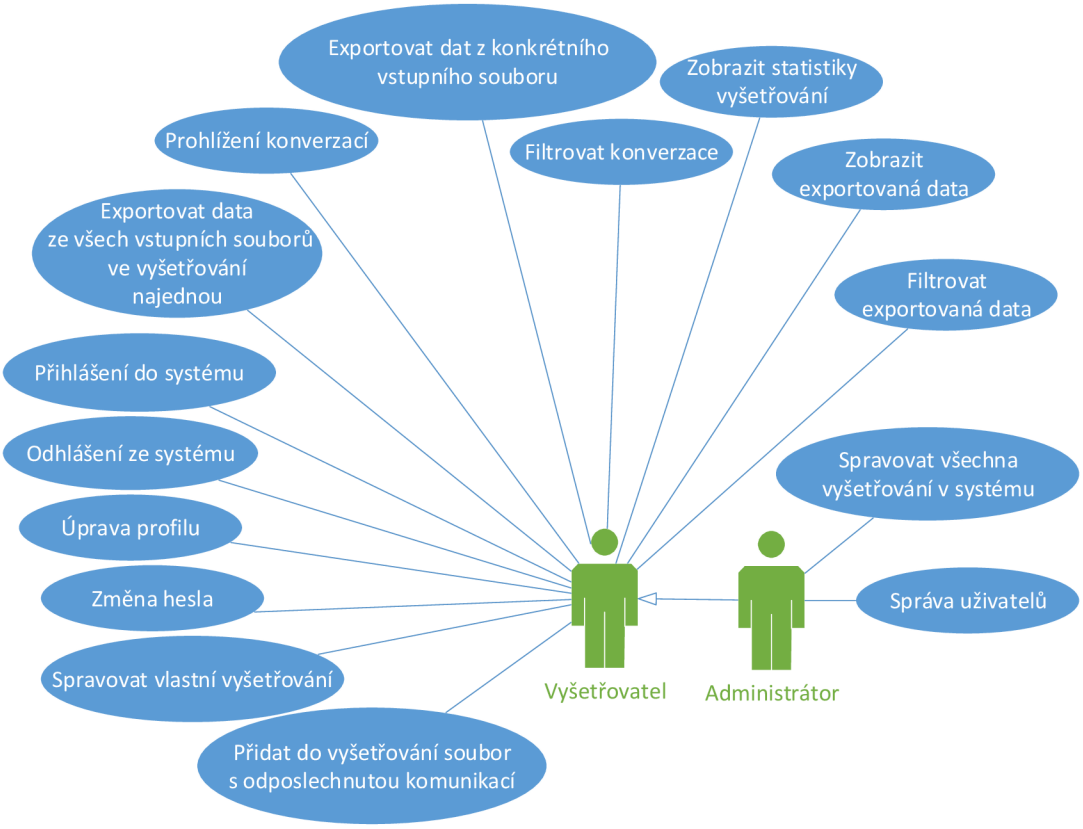
Tato podkapitola se věnuje popisu požadavků na části webového rozhraní. Specifikace požadavků obsahuje definici rolí, které by se ve webovém rozhraní měly vyskytovat. Poté bude specifikováno, co by měla nabízet a splňovat správa uživatelů a správa vyšetřování. Nakonec se specifikuje, jak by mělo probíhat zpracování vloženého vstupního souboru se zachycenou komunikací a následné zobrazení výsledků ze zpracování.

Uživatelské role

Z diagramu případů užití viz 4.1 je patrné, že webové rozhraní bude obsahovat dvě uživatelské role. Roli administrátor a roli vyšetřovatel. Role administrátora se od role vyšetřovatele liší ve dvou případech, kterými jsou správa uživatelů, jenž je popsána níže a správa do všech vyšetřování v systému. Z toho vyplývá, že roli administrátora by měl mít uživatel, který bude zajišťovat správu celého systému.

Požadavky na správu uživatelů

Ve správě uživatelů bychom kromě vytváření a mazání uživatele měli být schopni nastavit, zda daný uživatelský účet je aktivní či nikoli. To znamená, že pokud uživatelský účet nebude aktivní, tak se uživatel s tímto účtem nebude moci přihlásit. Vytváření a mazání uživatelských účtů bude moci provádět pouze administrátor. Dále bude dostupná stránka s informacemi o profilu uživatele, na níž bude mít přístup jen konkrétní uživatel. Zde nalezne informace o svém účtu, například jméno vlastníka uživatelského účtu, přiřazenou roli atd. Dále si uživatel bude moci na této stránce změnit heslo.



Obrázek 4.1: Diagram případů užití.

Požadavky na správu vyšetřování

Správa vyšetřování by měla umožnit vytváření a mazání vyšetřování. Práva uživatelů na vytváření a mazání vyšetřování jsou popsána výše. Vlastník vyšetřování může:

- editovat informace o vyšetřování,
- přidávat vyšetřovatele do vyšetřování,
- odebírat vyšetřovatele z vyšetřování.

Vlastník vyšetřování může být kdykoliv změněn. Tuto změnu může provést buď administrátor nebo může být provedena samotným vlastníkem vyšetřování.

Uživatel s rolí vyšetřovatel by měl mít přístup pouze do vyšetřování, která vlastní nebo do nichž dostal přístup od vlastníka nějakého vyšetřování. Vlastníkem vyšetřování je vždy uživatel, který vyšetřování založí. Dále by vyšetřovatel neměl mít možnost upravovat informace o vyšetřování (název, popis vyšetřování) a zároveň by neměl mít možnost smazat vyšetřování, pokud není jeho vlastníkem. Uživatel s rolí administrátora by pak měl mít plný přístup do všech vyšetřování a měl by mít možnost měnit informace o vyšetřování a smazat vyšetřování bez nutnosti mít oprávnění přistupovat do vyšetřování nebo bez nutnosti být vlastníkem vyšetřování.

Požadavky na zpracování vstupních souborů

Uživatel s přístupem do daného vyšetřování by měl být schopen vložit soubory se zachycenou komunikací. Tím, že daný soubor vloží, se spustí zpracování souboru na konverzace. Dále by mělo být umožněno uživateli vybrat, jaké z dostupných protokolů chce exportovat. Pokud by u konkrétního vstupního souboru byl některý z vybraných protokolů v minulosti již exportován, tak se tento protokol na konkrétním vstupním souboru nesmí exportovat. Aby bylo možné znovu exportovat již zpracovaný protokol na určitém vstupním souboru, musí se odebrat ze seznamu exportovaných protokolů daného vstupního souboru. Exportovat data by se mělo dát jak z pouze jednoho konkrétního vstupního souboru, tak i ze všech najednou.

Požadavky zobrazení výsledků

Co se týče požadavků na zobrazení výsledků. U vstupního souboru by se měly zobrazit informace například velikost, počty jednotlivých typů konverzací, použité transportní protokoly atd. Dále taky seznam L3, L4, L7 konverzací vyskytujících se ve vstupním souboru. Konverzace by mělo být možné filtrovat a to podle:

- IP adresy a portu,
- časového období výskytu konverzace,
- počtu rámců v dané konverzaci,
- velikosti přenesených dat v konverzaci.

Dále by se mělo u konverzací zobrazit jaké konverzace a rámce do dané konverzace patří. Například pro L4 konverzaci by se mělo zobrazit do jaké L3 konverzace patří a jaké L7 konverzace a rámce patří do dané L4 konverzace.

U zobrazení výsledků z daných protokolů bude muset uživateli umožnit filtrovat exportovaná data, a to alespoň podle IP adresy a portu a časového rozpětí výskytu exportovaných dat. Další možnosti filtrace dat záleží na jednotlivých protokolech.

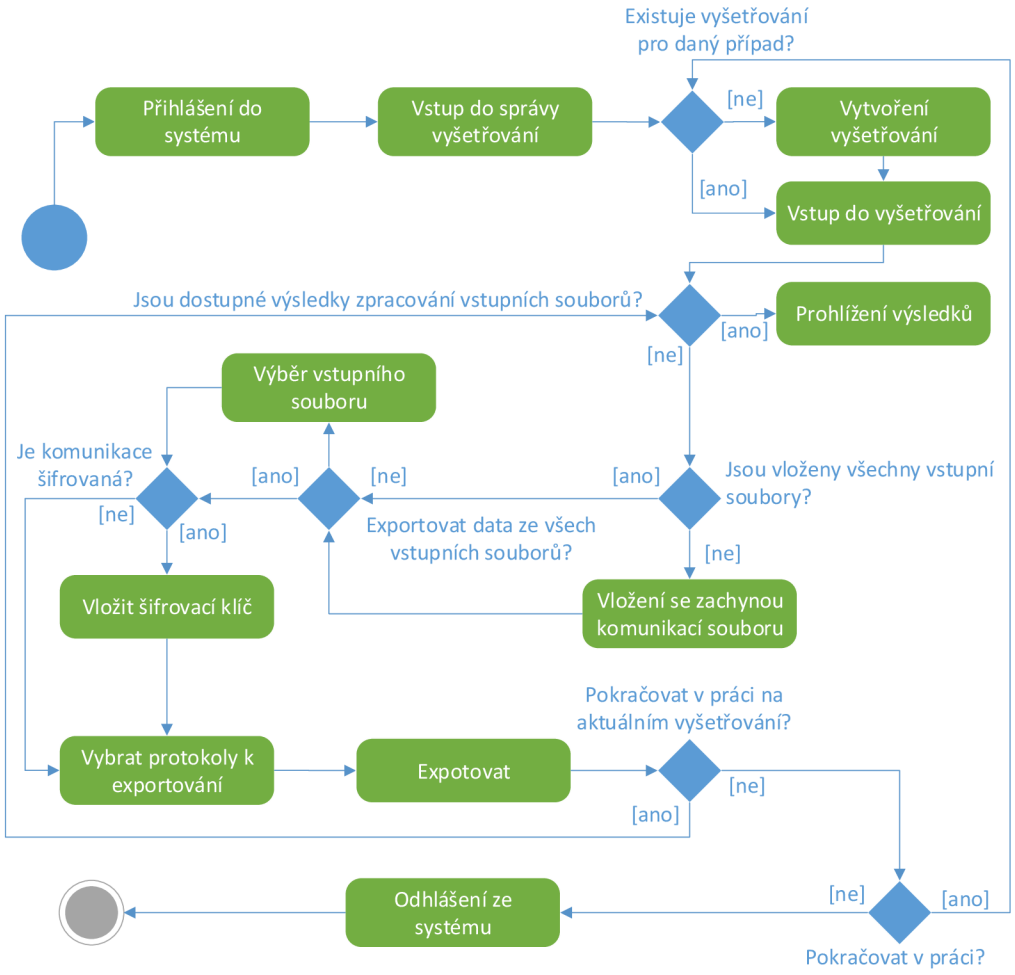
Požadavky na použití technologií

Mezi technickými požadavky je použití programovacího jazyka C#. Tento požadavek vychází z toho, že Netfox Framework je napsaný právě v tomto programovacím jazyce. Dalším požadavkem je použití návrhového vzoru Dependency Injection. Použití tohoto návrhového vzoru se poté využije pro lepší testování jednotlivých částí webového rozhraní. Návrhový vzor, který bude nutné použít je MVVM, který nyní bude popsán podrobněji.

4.2 Postup vyšetřování

Celý postup vyšetřování, který bude v podkapitole popisován, je znázorněn v diagramu aktivit 4.2. Prvním krokem bude pro vyšetřovatele přihlášení do systému přihlašovacími údaji, jenž dostane od správce systému. Po přihlášení do systému bude muset vyšetřovatel vybrat vyšetřování, v němž chce provádět analýzu provozu. Za tímto účelem vstoupí do správy vyšetřování, kde se mu vypíše seznam dostupných vyšetřování. Pokud neexistuje pro daný případ vyšetřování, tak se zde bude muset vytvořit.

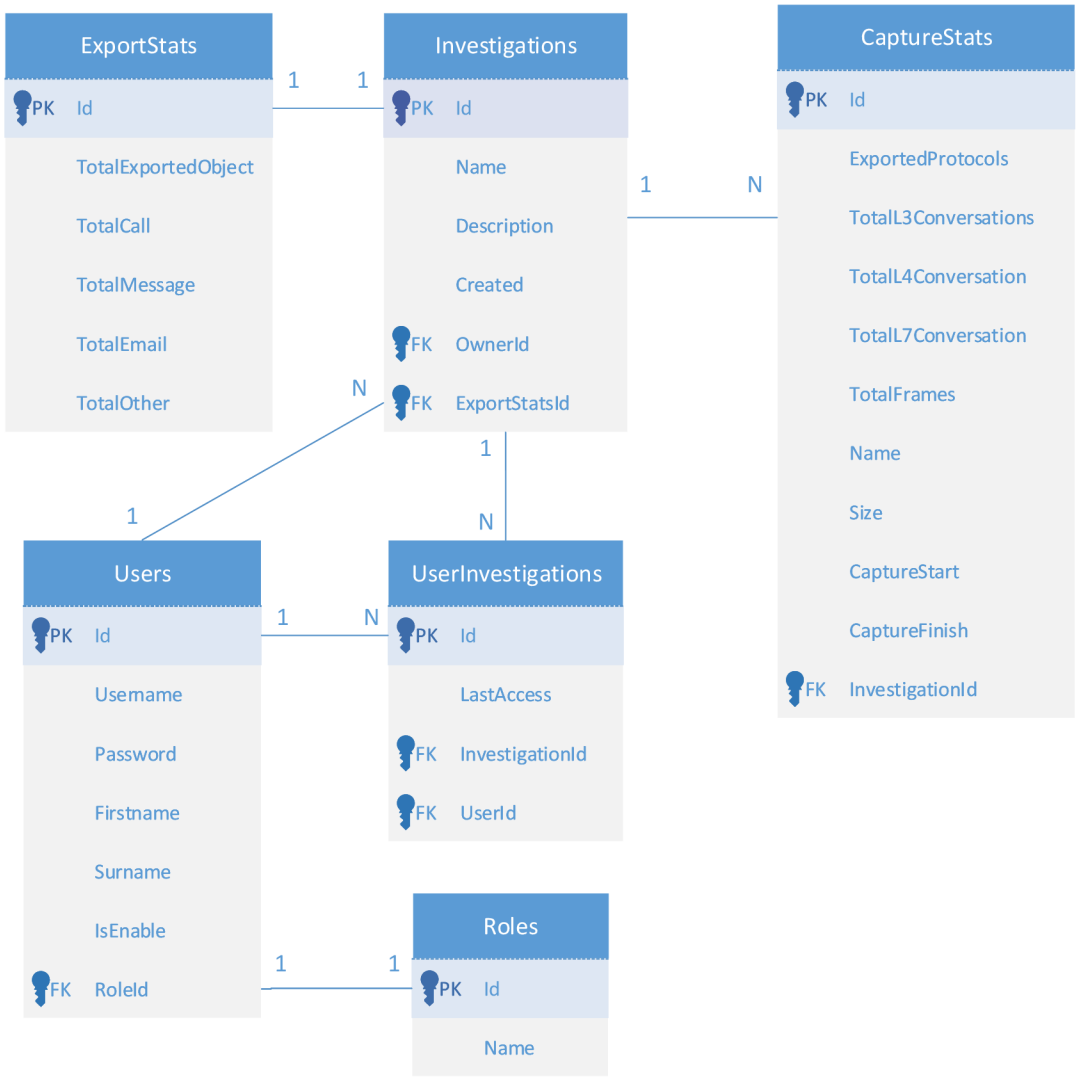
Po vybrání a vstupu do vyšetřování vyšetřovatel může prohlížet výsledky předešlého zpracování zachycené komunikace. Pokud se jedná o nově vytvořené vyšetřovaní, bude muset vyšetřovatel vložit do vyšetřování vstupní soubory se zachycenou komunikací. Jakmile všechny vstupní soubory vyšetřovatel přidá do vyšetřování, pak má na výběr ze dvou možností, jak extrahovat data ze zachycené komunikace. Buď se dají extrahovat data ze všech vstupních souborů najednou nebo lze zvolit konkrétní vstupní soubor a extrahovat data pouze z něj. V dalším kroku musí vyšetřovatel ke komunikaci, která je šifrovaná, připojit klíč pro její dešifrování, jinak nebude možné získat všechna přenášená data. Teď již vyšetřovateli zbývá jen vybrat z jakých protokolů chce extrahovat data a následně spustit exportování. Po dokončení pak může vyšetřovatel přistoupit k výsledkům zpracované zachycené komunikace. Až bude vyšetřovatel chtít ukončit práci, musí se odhlásit.



Obrázek 4.2: Diagram aktivit při vyšetřování.

4.3 Návrh databáze

Návrh databáze se týká pouze správy uživatelů a vyšetřování. Návrh zmíněné části systému je vyobrazen ER diagramem 4.3. Začneme popisem návrhu správy uživatelů. Pro ni jsou důležité dvě tabulky *Users* a *Roles*. Tabulka *Roles* uchovává dostupné uživatelské role v systému. V tuto chvíli by měla obsahovat dva záznamy pro roli administrátora a vyšetřovatele. Uživatel může mít pouze jednu roli, proto je tedy kardinalita vztahu mezi tabulkami *Users* a *Role* 1:1. Do tabulky *Users* by se pak měly zaznamenávat základní informace o uživateli, jako jeho jméno, příjmení, uživatelské jméno, heslo a role. *IsEnable* určuje, zda se může daný uživatel přihlásit nebo je jeho účet blokován.



Obrázek 4.3: ER diagram části databáze pro správu uživatelů a vyšetřování.

Informace o samotném vyšetřování se budou podle návrhu uchovávat v tabulce *Investigations*. Z ní se bude dát získat název vyšetřování, popis a kdy bylo vyšetřování založeno. Tabulka *Investigations* má relace se čtyřmi tabulkami:

- Díky relaci s tabulkou *ExportStats* lze získat statistiky exportovaných objektů. Do návrhu se počítalo s celkovým počtem exportovaných objektů, počtem hovorů, počtem zpráv, počtem extrahovaných emailů a s počtem ostatních vyexportovaných objektů.
- Relací s tabulkou *CaptureStats* jsme schopni dopočítat souhrnné statistiky z jednotlivých vstupních souborů.
- Přímá relace s tabulkou *Users* určuje vlastníka daného vyšetřování.
- Druhá relace s tabulkou *Users*, v tomto případě přes vazební tabulku *UserInvestigations* umožňuje přiřadit přístup vyšetřovatelům, jenž se mají podílet na forenzní analýze v konkrétním vyšetřování.

CaptureStats kromě údajů pro statistiky, jak již bylo řečeno výše, se v tabulce nachází navíc sloupec *ExportedProtocols*. Můžeme, z něho zjistit z jakých protokolů již byla data extrahována. Poslední, co je ještě potřeba zmínit je sloupec *LastAccess* ve vazební tabulce *UserInvestigations*. Bude sloužit pro zaznamenání času posledního přístupu uživatele do konkrétního vyšetřování.

Kapitola 5

Implementace

Kapitola o implementaci se bude nejprve věnovat popisu klíčových frameworků a knihoven, které velmi zjednodušily samotnou implementaci webového rozhraní. Následně se budeme v kapitole zabývat, jakým způsobem bylo implementováno přihlašování uživatelů a propojení webového rozhraní a Netfox Framework. Poslední pasáž kapitoly se zabývá popisem vytvořeného uživatelského rozhraní.

5.1 Použité frameworky a knihovny

Mezi použité frameworky patří hlavně DotVVM, u kterého bude zmíněno k čemu slouží a způsob generování požadovaných stránek. Dalšími použitými frameworky jsou Bootstrap usnadňující vývoj webových stránek předdefinovanými styly a Chart.js pro zobrazení grafů. Poslední zmíněnou knihovnou je Hangfire. Ta slouží pro spouštění úloh na pozadí webového serveru.

5.1.1 Dotvvm

Pro vývoj aplikací pod platformou .NET může programátor využít oficiálních řešení, jako jsou ASP.NET MVC nebo ASP.NET Web API v kombinaci s nějakým javascriptovým frameworkem jako React, Angular nebo Knockout. Toto řešení není ale úplně pohodlné při vývoji větších aplikací, protože u větších aplikací většinou budeme potřebovat další prvky na stránce, například nějaké posuvníky či komponentu pro zadávání kalendářního data. V tom případě sáhneme po další javascriptové knihovně řešící tento problém. Při vkládání více knihoven může dojít k velmi nepříjemné věci, a to k tomu, že javascriptové knihovny mohou být v konfliktu. To znamená, že se budou navzájem ovlivňovat a bude docházet k neočekávanému chování jednotlivých komponent uživatelského rozhraní. Za tím účelem je dobré použít nějaký framework poskytující všechny potřebné komponenty.

V rámci tvorby webového rozhraní pro Netfox Detective byl zvolen framework DotVVM, kde Dot znamená .NET Framework a VVM znamená MVVM návrhový vzor. DotVVM používá návrhový vzor MVVM popsaný níže v této kapitole. Výhodou tohoto frameworku je, že nevyžaduje po programátorovi znalost jakékoliv javascriptové knihovny, aby mohl poskytované komponenty používat. Postačuje znalost C#, HTML a CSS. Práce s frameworkem je velmi jednoduchá. Pohled stránky se definuje za pomoci syntaxe značkovacího jazyka HTML a přidaných značek pro DoVVM, proto se tento jazyk nazývá DOTHTML. Pro použití komponent z DotVVM se dále používá následující značka `dot:název komponenty`. Data z viewmodelu se pak následně získávají pomocí data bidingu. Důležité je také přidat

na začátek každého pohledu anotaci říkající která třída je viewmodelem pro daný pohled. Viewmodel je pak standardní C# třídou. Ve třídě jsou pak pouze definovány vlastnosti a metody, například pro obsluhu události kliknutí na nějaké tlačítko.[9]

Obrázek 5.1: Schéma zpracování požadavku v DotVVM. Převzato z [18].

U Dotvvm stojí rozhodně za zmínku proces, jak se zpracovává úvodní požadavek při prvním načtení stránky (HTTP GET požadavku) a jak probíhá zpracování provedení příkazu na stránce (AJAX HTTP POST), například při kliknutí na tlačítko, čímž zavoláme metodu ve viewmodelu. V levé části následujícího obrázku 5.1 jsou znázorněny jednotlivé fáze procesu zpracování odpovědi na úvodní požadavek (HTTP GET) a v pravé části obrázku pak fáze procesu zpracování příkazu na stránce (AJAX HTTP POST). Po přijetí požadavku na první načtení stránky se nejprve vytvoří třída viewmodelu požadované stránky. Poté se provedou metody Init(), Load() a PreRender(). Po provedení těchto metod se provede renderování HTML stránky. Před odesláním vyrenderované HTML stránky se musí serializovat ještě viewmodel patřící ke stránce. Následně se HTML a serializovaný viewmodel pošle klientovi.

Když se zaměříme na zpracování provedení akce na stránce, na obrázku 5.1 můžeme vidět pár rozdílů. V prvním řadě, po vytvoření viewmodelu a jeho inicializaci metodou

Init(), proběhne deserializace hodnot viemodelu zaslaných od klienta a přiřazení hodnotám právě vytvořeného viewmodelu. Po provedení metody Load() se vykoná požadovaný příkaz, například zavolání metody ve viewmodelu. Po provedení příkazu se zavolá metoda PreRender(), ale po jejím vykonání se již neprovádí renderování HTML, ale aktuální stav viewmodelu se serializuje a odešle se klientovi.[18]

5.1.2 Bootstrap, HangFire a Chart.js

Bootstrap je open source framework pro vývoj webových stránek a aplikací využívající HTML, CSS a Javascript. Framework obsahuje předpřipravené komponenty. Jedná se například o navigaci, formulářové prvky, ikony, stránkování, drobečková navigace atd. Veškeré komponenty ve frameworku se chovají responzivně, což zajišťuje, že výsledná webová stránka se korektně zobrazí ve všech možných velikostech rozlišení displeje.[1]

Další knihovnou, která bude využita ve webovém rozhraní je HangFire. Jde o knihovnu umožňující spouštět výpočetní úlohy na pozadí webového serveru [16, 12]. Tuto funkcionalitu využijeme pro zpracování vstupních souborů se zachycenou komunikací. Můžeme si představit kupříkladu, že přejdeme na stránku s extrahovanými daty a po příchodu na tuto stránku by se spouštěla analýza provozu zachycené komunikace. Analýza komunikace pak trvá značnou dobu v řádu sekund až hodin. A to není žádoucí, protože stránka by se měla načíst v řádu stovek milisekund.

Poslední důležitou knihovnou, která se využívá v implementaci, je open source javascriptová knihovna Chart.js umožňující jednoduše vykreslit spoustu grafů[1]. K tomu, aby knihovna fungovala správně potřebuje, aby webový prohlížeč podporoval HTML 5, což v dnešní době podporují všechny prohlížeče. Knihovna podporuje responzivní zobrazení a je testována pro většinu prohlížečů používaných v současnosti.

5.2 Docker

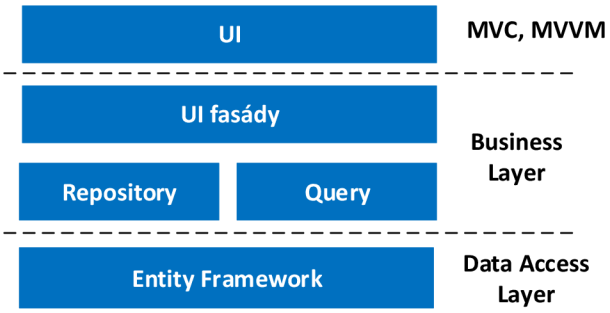
Jedná se o aplikační platformu. Umožňuje spouštět aplikace navzájem odděleně v tzv. kontejnerech. Použitím kontejneru jako prostředku odlehčené virtualizace ve výsledku znamená velmi efektivní způsob oproti běžné virtualizaci. Kontejner se dokáže spustit do několika sekund, a co je důležité, nevznikají tu režijní nároky na paměť či výpočetní výkon. Pro nahrání a spuštění aplikace pomocí Dockeru slouží *Docker image*. *Docker image* neobsahuje jen samotnou aplikaci, ale i všechny závislosti, které jsou potřeba pro běh aplikace. To, jak sestavit *Docker image* se definuje textovým souborem nazvaným *Dockerfile*. V našem případě *Dockerfile* byl vygenerován Visual Studio, které vytvořilo s *Dockerfile* i samostatný projekt Netfox.Web.Docker, jenž vygenerovaný *Dockerfile* obsahuje.[21]

5.3 Architektura webového rozhraní

Před popisem implementací jednotlivých komponent webového rozhraní, je nezbytné zmínit, jak vypadá architektura celého řešení a strukturu dílčích projektů. Jde-li o architekturu řešení, pak je zde využita architektura třívrstvého modelu vyobrazená v obrázku 5.2. Architektura se dělí na tři vrstvy, jenž mezi sebou komunikují.

Pokud začneme horní vrstvou z obrázku 5.2, tak ta obstarává pouze zobrazování dat a komunikaci s uživatelem. Jedná se tedy o samotné uživatelské rozhraní. Prostřední vrstva

[1] http://www.chartjs.org/samples/latest/



Obrázek 5.2: Použitá architektura při implementaci. Převzato z [4].

z obrázku 5.2 tzv. Business Layer zastřešuje veškerou logiku aplikace. Obsahuje UI fasády, které zajišťují komunikaci s uživatelským rozhraním. Dále v této vrstvě nalezneme repozitáře a Query objekty, které slouží na formulaci databázového dotazu pro získání či úpravu dat v databázi. Tím se dostáváme k poslední vrstvě tzv. Data Access Layer, jenž zajišťuje komunikaci se samotnou databází. Pro usnadnění implementace této architektury se využívá knihovny *Riganti.Utils.Infrastructure*[2].

Řešení projektu se pak dělí do dílčích projektů právě podle těchto tří vrstev architektury. Jedná se o tyto tři projekty:

- Netfox.Web.App,
- Netfox.Web.BL,
- Netfox.Web.DAL.

Netfox.Web.App koresponduje s vrstvou architektury, jenž zastřešuje samotné uživatelské rozhraní. Adresářová struktura projektu a význam jednotlivých složek je následující:

- View - složka obsahuje pohledy pro jednotlivé stránky,
- ViewModel - obsahuje třídy viewmodelů pro jednotlivé pohledy,
- Controls - vlastní nakódované komponenty,
- Installers - nachází se zde třídy pro načtení potřebných závislostí do kontejneru pro dependency injection,
- Helpers - pomocné třídy pro viewmodely,
- Template - obsahuje soubory týkající se stylů stránky a skripty v jazyku Javascript.

Projekt Netfox.BL reprezentuje vrstvu Bussiness Layer. Struktura tohoto projektu je tato:

- Repositories - složka pro třídy repozitářů,
- Provider - třídy poskytující objekty pro přístup k datům,

[2] https://github.com/riganti/infrastructure

- Facades - obsahuje UI fasády,
- Queries - složka pro třídy sloužící pro sestavení databázového dotazu,
- DTO - obsahují třídy zastupující roli DTO (Data Transfer Object),
- Mappings - sem se umisťují třídy definující, jak se jednotlivé Entity namapují na DTO a naopak, jak se DTO převedou na Entity.

Netfox.Web.DAL zastupuje zbývající vrstvu Data Aceess Layer. V projektu nalezneme pouze třídy definující entity z návrhu databáze z předešlé kapitoly.

U architektury bych ještě zmínil důležité návrhové vzory, kterých se v implementaci využívá. Jedním z nich je návrhový vzor *Dependency Injection*. V implementaci se využívá ze dvou důvodů, a to, protože Netfox.Framework k předávání závislostí tento návrhový vzor používá. Druhým důvodem je modularita webového rozhraní. Modularitu potřebujeme hlavně pro případné rozšíření dalších protokolů a zároveň umožňuje lepší testování dílčích částí systému. Pro realizaci tohoto návrhového vzoru ve webovém rozhraní se využívá knihovny *Castle.Windsor*, Ta, jak je výše zmíněno, se používá i v Netfox Frameworku. Kontejner pro registrování komponent se inicializuje v konfiguračním souboru pro webové rozhraní, tedy ve *Statup.cs*. Po vytvoření instance kontejneru se zavolají instalátory (třídy implementující rozhraní `IWindsorInstaller`), které zaregistrují jednotlivé komponenty webového rozhraní. Aby se *Dependency Injection* aplikovalo v rámci DotVVM, bylo potřeba přidat do konfigurace DotVVM vlastní třídu implementující rozhraní `IViewModelLoader` z důvodu, aby se při načítání stránky použil pro získávání viewmodelu a všech jeho potřebných závislostí kontejner vytvořený na začátku konfigurace webového rozhraní. Podrobný postup, jak aplikovat *Dependency Injection*, se nachází v dokumentaci samotného DotVVM[3].

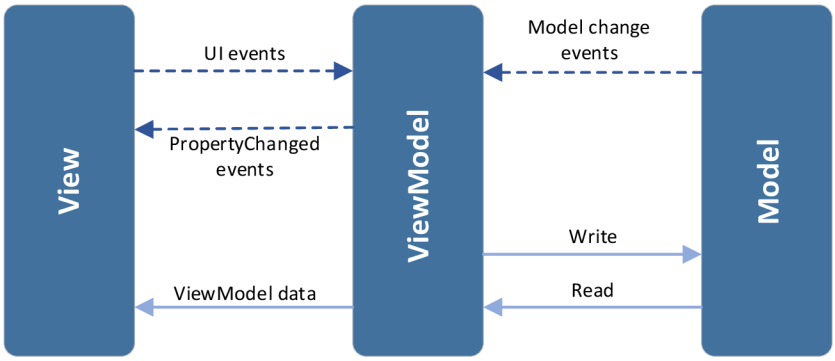
Návrhový vzor MVVM

Další z důležitých návrhových vzorů použitých při implementaci je MVVM. Jedná se o návrhový vzor, pomocí kterého lze při vývoji aplikace oddělit uživatelské rozhraní od logiky aplikace. Po rozdělení těchto dvou částí aplikace se zdrojový kód aplikace stává přehlednější a následné úpravy aplikace či vývoj a rozšíření aplikace jsou jednoduší. Další výhodou, kterou oddělením uživatelského rozhraní od logiky aplikace získáme, je zlepšení možnosti testovat jednotlivé části aplikace [2, 8]. Návrhový vzor MVVM se skládá z následujících částí [8]:

- *Model* je třída, která zajišťuje definici dat, se kterými logika aplikace pracuje. Například pokud by se jednalo o webovou aplikaci, tak model bude definovat, jak se data budou ukládat do databáze a jakým způsobem se budou z databáze získávat.
- *View* je třída definující uživatelské rozhraní. Bude tedy definovat, kde se dané informace zobrazí a jakým způsobem. Případně může informovat zasláním zprávy ViewModel o nějaké události uživatelského rozhraní, kupříkladu kliknutí na tlačítko.
- *ViewModel* je třídou, kterou můžeme chápat, jako spojovací článek mezi Modelem a View, jak může být patrné z obrázku 5.3. Každá třída View má k sobě korespondující třídu ViewModel. ViewModel přijímá data od Modelu a ty zpracuje tak, aby do View posléze tyto data předal v korektním formátu, kterému třída View rozumí. Pokud

[3] https://www.dotvvm.com/docs/tutorials/advanced-ioc-di-container/1-1

dojde ke změně dat Modelu, tak o tom ViewModel informuje pomocí notifikace View. A stejně tak i pokud dojde k nějaké události v uživatelském rozhraní, například kliknutí na tlačítko nebo vyplnění textového pole, zašle View do ViewModelu notifikaci o události a ten následně událost vyhodnotí a podle toho změní data v Modelu.



Obrázek 5.3: Vztahy mezi vrstvami View, Viewmodel, Model. Převzato z [8].

Pro přístup k datům a jejich úpravám se používá návrhový vzor *Unit Of Work*. To aplikaci umožňuje zaznamenávat si požadované změny databáze v určeném databázovém kontextu. Pro manipulaci s jednotlivými entitami se ve webovém rozhraní používá návrhový vzor `Repository`. Návrhový vzor umožňuje odstínit přístup k úložišti perzistentních dat.[10]

5.4 Autentizace uživatelů

V podkapitole bude popsáno, jak je implementováno přihlašování uživatelů. Tato část nebyla řešena na desktopové variantě, protože nebylo potřeba. Uživatelé měli totiž aplikaci nainstalovanou lokálně a ukládat svá vyšetřování mohli také lokálně. Autentizaci uživatelů bude částečně zajišťovat DotVVM. Ověření a ukládání údajů o uživateli do databáze bude nutné si navrhnout pro naše potřeby vlastní. Abychom mohli použít autentizaci za pomoci DotVVM, musí se nejprve naimportovat NuGet balíček `Microsoft.OWiN.Security.Cookies`. Poté je nutné v souboru *Startup.cs* OWIN Cookies zaregistrovat.[17]

Návrh, jak by měla vypadat část databáze pro autentizaci uživatele byl uveden v předešlé kapitole. Co by se mohlo k tomuto dodat, že do sloupce *Password* se ukládá hash hesla, a to především kvůli bezpečnosti. Hašovací funkce, používaná v rámci webového rozhraní, je SHA-256, takže očekáváme, že hash hesla bude mít délku 64 znaků.

Dále se v podkapitole budeme zabývat implementací přihlášení do webového rozhraní. Pohled pro přihlášení je definovaný v šabloně `login.dohtml` a viewmodel pro pohled je třída `LoginViewModel`. Jak vypadá výsledný přihlašovací formulář v prohlížeči se můžeme podívat na snímek obrazovky A.1. Formulář obsahuje pouze pole pro uživatelské jméno a heslo a tlačítko pro odeslání formuláře. V případě nějaké chyby se hláška o chybě zobrazuje mezi nadpisem a polem pro uživatelské jméno.

Při pokusu o přihlášení se postupuje následujícími kroky:

1. Provede se validace vstupu. V tomto případě kontrola, zda jsou vyplněna všechna vstupní pole, tedy je-li vyplněno uživatelské jméno a také heslo. Pokud se vypíše chybová hláška, není nějaké z polí vyplněno.

2. Provede se verifikace vyplněných přihlašovacích údajů. Verifikace obstarává fasáda `LoginFacade`, jenž metodou `VerifyCredetials()` ověří zadané přihlašovací údaje vůči databázi. V případě, že se údaje shodují s databází, získáme informace o uživateli, jemuž přihlašovací údaje patří.

3. Ze získaných informací se ověří, jestli je uživatelský účet aktivní. To znamená, že uživatel se s ním může přihlásit do systému.

4. Jak je popsáno v dokumentaci DotVVM, bude potřeba vytvořit objekt `ClaimIdentity`, který uchovává údaje o uživateli. V našem případě to bude uživatelské jméno a role, kterou má uživatel přiřazenou. Po vytvoření objektu reprezentující identity uživatele je třeba vytvořit autentizační token a uložit ho do cookies. V DotVVM se toto dělá přes metodu `Context.GetAuthentication().SignIn()`.

5. Přesměrování na úvodní stránku systému. Přesměrování se v DotVVM realizuje metodou `Context.RedirectToRoute()`, jíž předáme název cesty v routovací tabulce.

Úvodní stránka po přihlášení obsahuje v horní části lištu pro menu. Menu se zobrazuje na všech stránkách webového rozhraní kromě stránky s formulářem pro přihlášení. Když začneme z levé strany lišty, uvidíme text „Netfox“, jenž je odkazem na úvodní stranu. Za textem následují položky menu. V pravé části se potom nachází tlačítko pro odhlášení a uživatelské jméno jako odkaz na stránku s profilem uživatele. V obsahové části jsou ikony pro rychlé přejití, například na stránku se správou uživatelů, správou vyšetřování atd. Vedle ikon se zobrazuje pět posledně otevřených vyšetřování. Při kliknutí na vyšetřování se pak zobrazí pohled daného vyšetřování. Příklad úvodní stránky je možné nalézt na snímku A.3.

Přihlášený uživatel se může po ukončení práce ze systému odhlásit kliknutím na tlačítko s ikonou znázorňující odhlášení, jenž se nachází v pravém horním rohu obrazovky, jak můžeme vidět například na snímku A.3. Po kliknutí na tlačítko bude zavolána metoda, která provede odhlášení uživatele. V metodě se bude muset zneplatnit cookie s autorizačním tokenem, které se do cookies uložily při přihlášení. To se provede metodou `Context.GetAuthentication().SignOut()`. Po zneplatnění cookie je nutné uživatele přesměrovat na stránku s přihlašovacím formulářem.

5.5 Správa uživatelů

Správa uživatelů je implementována dvěma pohledy. Jedná se o pohled pro výpis všech uživatelů, ze kterého lze odebírat uživatele, přecházet na druhý pohled, ze kterého se dají upravit údaje stávajícího uživatele či přidat nového uživatele.

Pohled, sloužící pro výpis přehledu uživatelů, se vypisuje podle šablony `UserManagement.dothtml` a funkci viewmodelu zajišťuje třída `UserManagementViewModel`. Vzhled výpisu přehledu uživatelů je vyobrazen na snímku A.2. Na snímku si můžeme všimnout, že pohled se skládá z výpisu seznamu uživatelů, sekcí s tlačítky a polem pro filtrování nad výpisem. V seznamu uživatelů se v systému zobrazují tyto údaje:

- id uživatele,
- jméno, složené z křestního jména a příjmení,

- uživatelské jméno,
- přiřazenou roli.

U každého záznamu jsou pak dvě klikatelné ikony. Ikona tužky nás po kliknutí přesměruje na stránku detailu uživatele, kde se dají editovat jeho údaje.

Lišta s tlačítky nad seznamem uživatelů obsahuje tlačítko pro vytvoření nového uživatele. Po kliknutí nás přesměruje na detail nového uživatele, kde vyplníme jeho údaje. Druhé tlačítko slouží k hromadnému odstranění označených uživatelů. Napravo v liště je pole pro filtrování. Filtrovat lze podle uživatelského jména, křestního jména, příjmení nebo celého jména, jak je uvedeno v seznamu uživatelů.

Omezení, aby do správy uživatelů měli přístup uživatelé pouze s rolí administrátor se řeší anotací, kterou nabízí DotVVM. Veškeré akce týkající se uživatelů ve webovém rozhraní má na starost UI fasáda `UserFasade`. Pod akcemi si můžeme představit přidávání, odebírání, editace uživatelů či načtení seznamu uživatelů nebo pouze jednoho konkrétního.

Druhý pohled je definovaný šablonou `user.dothtml` a viewmodel třídou `UserViewModel`. Pohled slouží pro vytváření nového uživatele nebo editaci existujícího uživatele. To, jak pohled vypadá zachycuje snímek A.5. Z něho je patrné, že se jedná o běžný formulář se dvěma tlačítky. Kliknutím na tlačítko *Cancel* lze odejít zpět na výpis seznamu uživatelů bez vytvoření nového uživatele nebo provedení změn u uživatele. Kliknutí na tlačítko totiž způsobí přesměrování na pohled s výpisem uživatelů. Když se klikne na tlačítko *Save* dojde nejprve k validaci požadovaných vstupních hodnot. Validace se opět týká toho, jestli jsou všechny potřebné hodnoty vyplněny. Pokud validace proběhne úspěšně, tak se vykoná přidání nového uživatele nebo aktualizace údajů editovaného uživatele.

5.6 Profil uživatele

Profil uživatele zobrazuje informace o uživateli a umožňuje uživateli měnit své heslo k účtu. Profil uživatele je definován šablonou `profile.dohtml` a viewmodel obstarává třída `ProfileViewModel`. Jelikož se u tohoto pohledu jedná opět o práci pouze s uživateli, veškeré změny při načítání dat se řeší přes fasádu `UserFacade`.

Zmíněnou změnu hesla uživatel provede tak, že klikne na tlačítko *Change Password*. Kliknutí na tlačítko způsobí zobrazení modálního okna s formulářem obsahující pole pro současné heslo a pole pro nové heslo. Uživatel musí vyplnit obě pole, jinak při pokusu o změnu hesla dojde k chybě při validaci. Pokud validace vstupu projde v pořádku, tak se zkontroluje, zda se současné heslo shoduje s heslem v databázi. V případě, že hesla souhlasí, tak se změna hesla provede a modální okno se skryje. V případě neúspěchu se zobrazí chybová hláška, že vyplněné současné heslo nesouhlasí s heslem v databázi. Modální okno lze zavřít i bez toho, aby uživatel musel změnit heslo a to tak, že klikne na tlačítko *Cancel*.

5.7 Správa vyšetřování

Správa vyšetřování je opět jako správa uživatelů implementována dvěma pohledy. Jedná o pohled sloužící pro výpis seznamu dostupných vyšetřování a o pohled pro vytváření či editaci vyšetřování. Pro výpis seznamu vyšetřování je pohled zapsaný v šabloně `overview.dothtml` a viewmodel je definován třídou `InvestigationOverviewViewModel`. Provedení veškerých akcí týkajících se správy vyšetřování se zajišťuje UI fasádou `InvestigationFacade`. Snímek A.4 zobrazuje stránku s výpisem vyšetřování. Stránka se dělí na dvě části, a to

na levou a pravou část. Levá část je sloupec, ve kterém se zobrazují posledně navštívená vyšetřování. Posledně navštívená vyšetřování jsou řazena od nejnovějšího k nejstaršímu datu posledního navštívení. V pravé části se zobrazuje obsah stránky, tedy výpis seznamu dostupných vyšetřování a tlačítek vytvoření a smazání vyšetřování. V seznamu vyšetřování se zobrazují tyto informace:

- název vyšetřování,
- celé jméno vyšetřovatele,
- popis vyšetřování.

Vedle popisu daného vyšetřování jsou stejně jako u výpisu seznamu uživatelů ikony pro přesměrování na editaci a odebrání vyšetřování. Význam tlačítek nad výpisem seznamu vyšetřování je stejný jako u správy uživatelů. Jedno slouží pro vytvoření nového vyšetřování a druhé pro hromadné odebrání. Pokud uživatel klikne na název vyšetřovaní v seznamu vyšetřování nebo na vyšetřování v levém sloupci s posledně navštívenými vyšetřováními, dostane jeho úvodní stranu. Toto si popíšeme níže.

Druhý pohled se implementací do jisté míry podobá pohledu pro vytvoření či úpravu údajů o uživateli. Pohled je popsán šablonou `investigation.dothtml` a pozici viewmodelu zastává třída `InvestigationViewModel`. Pohled kromě úpravy dat o vyšetřování musí také umožnit výběr vyšetřovatelům, kteří mají do vyšetřování přístup. Proto můžeme vidět na snímku A.6, že v pohledu se nacházejí dvě záložky. V první záložce je nastavení základních hodnot pro vyšetřování. V druhé záložce je uživateli umožněno vybrat další uživatele, kteří budou mít přístup do nyní vytvářeného či upravovaného vyšetřování. Výběr se provede zaškrtnutím checkboxu u uživatele, jak můžeme vidět na snímku A.7. Nahoře jsou dvě tlačítka opět pro uložení změn či odejití bez uložení.

5.8 Propojení s Netfox Framework

Propojení webového rozhraní s Netfox Framework by se dalo rozdělit na dvě části:

1. Část, kdy je potřeba zajistit extrahování konverzací ze vstupních souborů s odposlechnutou komunikací a extrahování forenzních dat z extrahovaných konverzací.
2. Část zabývající se zobrazením extrahovaných konverzací a extrahovaných forenzních informací.

Realizace extrahování konverzací a export forenzních dat probíhá následovně. Napojení na Netfox Framework za účelem extrahování konverzací a extrahování obstarává třída `NetfoxAPIFacade`. Takže se nejprve popíše třída `NetfoxAPIFacade` a základní struktura stránky v sekci vyšetřování a až poté celý proces extrahování konverzací a forenzních dat. Třída `NetfoxAPIFacade` provádí veškerou svou inicializaci propojení s Netfox Framework v konstruktoru. Konstruktoru je potřeba předat pomocí parametru:

- Kontejner s registrovanými komponentami, které bude vyžadovat.
- Id vyšetřování, pro které je potřeba extrahování konverzací a forenzních dat provést.
- Cestu ke složce odkud se implementace spustila. Tuto cestu budeme potřebovat kvůli složení cesty k samotnému vyšetřování.

Nejprve se v konstruktoru zjistí, jaké jsou všechny dostupné snoopery. Poté se nastaví informace o vyšetřování pomocí třídy `InvestigationInfo`. V objektu třídy `InvestigationInfo` je především nutné nastavit správné:

- id vyšetřování,
- název,
- cestu, kam se ukládají adresářové struktury k jednotlivým vyšetřováním.

Nastavení těchto údajů je nesmírně důležité, protože se pomocí nich ve třídě `InvestigationInfo` generují cesty v rámci konkrétního vyšetřování, například cesta, kde jsou uloženy vstupní soubory se zachycenou komunikací. Následně se musí ještě nastavit kontext k databázi patřící k danému vyšetřování. Po nastavení výše uvedených vlastností se provede kontrola, zda daná databáze existuje a pokud neexistuje, tak se vytvoří. Tímto krokem končí inicializace třídy `NetfoxAPIFacade` a nyní se může spustit buď zpracování vstupních souborů se zachycenou komunikací na konverzace pomocí metody `ProcessCapture()` nebo se může spustit extrahování dat metodou `ExportData()`.

V sekci pro konkrétní vyšetřování je rozložení stránky následující. Stránka je opět rozložena na dvě části, a to na levou a pravou část. Příklad můžeme vidět na snímku A.8, jenž zobrazuje úvodní stránku sekce vyšetřování. V levé části je opět sloupec, kde ve vrchní části nalezneme název vyšetřování, v jakém se zrovna nacházíme. Vedle toho se nachází ikona dvou šipek, jenž slouží jako tlačítko pro opětovné načtení údajů. Pod názvem se nacházejí dvě rozbalovací menu. V menu *Captures* se nachází odkazy na detaily zpracovaných vstupních souborů. Vedle nadpisu se vyskytuje tlačítko se zelenou ikonou plus. Při kliknutí se zobrazí modální okno pro vložení souboru s komunikací. Menu *Exports* obsahuje pak odkazy na pohledy, kde jsou zobrazena vyexportovaná data. Pravá část stránky je určena pro zobrazení samotného obsahu. Nahoře se pak zobrazuje nadpis, na jakém pohledu v sekci vyšetřování se uživatel nachází.

Pro exportování dat ze zachycené komunikace musí nejprve uživatel vložit vstupní soubor. Uživatel musí tedy prvně kliknout na ikonu zeleného plus. Zobrazí se modální okno obsahující komponentu z Dotvvm, která se nazývá FileUpload[4]. Pro účely této práce jsem vybral komponentu z balíku business pack, který je placený. Komponenta FileUpload z balíčku business pack umí oproti své základní variantě, která je zdarma, drag & drop, což je jistě pro každého uživatelsky přívětivější než zadávat cestu ručně pomocí prohlížeče souborů. Komponenta umožňuje nahrání více souborů současně.

Pro použití komponent je potřeba nastavit cestu k dočasnému úložišti, kam se má nahraný soubor dočasně uložit. To se nastavuje v souboru *Startup.cs* kde, jak bylo již zmíněno výše v této kapitole, je soubor obsahující konfiguraci webové aplikace. Po nastavení dočasné složky bude potřeba zajistit zpracování vloženého souboru. To uděláme tak, že komponentě FileUpload můžeme nastavit atribut `UploadCompleted`, který řekne, jakou metodu má ve viewmodelu zavolat po dokončení nahrání souborů na webový server do složky s dočasnými soubory.

V tomto případě se po nahrání souborů na webový server volá metoda `Process()`. V metodě `Process()` se pro každý nahraný soubor volá metoda `StartProcessCapture()`, jenž spustí HangFire úlohu, která zpracuje následujícím způsobem vstupní soubor. Jako vstup dostane úloha id vyšetřování, k němuž soubor patří a cestu, kde se nachází nahraný soubor. Úloha nejprve překopíruje soubor z dočasného úložiště do složky se vstupními soubory

[4] https://www.dotvvm.com/docs/controls/businesspack/FileUpload/latest

daného vyšetřování. Po překopírování souboru úloha spustí zpracování konverzací ze vstupního souboru se zachycenou komunikací a to pomocí metody `ProcessCapture()` ze třídy `FrameworkController` a předá metodě umístění.

Po dokončení zpracování HangFire úlohy se objeví nahraný soubor v levé části. Po kliknutí na odkaz nás přesměruje na stránku s detaily o souboru. Tento pohled bude popsán později v druhé části propojení.

Extrahovat data je možné dvěma způsoby. Buď se extrahují data ze všech vložených souborů ve vyšetřování naráz nebo z konkrétního souboru. V případě, že se má extrahovat ze všech souborů, tak se na úvodní stránce vyšetřování vyberou protokoly pro export a klikne se na tlačítko *Export Data*.

Pro extrahování forenzních dat se musí kliknout na tlačítko *Export Data*. To zapříčiní vytvoření nové HangFire úlohy, jenž spustí metodu `ExportData()` z `HangfireFacade`. Metodě se předá:

- id vyšetřování,
- vybrané protokoly pro exportování,
- cesta, kam se ukládají adresářové struktury k jednotlivým vyšetřováním.

`HangfireFacade` slouží jako obalovací třída pro `NetfoxAPIFacade`, protože spouštěný objekt v HangFire musí mít konstruktor bez parametrů. Následně tedy v `HangFireFacade` připraví kontejner a vytvoří objekt třídy `NetfoxAPIFacase`. Metoda `ExportData()` ve vytvořeném objektu nejprve zjistí všechny soubory se zachycenou komunikací ve vyšetřování. Pak se začne zpracování každého z těchto souborů zvlášť. U každého souboru se určí, jaké z požadovaných protokolů nebyly exportovány, a to z toho důvodu, aby se neduplikovaly exportované objekty v databázi jak je tomu u desktopové verze. Nyní se vyberou všechny L7 konverzace, jenž patří do daného souboru. Následně použije metodu `ExportData()` třídy `FrameworkController`, které se předá:

- snoopery, jenž má při exportování použít,
- vybrané L7 konverzace, ze kterých se mají extrahovat forenzní data,
- cesta ke složce, kde se mají ukládat exporty.

Tato metoda se volá celkem dvakrát. Při druhém volání se místo L7 konverzací, předají exportované objekty z prvního volání. Důvodem je extrahování, například komunikace přes Facebook, kde potřebujeme extrahované objekty z HTTP protokolu. Po provedení extrahování je ještě nutné přidat záznam k danému souboru, jaké protokoly se z něho exportovaly a dopočítají se statistiky exportovaných dat. Pokud se rozhodneme využít možnost extrahovat data pouze z jednoho konkrétního souboru, tak se musí pro spuštění exportovaní přistoupit do detailu souboru se zachycenou komunikací a v záložce *Export* vybrat opět protokoly pro export a stisknout tlačítko *Export Data*. To zapříčiní zavolání metody `ExportData()`, které kromě výše uvedených parametrů předáme ještě id souboru určeného pro exportování. Pak se exportuje pouze z konkrétního souboru. Události po exportu jsou pak stejné jako u extrahování všech souborů naráz.

Detail souboru se zachycenou komunikací

Na detail souboru se zachycenou komunikací se můžeme dostat, jak již víme, přes rozbalovací menu *Captures*. Stránka detailu je rozdělena pomocí záložek do sekcí:

- statistiky o souboru se zachycenou komunikací,
- výskyt transportních protokolů,
- výskyt aplikačních protokolů,
- L3 konverzace,
- L4 konverzace,
- L7 konverzace,
- rámce,
- exportování.

Začneme-li záložkou se statistikami, tak se jedná o sekci, kterou uživatel uvidí jako první. Tato záložka obsahuje např. údaje o tom kolik je v komunikaci konverzací daného typu, kolik rámců prošlo daným směrem, kolik rámců obsahuje soubor atd. Jak tato sekce vypadá můžeme vidět na snímku A.9.

Další dvě sekce jsou prakticky totožné. Jedná se o zobrazení výskytu transportních a aplikačních protokolů. Obě sekce zobrazují v levé části graf, který reprezentuje procentuální zastoupení jednotlivých protokolů. Na pravé straně se nalézá tabulka, jenž obsahuje název protokolu použitého pro komunikaci a v kolika procentech je přesně v komunikaci zastoupen.

Sekce pro L3, L4 a L7 konverzace se skládají z filtru a tabulky se zobrazenými údaji o konverzaci, příklad pohledu je na snímku A.10. Celkem se vypisuje 15 záznamů najednou. Pokud je tato hodnota překročena, tak se dole pod tabulkou ukáže stránkovač, který nám umožní přecházet mezi stránkami s konverzacemi. Když uživatel klikne na řádek v tabulce, vyskočí modální okno s detailem konverzace. Z detailu se lze dozvědět nejen základní informace o konverzaci, ale i kupříkladu u L4 konverzace jsme schopni se dozvědět, do jaké L3 konverzace patří a jaké L7 konverzace a rámce obsahuje daná L4 konverzace. Filtrovat se konverzace dají podle:

- adresy a portu zdroje a cíle,
- výskytu v určitém časovém rozmezí,
- rozmezí počtu rámců v konverzaci,
- rozmezí počtu přenesených bajtů.

Filtrování podle adresy a portu lze provést zapsáním těchto údajů do textového pole. Zadávání časového rozmezí je řešeno pomocí datepickeru. Datepicker v tomto případě umí nejen datum, ale také čas. Pro filtrování podle rozmezí počtu rámců a bajtů v konverzaci jsou použity tzv. slidery, pokud ale uživatel chce zadat hodnoty rozmezí číslem, tak mu to je také umožněno.

Záložka *Frames* v sobě skrývá zobrazení rámců nesených ve zpracovaném souboru. Z výpisu je uživatel schopen vyčíst údaje:

- pořadové číslo v souboru,
- čas zachycení,

- zdrojová a cílová adresa,
- transportní protokol,
- velikost přenesených dat v rámci.

Jako u zobrazení konverzací je možné se dostat na zobrazení detailu rámce. Na detailu je možné zjistit základní informace o rámci, v jaké konverzaci se rámec nachází a přenášená data v rámci v hexadecimálním tvaru a ASCII kódování. Stejně jako u konverzací je možné rámce filtrovat a to podle:

- zdrojové a cílové adresy,
- výskytu v časovém rozmezí,
- velikosti rámce.

Poslední záložkou v pohledu je záložka *Export*. Jak název napovídá bude se záložka týkat exportování dat daného souboru. Jak můžeme vidět ze snímku A.11, jsou zde tři boxy. V prvním boxu se zobrazují protokoly, jež byly ze souboru již extrahovány. Pokud by uživatel chtěl extrahovat u souboru již exportovaný protokol, musí na něj v tomto boxu kliknout. To způsobí odebrání protokolu z exportovaných protokolů pro daný soubor. Druhý box slouží pro nahrání klíče pro šifrovanou komunikaci. Po kliknutí na tlačítko *Add Cypher Key* se zobrazí modalové okno s komponentou pro nahrání souboru s klíčem. Po nahrání se klíč přidá ke všem L7 konverzacím obsažených ve vybraném souboru se zachycenou komunikací. Poslední box pak slouží k výběru a extrahování dat z protokolů pouze z daného souboru.

Zobrazení exportovaných dat

Zobrazování exportovaných dat z protokolů se ve webovém rozhraní řeší dvěma způsoby. Jedním je definice pohledů přímo v projektu *Netfox.Web.App*. Tento způsob byl použit pro zobrazení exportu zpráv, emailů a hovorů. Druhým způsobem je definovat pohledy v projektu určeném danému protokolu. Po přeložení je pak nutné nastavit zkopírování pohledů po kompilaci do projektu *Netfox.Web.App*.

Pro přistoupení je ještě potřeba přidat do routovací tabulky záznamy o přidaných pohledech. Za tímto účelem se musela nadefinovat třída `NetfoxRouteStrategy`, která toto bude dělat. Třída vychází ze třídy `DefaultRouteStrategy`. Vůči rodičovské třídě se liší pouze v metodě `GetRouteUrl`. Metoda je upravena, aby se zajistilo, že pohledy exportovaných dat budou mít v URL id vyšetřování.

Bylo potřeba předělat současné snoopery, protože snoopery kromě modelů a třídy pro extrahování v sobě mají součásti z Netfox Detective vztahující se k technologii WPF (Windows Presentation Foundation), kterou používá desktopová aplikace, například viewmodely a pohledy. Proto bylo nutné oddělit tyto části do samostatného projektu nazvaného například *SnooperFTP.WPF*. Dále se u těchto projektů muselo nastavit vykonání příkazu po sestavení projektu, který nakopíruje sestavený projekt do složky s binárními soubory Netfox Detective, jako tomu bylo u původního projektu snooperu.

Poté co se oddělily prvky WPF od snooperů, tak se mohou vytvořit projekty nazvané, kupříkladu *SnooperFTP.WEB*, které budou obsahovat výše zmíněné pohledy, viewmodely k těmto pohledům, UI fasády, DTO objekty, a definici mapování.

Pohledy přehledu exportů a detailu exportu pro konkrétní snoopery definované například v projektu *SnooperFTP.WEB* vyžínají masterpage, která je definovaná v projektu *Netfox.Web.App*. Každý pohled využívá jinou masterpage. Pro přehled exportů je

masterpage uložena v souboru `ExportBase.dotmaster` a funkci viewmodelu v případě *SnooperFTP.WEB* zajišťuje třída `ExportFTPViewModel`, jenž dědí třídu ExportBase. Pohled pro přehled exportů vypisuje jednotlivé FTP příkazy a pokud se jedná o přesun souboru, je u výpisu odkaz pro stažení. Pohled pro detail u FTP není definován, protože nebyl potřeba. Dá se ale říci, že kde je detail exportu definován, tak používá masterpage `BlankLayout.dotmaster`. U detailů se využívá masterpage `BlankLayout.dotmaster` z důvodu, že detaily exportů se zobrazují do modalového okna stejně jako detaily konverzací. Stránku pro detail pak lze upravit dle svého. Detaily jsou definovány například u protokolů HTTP a DNS.

Některé pohledy umožňují stáhnout extrahovaná data, bohužel některé soubory nemají určený formát příponou. Tím vzniká problém s webovým serverem, protože nám k těmto souborům nedovolí přístup, což je správné z hlediska bezpečnosti. Proto bylo potřeba vytvořit pomocný pohled pro stahování těchto souborů z vyšetřování. Pohled `DownloadFile` je omezen pouze na složky s vyšetřováními. Přes URL se zadají parametry:

- id vyšetřování, ze kterého chceme soubor,
- název stahovaného souboru,
- cesta k souboru v rámci vyšetřování,
- typ souboru, jestli se jedná kupříkladu o obrázek, text, binární data.

Po zadání a ověření parametrů vrátí požadovaný soubor.

Kapitola 6

Testování a srovnání s jinými nástroji

V této kapitole se budeme zabývat testováním implementace popsané v předchozí kapitole. Mezi testovanými částmi tedy jsou správa uživatelů, správa vyšetřování a propojení webového rozhraní a Netfox Framework. Dále bude provedeno srovnání implementace webového rozhraní s nástroji zmíněnými v kapitole 2. Porovnání bude provedeno na cvičném případu. Z příkladu zjistíme, co vše jsou nástroje schopny vyextrahovat z komunikace. U porovnání budeme srovnávat postup vyšetřování z pohledu uživatelské přívětivosti.

6.1 Testování implementace

Veškeré testy jsou obsaženy v projektu pojmenovaném *Netfox.Web.Tests*. Testování probíhá tzv. integračními testy. Každá testovaná část má svou testovací třídu. Tyto třídy pak dědí ze třídy `TestViewModelBase`. `TestViewModelBase` obsahuje metody, jenž se spouští na začátku a na konci všech testů. Na začátku testování bázová třída `TestViewModelBase` zajistí zaregistrování všech komponent pro testování do kontejneru a nastavení mapování mezi objekty. Před každým testem se smaže testovací databáze, pokud nebyla smazána při předchozím testování v důsledku chyby. Poté se vytvoří nová databáze a vloží se do ní záznamy o rolích v systému.

Po ukončení každého testu se pak maže testovací databáze. Na konci celého testování se ještě provede vyprázdnění testovacího kontejneru.

6.2 Testování správy uživatelů

Testování obstarává třída `UserManagementTests`. Jak již bylo řečeno, rodičovská třída zajišťuje veškeré rutiny před a po testování. V rámci testování správy uživatelů jsou testovány tyto případy:

- přidání nového uživatele,
- odebrání uživatele,
- odebrání vybraných uživatelů,
- úprava údajů o uživateli,

- odebrání všech uživatelů kliknutím na vybrat vše,
- pokus o přihlášení s nevyplněním hesla,
- pokus o přihlášení se správnými údaji,
- pokus o přihlášení s nesprávným heslem,
- pokus o přihlášení s uživatelem, který není aktivní,
- změna hesla.

Způsob testování je z většiny stejný pro všechny testy uživatelské správy. V testech se testuje chování viewmodelů pro jednotlivé pohledy. Při testování viewmodelů se vychází z dokumentace DotVVM[1]. Vytvoříme objekt, jenž nám supluje úlohu kontextu. Do objektu kontextu se nastaví vstupní parametry, které předáváme pomocí url a v případě potřeby záznam do routovací tabulky, pokud má viewmodel provádět přesměrování. Pak se vytvoří viewmodel a provedou se nad ním požadované operace. Posléze se zkontrolují výstupy toho, co viewmodel vykonal.

6.3 Testování správy vyšetřování

Pro testování správy vyšetřování je určena třída `InvestigationTests`. Třída obsahuje část podobných testů jako třída pro testování správy uživatelů. Jedná se například o testování přidání vyšetřování, odebírání vyšetřování. Za zmínku stojí pouze test zobrazení vyšetřování různým uživatelům.

Testování zobrazení vyšetřování provádí metoda `TestInvestigationList()`. V metodě se otestuje, zda se administrátorovi zobrazí všechny vyšetřování, pak jestli se vyšetřovateli zobrazí pouze vyšetřování, která vlastní a posledním případem je situace, jestli se uživateli zobrazí vyšetřování, do kterých je přiřazen. Metoda tedy musí vytvořit tři uživatele a to, administrátora s uživatelským jménem *admin* a dva vyšetřovatele *owner* a *user*. Následně vytvoří tři vyšetřování, kdy u jedno vyšetřování je vlastníkem administrátor a u zbylých dvou je vlastníkem vyšetřovatel *owner* a k jednomu z těchto vyšetřování má povolený přístup vyšetřovatel *user*. Ještě je potřeba vytvořit instanci viewmodel v tomto případě instanci třídy `InvestigationOverviewViewModel`.

Potom se pro každého ze tří uživatelů volá metoda `FillDataSet()`, které se předá reference na objekt z vlastnosti `Investigations` a uživatelské jméno uživatele. Pak se zkontroluje počet záznamů, které se získají pro uživatele metodou `FillDataSet()`. Prvně se proces provede pro administrátora, u kterého se očekává, že bude mít zobrazena všechna vyšetřování, což jsou tři vyšetřování. Další na řadě je vyšetřovatel *owner*, který vlastní dvě vyšetřování a v žádném dalším není přiřazený, takže se očekává, že bude mít zobrazené pouze dvě vyšetřování. Posledním testovaným je vyšetřovatel *user*. Ten žádné vyšetřování nevlastní a je přiřazený k jednomu vyšetřování. Mělo by se mu tedy zobrazit pouze jedno vyšetřování. Pokud počty všech vyšetřování u všech uživatelů souhlasí, tak test dopadl úspěšně.

6.4 Testování propojení s Netfox Framework

Poslední testovanou částí, která je v rámci otestování implementace testována, je propojení webového rozhraní a Netfox Framework. Propojení se testuje několika testy. V testech se

[1] https://www.dotvvm.com/docs/tutorials/advanced-testing-viewmodels/1-1

nejprve otestuje proces zpracování souboru se zachycenou komunikací na konverzace. Poté se ještě v testu otestuje exportování forenzních dat. Testy pro otestování propojení se nacházejí ve třídě `ExportTests`. Třída jako všechny předešlé třídy dědí třídu `TestViewmodelBase`. Při inicializaci testů je ještě nutné oproti běžné inicializaci, vytvořit nový kontejner se všemi závislostmi pro třídu `NetfoxAPIFacade`, která prování zpracování komunikace. V testech propojení s Netfox Framework se v první řadě vytvoří testovací uživatel a vyšetřování. Pak dojde na vytvoření instance třídy `NetfoxAPIFacade` a otestování, jestli instance byla vytvořena. Každý test pokračuje spuštěním zpracování souboru se zachycenou komunikací. Pro každý test se používají soubory umístěné v testovací sadě pro testování Netfox. Zpracování vstupního souboru započne zavoláním metody `ProcessCapture()` z třídy `NetfoxAPIFacade`. Po zpracování se zavolá metoda `GetCaptureList()`, aby se získal seznam zpracovaných souborů. Zde se zkontroluje, zda se v seznamu vyskytuje jeden prvek a také se zkontrolují počty vyextrahovaných konverzací.

Teď zbývá v každém testu otestovat už jen extrahování forenzních dat. Aby se extrahování dat započalo, zavolá se metoda `ExportData()` z třídy `NetfoxAPIFacade`. Po ukončení extrahování se zkontroluje na základě počtu vyexportovaných objektů, zda se všechny objekty z komunikace vyextrahovaly. Pokud počet extrahovaných objektů souhlasí s očekávaným počtem, tak test skončí úspěšně.

6.5 Zpracování komunikace pro porovnání

Srovnání se provede vyšetřením cvičného případu a zpracováním dalších vstupních souborů z testovací sady pro testování Netfox. Případ použitý v tomto srovnání se nazývá M57[2]. V tomto příkladu je zaznamenána historie prvních čtyř týdnů fiktivní společnosti M57 patents. Příklad lze použít nejen pro forenzní analýzu síťového provozu, ale i analýzu pevného disku. Během zmíněných čtyř týdnů byla zachycena komunikace o velikosti zhruba 4,54 GB. Komunikace je uložena ve 49 souborech.

Vyšetření případu provedeme prvně na vytvořeném webovém rozhraní. Po přihlášení vytvoříme vyšetřování nazvané M57. Do vyšetřování nyní nahrajeme všechny soubory se zachycenou komunikací. Nahrání provedeme označením souborů v průzkumníkovi a přetažením do vyznačené oblasti. Nyní se musí počkat, než se zpracují vstupní soubory na konverzace. Po dokončení zpracování označíme protokoly k vyexportování a spustíme exportování. Po extrahování můžeme zjistit, že se vyexportovalo 28 emailů a nějaká data z protokolů HTTP a FTP. Když přejdeme na pohled s emaily, z výpisu zjistíme základní údaje:

- čas zachycení,
- zdrojovou a cílovou adresu,
- příjemce a odesílatele emailu,
- příjemce kopie a skryté kopie emailu,
- předmět emailu.

Pro bližší informace se musí přejít na detail vyexportovaného emailu, kde si kromě základních informací můžeme přečíst obsah emailu a případně stáhnout přílohu.

Pro porovnání, co dané nástroje zjistily z odposlechnuté komunikace, se můžeme podívat do tabulky 6.2. V tabulce jsou zaznamenány počty jednotlivých vyexportovaných dat.

[2] https://digitalcorpora.org/corpora/scenarios/m57-patents-scenario

Tabulku jsem sestavil pro nástroje NetworkMiner, Xplico a NetWitness Investigator. Wireshark a Cloudshark jsem vynechal, protože nejsou schopny extrahovat soubory nebo emaily. V tabulce jsou zapsány počty extrahovaných souborů. Zde započítávají dokumenty, spustitelné soubory, obrázky atd. Dále ještě tabulka obsahuje počty obrázků a emailů.

Jelikož zachycená komunikace v cvičném případu M57 obsahuje v převážné většině komunikaci protokolem HTTP a emailové pošty, tak jsem do porovnání zařadil i soubory, které obsahují následující aplikační protokoly:

- FTP,
- SIP,
- RTP.

Dále jsem u webového rozhraní ještě vyexportoval zprávy z protokolů pro přenos chatových zpráv, jako jsou OSCAR, YMSG, XMPP viz tabulka 6.2. V tabulce 6.1 je uvedeno o jaké porovnání se jedná a jaký vstupní soubor byl použit. V tabulce 6.2 nalezneme zjištěné hodnoty.

6.6 Porovnání s Wireshark

Když provádíme vyšetřování s Wiresharkem, narazíme na jeho omezení oproti webovému rozhraní hned na začátku. Wireshark totiž umožňuje uživateli analyzovat jeden soubor v danou chvíli. Do webového rozhraní můžeme nahrát více souborů současně a zobrazit si extrahovaná data ze všech souborů najednou. Co se týče informací o vstupním souboru poskytuje Wireshark seznam konverzací, použití protokolů v komunikaci a výpis koncových uzlů. Tyto informace nám vesměs poskytne i webové rozhraní. Pokud se soustředíme na provádění analýzy přenášených dat, tak je lepší webové rozhraní, protože za nás data z jednotlivých rámců extrahuje a my je nemusíme hledat přímo v komunikaci. To Wireshark kompenzuje opravdu propracovanými filtry, jenž vyšetřovateli velmi usnadní práci s hledáním důležitých informací z komunikace. Další výhodou webového rozhraní je také sdílení souborů v rámci vyšetřování. Závěrem bych řekl, že v obou nástrojích lze důležité informace pro vyšetřování dohledat s tím, že při použití Wiresharku musí mít vyšetřovatel nastudováno chování různých protokolů na rozdíl od webového rozhraní, které tuto práci udělá za nás.

6.7 Porovnání s Cloudshark

Webová služba Cloudshark na tom je podobně jako Wireshark v porovnání s webovým rozhraním pro Netfox Framework. Do Cloudsharku můžeme stejně jako u webového rozhraní nahrát více souborů naráz. Narazíme zde na stejný problém jako u Wiresharku, že v daný okamžik může uživatel analyzovat pouze jeden konkrétní soubor. Cloudshark oproti webovému rozhraní nabízí zobrazení mapy, kde zvýrazní státy, ze kterých koncové uzly pochází. Dalším vylepšením je možnost vykreslit diagram komunikace. Dále je to ale stejné jako u Wiresharku. Vyšetřovatel musí veškeré informace hledat sám i zde práci usnadní filtr, ale pořád to není pohodlné jako u webového rozhraní.

6.8 Porovnání s NetworkMiner

NetworkMiner stejně jako webové rozhraní umožňuje vložit více souborů najednou a zároveň umožňuje vložit soubory pomocí drag & drop. Při zpracování souborů se zachycenou komunikací lze sledovat, jak velká část komunikace je již zpracována. U webového rozhraní se uživatel dozví pouze, že se už extrahování forenzních dat dokončilo. Pokud porovnáme, jaké informace nám poskytuje NetworkMiner oproti webovému rozhraní, tak za zmínku například stojí zobrazení jednotlivých koncových uzlů se statistikami o tomto uzlu. Webové rozhraní umožňuje zobrazit navíc kupříkladu zobrazení detailů rámců nebo přehled použitých protokolů při komunikaci. Asi největší výhodou webového rozhraní je, že extrahovaná data se ukládají v tomto případě do databáze. To potom umožňuje kdykoli k extrahovaným datům přistoupit, aniž by se musel znova zpracovat soubor se zachycenými daty. U NetworkMineru se data uloží pouze do paměti a po vypnutí aplikace jsou data ztracena. Při dalším otevření aplikace se musí znova vložit soubor se zachycenou komunikací a musí se znovu zpracovat. To potom vyšetřovatele zdržuje od samotného vyšetřování.

Při porovnání počtu exportovaných dat je zde stejný počet extrahovaných emailů z protokolů POP, IMAP a SMTP jako je tomu u webového rozhraní, tedy 28, jak je uvedeno v tabulce 6.2 a u NetworkMineru se pak ještě navíc vyexportovaly zprávy z webmailu. U NetworkMineru se dále ještě vyexportovaly přístupové údaje k emailům. Pokud se podíváme na počty souborů a obrázků, tak jsou srovnatelné, v NetworkMineru se navíc extrahují certifikáty použité v šifrovaném spojení.

Když porovnáme data v tabulce 6.2, tak zjistíme, že vyexportovaných FTP příkazů je u NetworkMinera více. Bohužel ale NetworkMiner nevyexportoval žádný nesený soubor tímto protokolem.

6.9 Porovnání s Xplico

V porovnání naimplementovaného webového rozhraní s Xplico se prvně zaměříme na porovnání správy uživatelů a způsobu vyšetřování. Ve výchozím stavu má Xplico dvě role, a to roli administrátora a vyšetřovatele. Administrátor může přidávat či odebírat uživatele a může jim změnit heslo. Editace tady není možná jako ve webovém rozhraní pro Netfox Framework. Dále je možné vytvořit skupiny pro uživatele. V rámci skupiny mohou uživatelé tvořit případy viz podkapitola 2.3. K případu mohou pak jen uživatelé ve skupině a administrátor. V Xplicu je umožněna spolupráce uživatelů, ale není to pohodné jako v implementaci webového rozhraní, kde se volí uživatelé, kteří mají mít do daného vyšetřování přístup. Navíc v Xplicu může být uživatel přiřazen pouze do jedné skupiny. Přiřazení má v moci pouze administrátor při vytváření uživatele.

Teď si porovnáme průběh vyšetřování. Pro vyšetřování si uživatel musí nejprve založit případ a v něm sezení, jak je zmíněno v kapitole 2.3. V naimplementovaném webovém rozhraní stačí založit vyšetřování. V sezení pak vkládáme soubory se zachycenou komunikací. To je taky značně nekomfortní vůči webovému rozhraní, protože se soubory musejí nahrávat po jednom a musíme k nim zadat cestu pomocí dialogu. Není tady možnost nahrát soubory pomocí drag & drop.

Poté, co se soubory na vstupu zpracují, můžeme se podívat na statistiky, co vše se extrahovalo z komunikace. Xplico tuto statistiku má podrobnější vůči webovému rozhraní. Vzhledem k tomu, že se v testovacím případu nachází především emaily a komunikace pomocí HTTP, tak jsem porovnal zobrazení těchto dat. U protokolu zobrazují obě řešení prakticky stejné informace. Jedná se o přenesené obrázky, HTTP zprávy a jejich detail

a přenesené soubory tímto protokolem. U zobrazení emailů tomu je stejně. V přehledu se zobrazují základní informace o emailu, jako jsou příjemce, odesílatel a předmět. Na detailu emailu si můžeme přečíst obsah emailu či stáhnout přílohy, pokud email nějaké obsahuje.

Při exportování dat mělo Xplico problém se zpracováním souboru se zachycenou komunikací ze dne 7. 12. 2009. Proto chybí jeden email, který byl v tento den poslán a také samozřejmě obrázky a soubory. Jinak by Xplico exportovalo zhruba stejný počet souborů jako ostatní nástroje.

U FTP se oproti webovému rozhraní vyexportovalo více příkazů, ale méně souborů viz tabulka 6.2. Menší počet vyexportovaných souborů je způsoben tím, že Netfox Framework exportuje navíc do souborů i textové odpovědi, například odpověď na dotaz obsahu složky. V exportování z protokolů SIP a RTP, jak si při srovnání v tabulce 6.2 můžeme všimnout, webové rozhraní poskytuje více informací. Celkově vyexportovalo více hovorů a ve většině případů i více nahrávek. Aby se hovor započítal do tabulky, muselo se dát zjistit alespoň kdo je volající a volaný účastník hovoru a délku hovoru. U nahrávek bylo kritériem možnost přehrát nahrávku.

6.10 Porovnání s NetWitness Investigator

NetWitness Investigator má nevýhodu oproti webovému rozhraní, a to zobrazení extrahovaných dat. Data se dají totiž pouze extrahovat soubory z komunikace na předem určené místo a uživatel tak musí procházet extrahovaná data mimo aplikaci. Naopak výhodou oproti webovému rozhraní jsou možnosti filtrování komunikace. Dokonce nám NetWitness Investigator umožňuje do jisté míry spolupráce v týmu, protože nabízí možnost připojit kolekci uloženou na vzdáleném serveru.

Po extrakci dat nám NetWitness Investigator extrahovaná data rozdělí do složek. Typy, podle kterých jsou roztříděny do složek jsou následující:

- archivní,
- spustitelné
- zvukové,
- dokumenty,
- obrázky,
- video,
- soubory týkající se webu jako jsou HTML soubory, soubory s kaskádovými styly či soubory se skriptovacím jazykem javascript,
- ostatní, jenž nepatří ani do jedné z předchozích typů.

Pro získání počtu vyexportovaných souborů jsem tedy musel sečíst všechny soubory ze všech složek. Důvod, proč není vyplněná hodnota počtu emailů v tabulce 6.2 je ten, že NetWitness Investigator emaily neextrahuje. Tím je myšleno, že nejsou k dispozici informace o textu emailu, odesílateli a příjemci.

Když porovnáme, kolik se vyexportovalo souborů z tabulky 6.2, tak dojdeme k závěru, že se vyexportoval prakticky stejný počet souborů. Rozdíl je vždy o jeden soubor méně u webového rozhraní. Z tabulky 6.2 je patrné, že NetWitness Investigator má problém vyexportovat nahrávky hovorů z některých použitých kodeků stejně jako Xplico.

	Porovnání	Vstup
1	Počet souborů / obrázků / emailů z M57	Všechny soubory s komunikací z M57
2	Počty FTP příkazů / souborů 1	ftp_xkarpi03_01.pcap
3	Počty FTP příkazů / souborů 2	ftp_xkarpi03_02.pcap
4	Počty FTP příkazů / souborů 3	ftp_xkarpi03_03_upload.pcap
5	Počty FTP příkazů / souborů 4	ftp_xkarpi03_04_delete.pcap
6	Počty FTP příkazů / souborů 5	ftp_xkarpi03_05_download.pcap
7	Počty FTP příkazů / souborů 6	ftp_xkarpi03_06_text_upload.pcap
8	Počet hovorů / nahrávek hovorů 1	sip-rtcp.pcap
9	Počet hovorů / nahrávek hovorů 2	martin-to-petr-gsm.pcap
10	Počet hovorů / nahrávek hovorů 3	martin_nb-to-petr.pcap
11	Počet hovorů / nahrávek hovorů 4	MLUVI_G723_G729_GSM_G711.cap
12	Počet zpráv z protokolu YMSG	ymsg_without_file.cap
13	Počet zpráv z protokolu XMMP	xmpp_nosecure.cap
14	Počet zpráv z protokolu OSCAR	icq1.cap
15	Počet zpráv z aplikace Minecraft	xberan33_minecraft_tell_cs.pcap

Tabulka 6.1: Tabulka vstupních souborů pro porovnání nástrojů.

	Webové rozhraní	NetworkMiner	Xplico	NetWitness Investigator
1	81647 / 27374 / 28	81852 / 27431 / 39	89606 / 28625 / 26	145449/ 27715 / –
2	10 / 2	14 / 0	14 / 1	– / 3
3	21 / 4	28 / 0	28 / 1	– / 5
4	24 / 4	21 / 0	21 / 2	– / 5
5	14 / 3	17 / 0	17 / 1	– / 4
6	24 / 3	17 / 0	17 / 2	– / 4
7	12 / 3	16 / 0	16 / 1	– / 4
8	1 / 2	– / –	1 / 3	– / 3
9	1 / 2	– / –	0 / 0	– / 3
10	2 / 4	– / –	1 / 3	– / 3
11	0 / 8	– / –	0 / 6	– / 3
12	2	–	–	–
13	26	–	–	–
14	3	–	–	–
15	1	–	–	–

Tabulka 6.2: Tabulka vstupních souborů pro porovnání nástrojů.

Kapitola 7

Závěr

Hlavním cílem v rámci práce bylo implementovat webové rozhraní pro forenzní analýzu síťového provozu. Webové rozhraní využívá k získávání forenzních dat z komunikace Netfox Framework. Nejstěžejnějším frameworkem pro celou implementaci je DotVVM. Framework podporuje vývoj webových aplikací pro obě .NET platformy, tedy .NET Framework a .NET Core.

V rámci webového rozhraní uživatel může vytvářet, mazat, upravovat další uživatele, ale musí k tomu být v roli administrátora. Dále uživatel může vytvářet, mazat a upravovat vyšetřování. Tyto akce může provádět jen pokud je administrátorem nebo vlastníkem vyšetřování. K vyšetřování pak má přistup buď administrátor, vlastník nebo vyšetřovatel s přiděleným přístupem k danému vyšetřování.

V konkrétním vyšetřování pak uživatel může přidávat soubory se zachycenou komunikací. Zpracované soubory se pak zobrazí v levé části po rozbalení menu se soubory. Při kliknutí na soubor se zobrazí detailní informace o souboru jako jsou například statistky, konverzace na určitých síťových vrstvách atd. Dále je umožněno spustit export forenzních dat, a to buď ze všech vložených souborů nebo z jednoho konkrétního souboru. Po dokončení exportu si pak uživatel může zobrazit exportovaná data. Dostupné pohledy najde uživatel v menu pro výpis exportů pro jednotlivé protokoly.

Webové rozhraní je modulární, protože je postavené na třívrstvé architektuře a využívá návrhového vzoru Dependency Injection. Tím nám umožňuje jednoduše testovat jednotlivé komponenty a jednoduchou možnost úpravy či přidání nových pohledů pro nové protokoly.

Jednotlivé části implementace byly otestovány. Mezi testovanými částmi jsou správa uživatelů, správa vyšetřování a propojení webového rozhraní s Netfox Framework. U správy uživatelů a vyšetřování se testovalo přidání, úprava a varianty odebrání záznamů. U vyšetřování se pak testovalo navíc zobrazení vyšetřování pro určitého uživatele. Propojení s Netfox Framework bylo otestováno zpracováním konverzací ze vstupního souboru a exportováním forenzních dat z konverzací.

Výhodou webového rozhraní oproti desktopové verzi Netfox Detective je exportování vstupních dat pouze jednou, aby nedocházelo k duplikování extrahovaných dat při znovu spuštění exportování. Pokud se zamyslíme, jakou výhodu přináší webové rozhraní vůči ostatním nástrojům, tak je to určitě možnost spolupráce na vyšetřování. Xplico spolupráci umožňuje, ale jen v rámci skupiny. Do skupiny navíc může uživatele přiřadit pouze administrátor. U webového rozhraní může vlastník vyšetřovaní určit sám, kdo bude mít k vyšetřování přistup. Určitou možnost spolupráce umožňuje i Cloudshark, ale není úplně ideální z pohledu řízení přístupu ke sdílenému souboru. Sdílí se tam pomocí odkazu, na který může přistoupit kdokoliv.

Uživatel vždy ví, kde se zrovna nachází. To dozajista přispívá k větší uživatelské přívětivosti. To, kde se uživatel právě nachází, může zjistit například z titulku stránky nebo spojením nadpisu v levém sloupci, jenž určuje název sekce, v níž se uživatel nachází a nadpisu v obsahové části, který určuje stránku v dané sekci. Dalším prvkem, který se podílí na zvýšení uživatelské přívětivosti jsou rychlé ikony pro rychlou navigaci ve webovém rozhraní a seznam posledně navštívených vyšetřování. Rychlé ikony uživateli umožní dostat se například na stránku, kde bude moci vytvořit vyšetřování jedním kliknutím a nebude se muset proklikávat skrz další pohledy. K tomu účelu jsou vypsaná poslední vyšetřování, protože je velice pravděpodobné, že uživatel bude chtít po přihlášení vstoupit znova do posledně navštíveného vyšetřování.

Webové rozhraní používá HangFire pro spouštění úloh zpracování a extrahování forenzních informací ze zachycené komunikace na pozadí serveru. V budoucnu by se dalo této knihovny využít pro zpracování komunikace více výpočetními uzly. HangFire umožňuje do daného řešení přidat další servery, které provádějí úlohy, jenž jim uživatel přes webové rozhraní zadá.

Literatura

[1] Bootstrap · The most popular HTML, CSS, and JS library in the world. [online], [cit. 2018-01-10].
URL https://getbootstrap.com/

[2] Akinshin, A.: *Getting Started with Knockout.js for .NET Developers*. Community experience distilled, Packt Publishing, 2015, ISBN 9781783984015.

[3] Davidoff, S.; Ham, J.: *Network Forensics: Tracking Hackers Through Cyberspace*. Prentice Hall, 2012, ISBN 9780132564717.

[4] Herceg, T.: Knihovna Riganti Utils Infrastructure 2. [online], [cit. 2019-05-19].
URL https://www.dotnetcast.cz/cs/video/587/Knihovna-Riganti-Utils-Infrastructure-2

[5] Johansen, G.: *Digital Forensics and Incident Response*. Packt Publishing, 2017, ISBN 9781787285392.

[6] Joshi, R.; Pilli, E.: *Fundamentals of Network Forensics: A Research Perspective*. Computer Communications and Networks, Springer London, 2016, ISBN 9781447172994.

[7] Matoušek, P.; Pluskal, J.; Ryšavý, O.; aj.: Advanced Techniques for Reconstruction of Incomplete Network Data. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, ročník 2015, č. 157, 2015: s. 69–84, ISSN 1867-8211.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10864

[8] Microsoft: Implementing the Model-View-ViewModel Pattern. [online], [cit. 2018-01-10].
URL https://msdn.microsoft.com/en-us/library/ff798384.aspx

[9] Microsoft: Introduction. [online], [cit. 2018-01-10].
URL https://www.dotvvm.com/docs/tutorials/introduction/latest

[10] Microsoft: The Repository Pattern. [online], [cit. 2018-05-18].
URL https://msdn.microsoft.com/en-us/library/ff649690.aspx

[11] Nagel, C.: *Professional C# 7 and .NET Core 2.0*. Wiley, 2018, ISBN 9781119449249.

[12] Odinokov, S.: Hangfire Overview. [online], [cit. 2018-01-10].
URL https://www.hangfire.io/overview.html

[13] Pluskal, J.: NetFox.Framework - The network forensic extandable analysis tool. In *Proceedings of the 20th Conference STUDENT EEICT 2014 Volume 2*, Brno University of Technology, 2014, ISBN 978-80-214-4923-7, s. 280–282.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10692

[14] Pluskal, J.: Netfox Detective 2.0 - Nástroj pro síťovou forenzní analýzu. Technická zpráva, 2017.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=11567

[15] Pluskal, J.; Matoušek, P.; Ryšavý, O.; aj.: Netfox Detective: A tool for advanced network forensics analysis. In *Proceedings of Security and Protection of Information (SPI) 2015*, Brno University of Defence, 2015, ISBN 978-80-7231-997-8, s. 147–163.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10863

[16] Ragupathi, M.; De Sanctis, V.; Singleton, J.: *ASP.NET Core: Cloud-ready, Enterprise Web Application Development*. Packt Publishing, 2017, ISBN 9781788293204.

[17] RIGANTI: Authentication and Authorization. [online], [cit. 2018-01-10].
URL https://www.dotvvm.com/docs/tutorials/advanced-authentication-authorization/latest

[18] RIGANTI: ViewModels. [online], [cit. 2018-05-18].
URL https://www.dotvvm.com/docs/tutorials/basics-viewmodels/latest

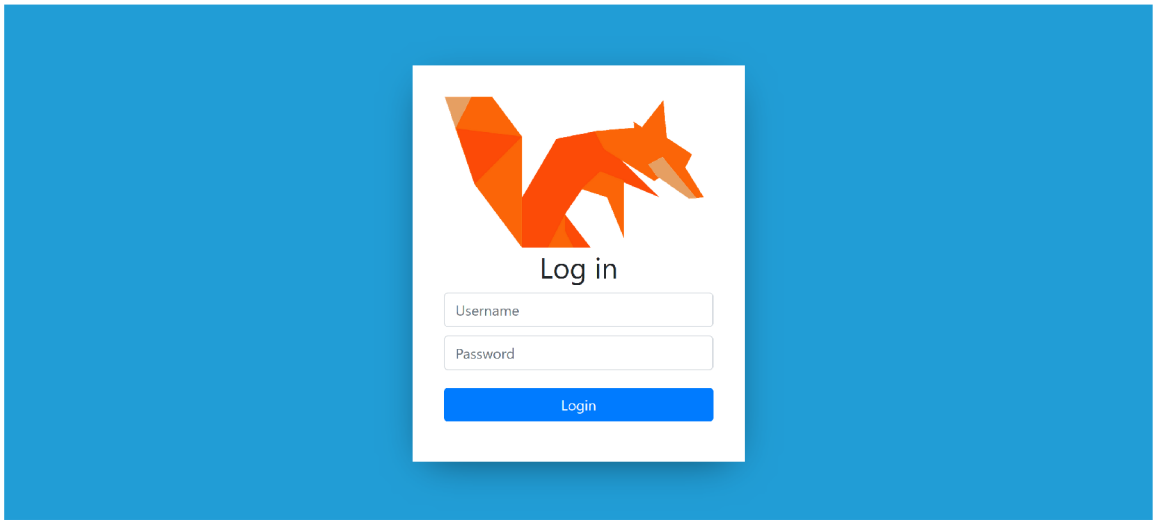
[19] Sanders, C.: *Analýza sítí a řešení problémů v programu Wireshark*. Computer Press, Albatros Media a.s., 2017, ISBN 9788025139790.

[20] Sanders, C.; Smith, J.: *Applied Network Security Monitoring: Collection, Detection, and Analysis*. Elsevier Science, 2013, ISBN 9780124172166.

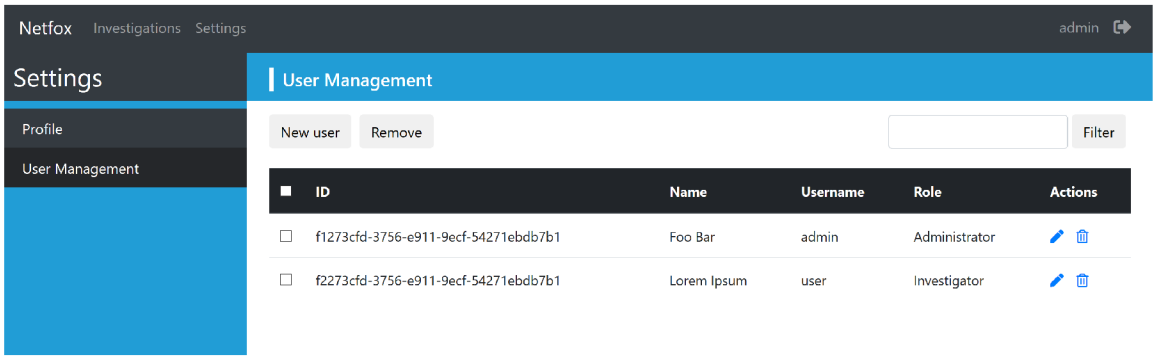
[21] Stoneman, E.: *Docker on Windows: From 101 to production with Docker on Windows, 2nd Edition*. Packt Publishing, 2019, ISBN 9781789610604.

Příloha A

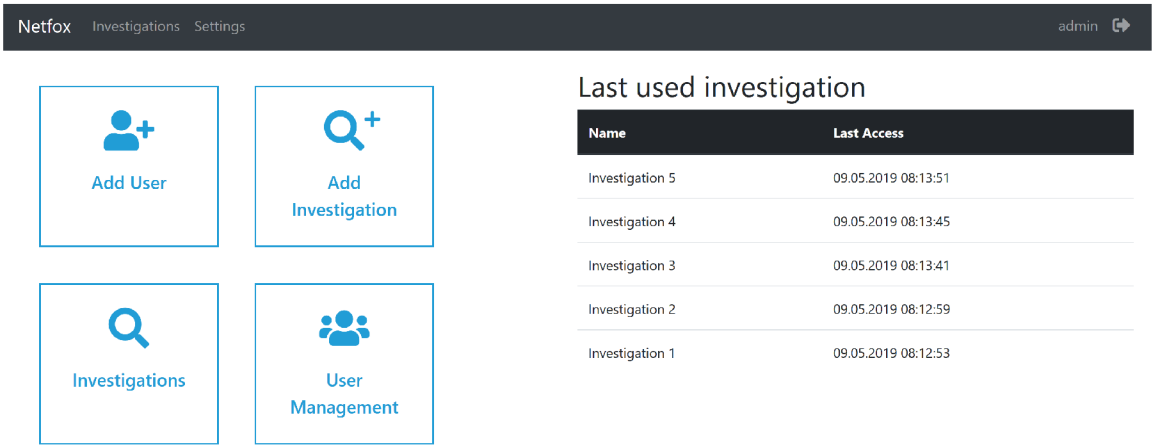
Snímky webového rozhraní



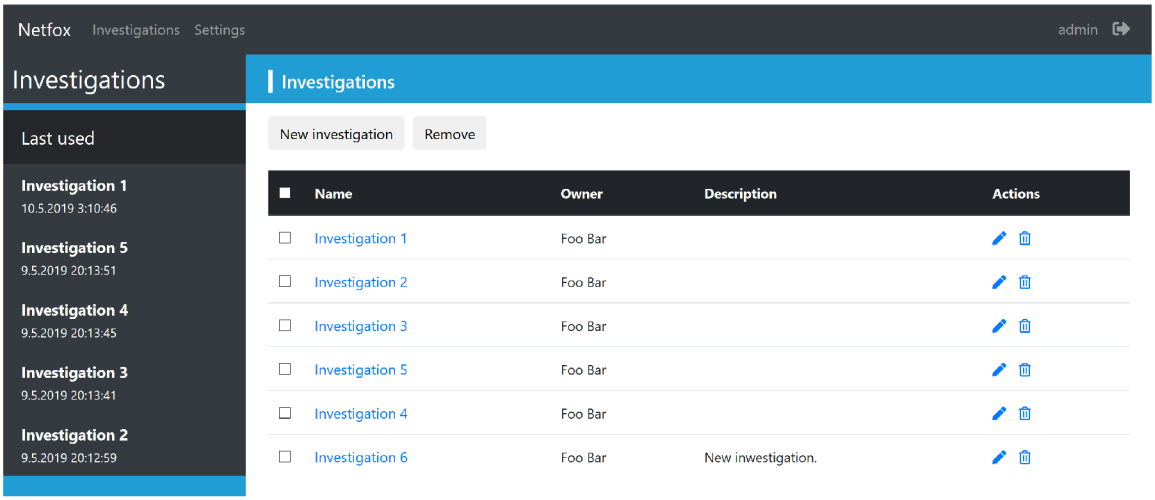
Obrázek A.1: Snímek stránky s přihlašovacím formulářem.



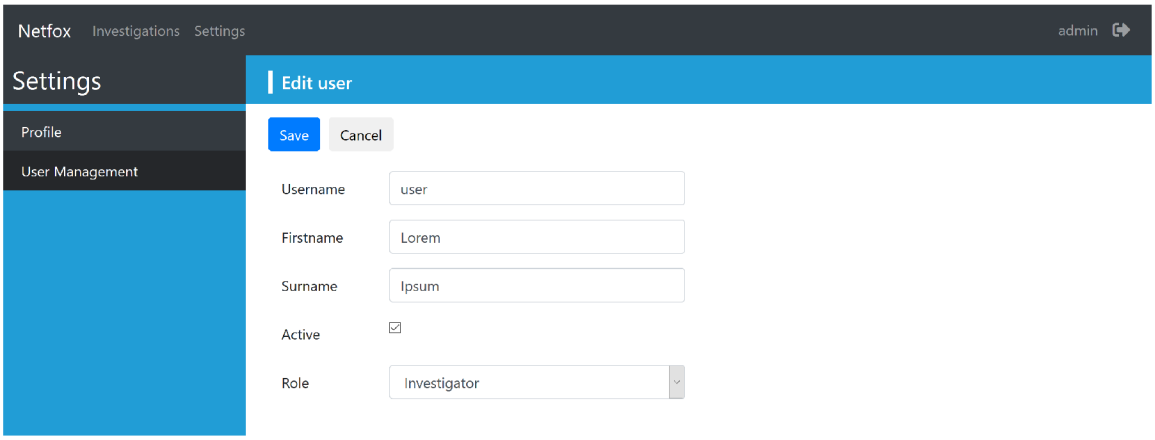
Obrázek A.2: Snímek stránky s přehledem uživatelů.



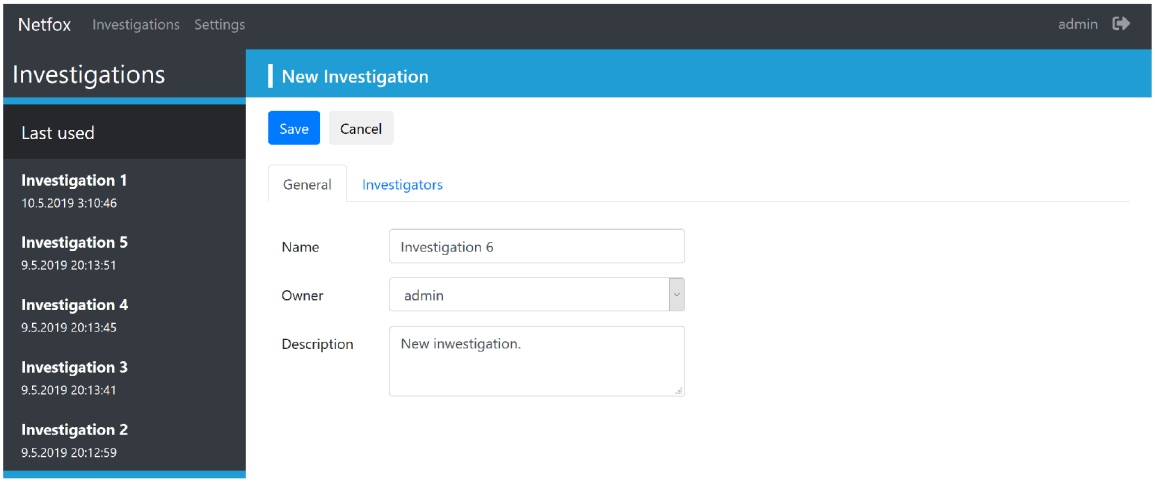
Obrázek A.3: Snímek úvodní stránky.



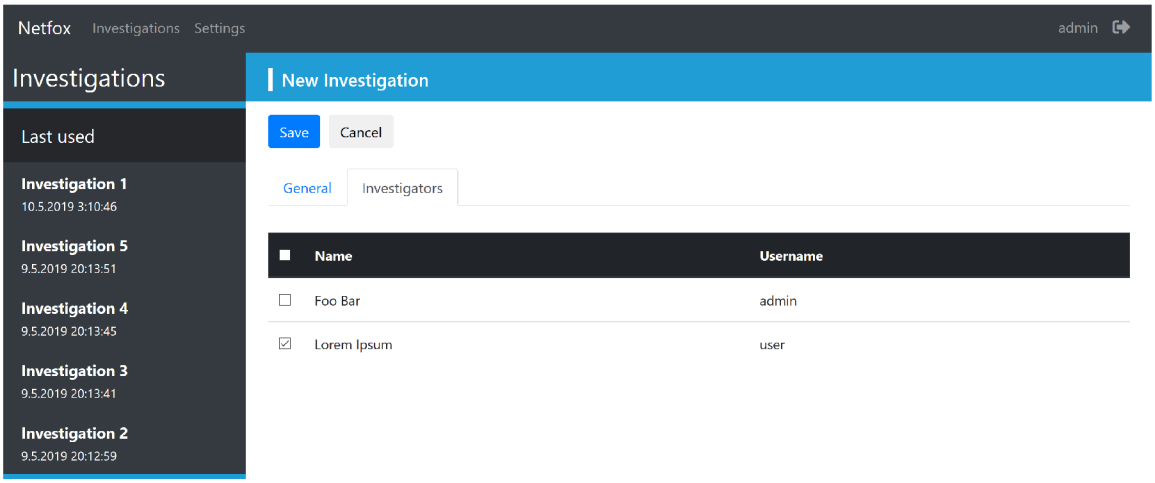
Obrázek A.4: Snímek stránky se správou vyšetřování.



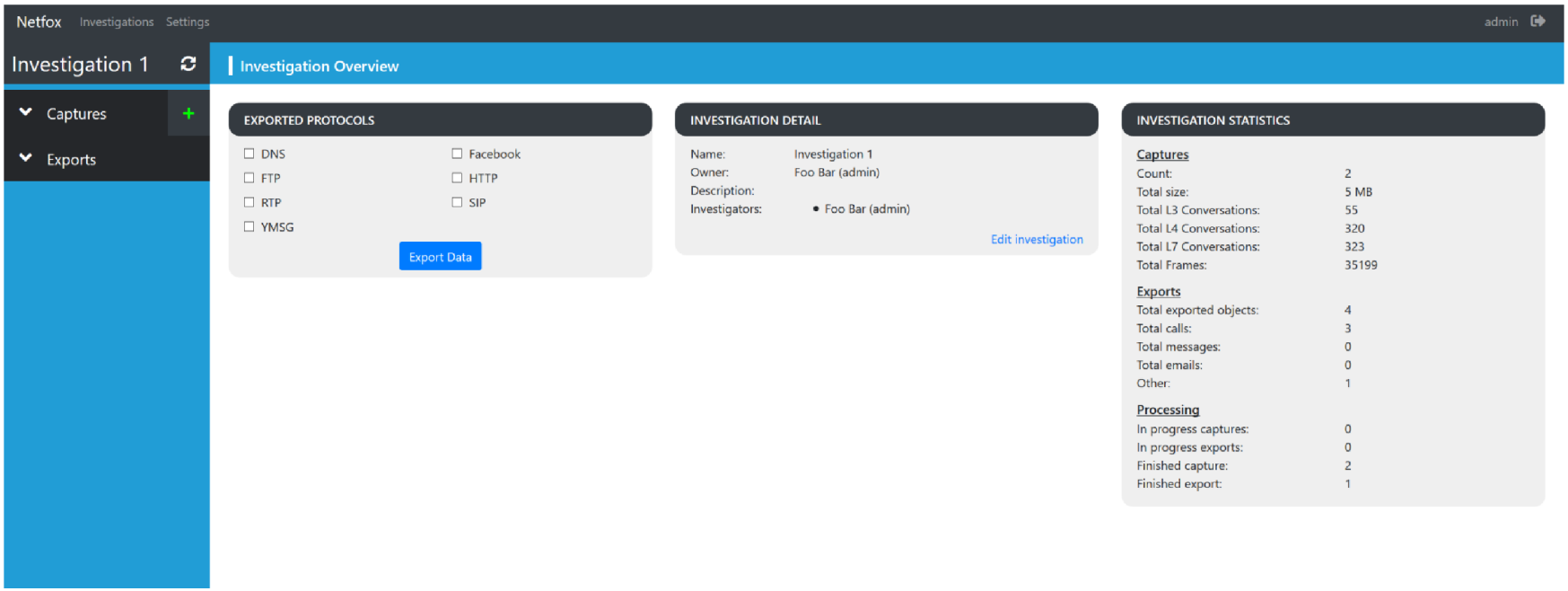
Obrázek A.5: Snímek stránky pro přidání nebo úpravu uživatele.



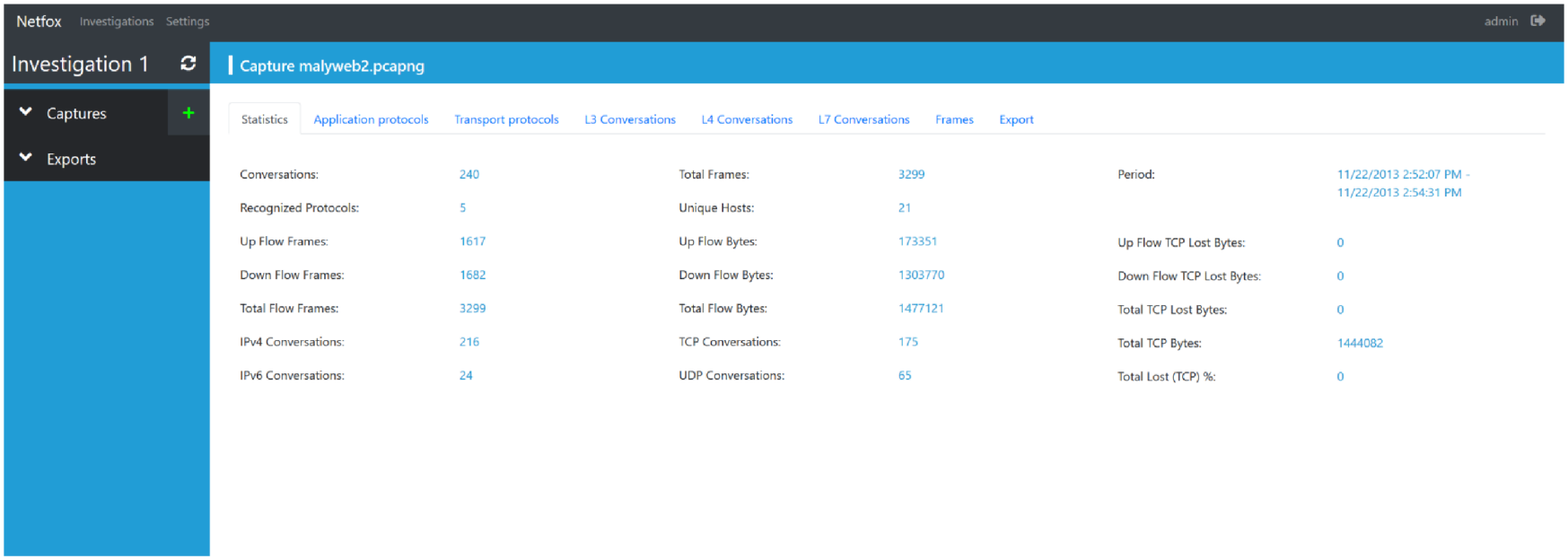
Obrázek A.6: Snímek stránky pro přidání nebo úpravu vyšetřování.



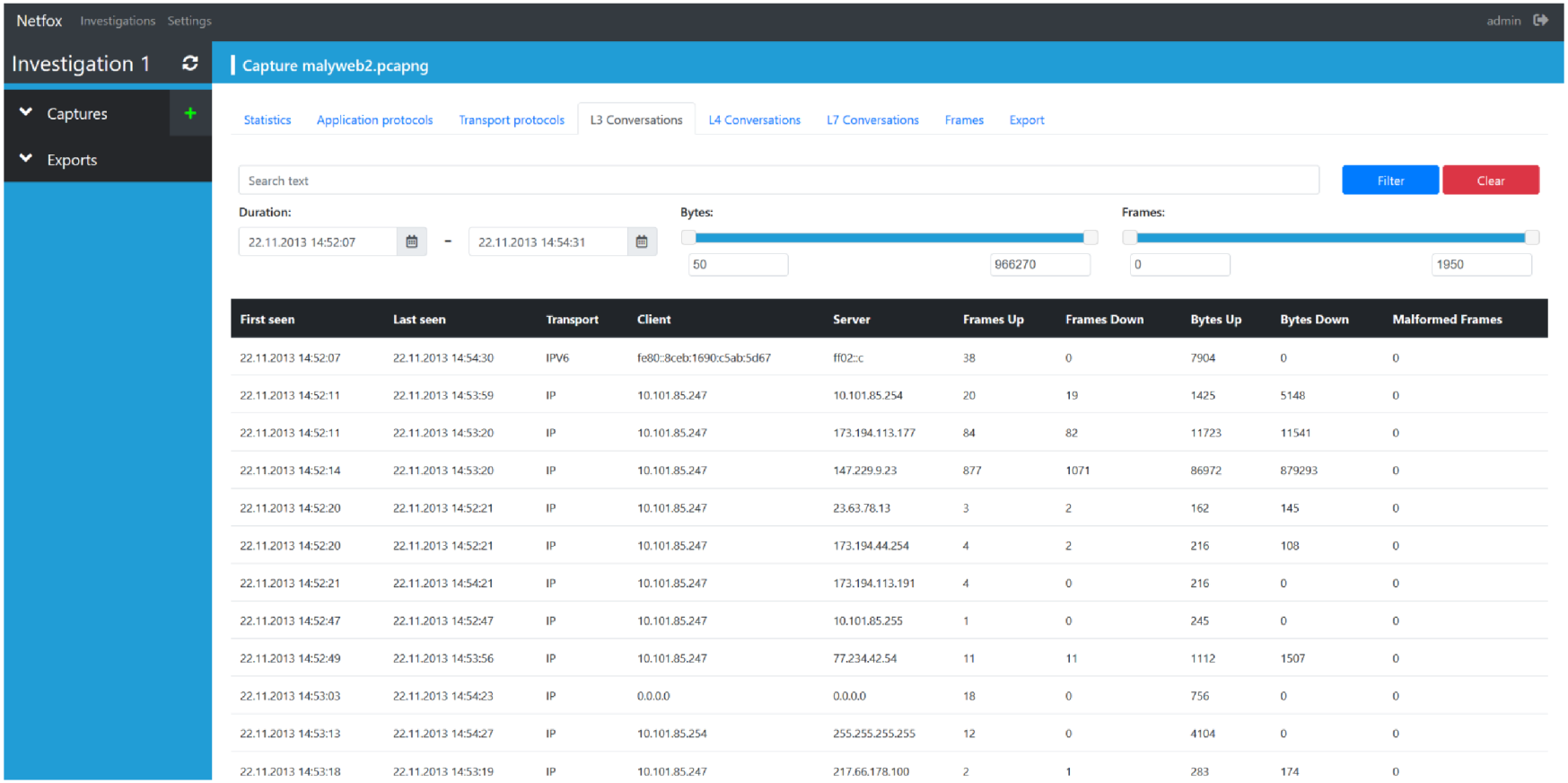
Obrázek A.7: Snímek záložky pro přidělení přístupu uživatelů na stránce pro přidání nebo úpravu vyšetřování.



Obrázek A.8: Snímek úvodní stránky vyšetřování se statistikami.

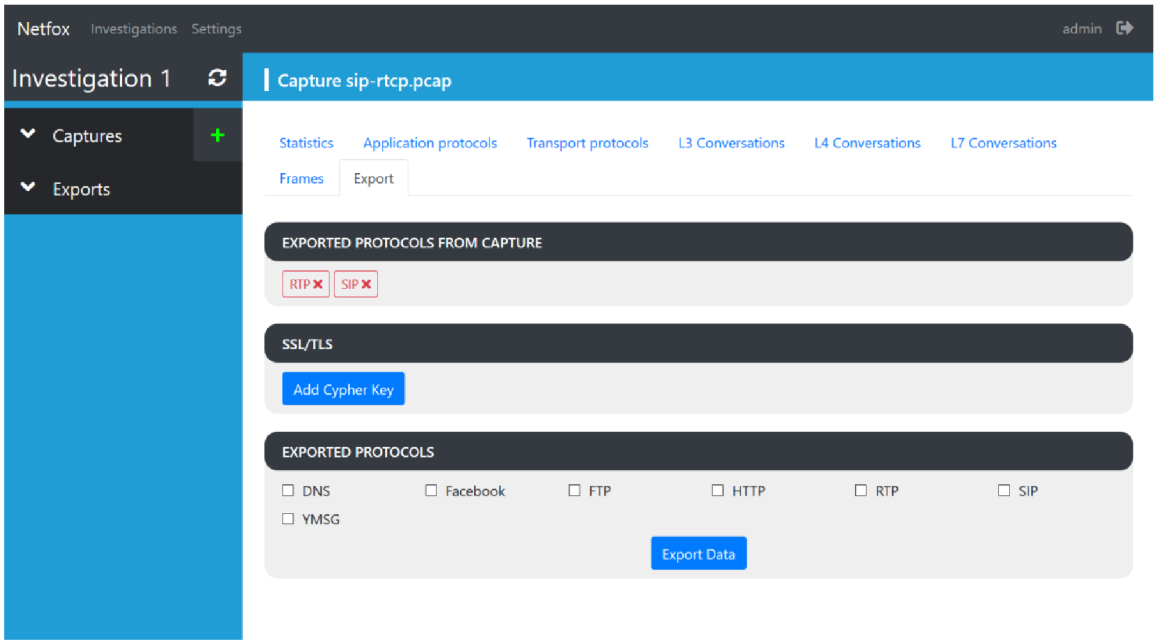


Obrázek A.9: Snímek záložky se statistikami o souboru se zachycenou komunikací.



First seen	Last seen	Transport	Client	Server	Frames Up	Frames Down	Bytes Up	Bytes Down	Malformed Frames
22.11.2013 14:52:07	22.11.2013 14:54:30	IPV6	fe80::8ceb:1690:c5ab:5d67	ff02::c	38	0	7904	0	0
22.11.2013 14:52:11	22.11.2013 14:53:59	IP	10.101.85.247	10.101.85.254	20	19	1425	5148	0
22.11.2013 14:52:11	22.11.2013 14:53:20	IP	10.101.85.247	173.194.113.177	84	82	11723	11541	0
22.11.2013 14:52:14	22.11.2013 14:53:20	IP	10.101.85.247	147.229.9.23	877	1071	86972	879293	0
22.11.2013 14:52:20	22.11.2013 14:52:21	IP	10.101.85.247	23.63.78.13	3	2	162	145	0
22.11.2013 14:52:20	22.11.2013 14:52:21	IP	10.101.85.247	173.194.44.254	4	2	216	108	0
22.11.2013 14:52:21	22.11.2013 14:54:21	IP	10.101.85.247	173.194.113.191	4	0	216	0	0
22.11.2013 14:52:47	22.11.2013 14:52:47	IP	10.101.85.247	10.101.85.255	1	0	245	0	0
22.11.2013 14:52:49	22.11.2013 14:53:56	IP	10.101.85.247	77.234.42.54	11	11	1112	1507	0
22.11.2013 14:53:03	22.11.2013 14:54:23	IP	0.0.0.0	0.0.0.0	18	0	756	0	0
22.11.2013 14:53:13	22.11.2013 14:54:27	IP	10.101.85.254	255.255.255.255	12	0	4104	0	0
22.11.2013 14:53:18	22.11.2013 14:53:19	IP	10.101.85.247	217.66.178.100	2	1	283	174	0

Obrázek A.10: Snímek záložky s L3 konverzacemi.



Obrázek A.11: Snímek záložky Export.

Příloha B

Obsah DVD

DVD obsahuje následující soubory a složky:

- **doc** - složka se zdrojovými soubory textu diplomové práce pro systém LaTeX
- **src** - složka se zdrojovými soubory implementace
- **DP.pdf** - text diplomové práce