



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE FORENZNÍCH SÍŤOVÝCH DAT
NETWORK FORENSIC DATA VISUALIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MAREK KLOFERA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN PLUSKAL

BRNO 2023

Zadání bakalářské práce



146282

Ústav: Ústav informačních systémů (UIFS)
Student: **Klofera Marek**
Program: Informační technologie
Specializace: Informační technologie
Název: **Vizualizace forenzních síťových dat**
Kategorie: Počítačové sítě
Akademický rok: 2022/23

Zadání:

1. Nastudujte alespoň dva vhodné aktuální systémy pro síťové forenzní vyšetřování implementované ideálně pro webové prostředí, případně desktopové aplikace. Zaměřte se na intuitivnost grafického uživatelského rozhraní a dle získané zkušenosti vytvořte funkční i nefunkční požadavky na obdobný systém, které budou reflektovat vámi navržená vylepšení slabých stran analyzovaných nástrojů.
2. Vytvořte alespoň dva základní a alespoň dva specifické případy užití, které jsou běžné a demonstrují použití nalezených, vhodných nástrojů.
3. Navrhněte systém respektující požadavky z bodu 1 schopný vyřešit případy užití z bodu 2. Systém bude obsahovat předpřipravená, případně generovaná data dle vašeho uvážení a konzultace s vedoucím.
4. Implementujte systém dle návrhu z bodu 3. Systém zveřejněte pod open source licencí.
5. Otestujte implementovaný systém. Zhodnotte použitelnost systému k řešení případů užití z bodu 2.

Literatura:

1. GEBHARDT, Tobias a Hans P. REISER. *Network Forensics for Cloud Computing*. Berlin, Heidelberg: Springer, 2013. Lecture Notes in Computer Science. ISBN 978-3-642-38541-4. DOI: 10.1007/978-3-642-38541-4_3
2. ZEMBJAKOVÁ, Martina. *Network Forensics Tools Survey and Taxonomy*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. 2021-06-22. Vedoucí práce Pluskal Jan.

Při obhajobě semestrální části projektu je požadováno:

- body 1, 2 a 3.

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

Vedoucí práce: **Pluskal Jan, Ing.**

Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.

Datum zadání: 1.11.2022

Termín pro odevzdání: 10.5.2023

Datum schválení: 24.10.2022

Abstrakt

Cílem této práce je vytvořit přehlednou a uživatelsky přívětivou webovou aplikaci, která bude sloužit jako vizualizační nástroj pro forenzně sítová data a ulehčit tak vyšetřovatelům a dalším odborníkům v oblasti informační bezpečnosti při analýze zachycené komunikace. Díky rozsáhlým možnostem filtrování dat, může uživatel zobrazit pouze relevantní data a ušetřit tak čas při analýze. Dále aplikace nabízí možnosti agregace dat v podobě grafů nad daty, upozornění na podezřelá slova obsažena v těle jednotlivých zachycených zpráv a nebo agregování dat dle identifikátoru uživatele.

Abstract

The aim of this work is to create a clear and user-friendly web application that will serve as a visualization tool for forensic network data, making it easier for investigators, administrators, and other experts in the field of information security to analyze captured communication. Thanks to extensive data filtering options, the user can display only relevant data and save time during analysis. Furthermore, the application offers options for data aggregation in the form of graphs, alerts for suspicious words contained in the body of individual captured messages and data aggregation by user identifier.

Klíčová slova

webová aplikace, blazor, síťová forenzní analýza, vizualizace, webassembly

Keywords

web application, blazor, network forensics, visualization, webassembly

Citace

KLOFERA, Marek. *Vizualizace forenzních sítových dat*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal

Vizualizace forenzních síťových dat

Prohlášení

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

.....
Marek Klofera
9. května 2023

Poděkování

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

Obsah

1	Úvod	3
2	Síťová forenzní analýza	4
2.1	Definice	4
2.2	Síťový důkazní materiál	5
2.2.1	Typy získávání důkazního materiálu	5
2.2.2	Zdroje získávání důkazního materiálu	7
2.3	Nástroje k získávání forenzních síťových dat	11
2.3.1	Síťové forenzní nástroje pro analyzování	11
2.3.2	Síťové programy pro skenování	12
2.3.3	Síťové nástroje pro monitorování	13
2.3.4	Nástroje pro posouzení zranitelnosti systému	14
2.4	Analýza nástrojů Xplico a NetworkMiner	15
2.4.1	Xplico	15
2.4.2	NetworkMiner	18
2.4.3	Funkční a nefunkční požadavky na nový systém	19
3	Návrh a implementace	22
3.1	Použité technologie	22
3.1.1	Jednostránková aplikace	23
3.1.2	WebAssembly	23
3.1.3	Blazor	23
3.1.4	Použité knihovny	25
3.2	Návrh definice aplikačního rozhraní	26
3.2.1	Struktura aplikačního rozhraní	26
3.2.2	Objekty pro datový přenos	27
3.3	Návrh uživatelského prostředí	27
3.3.1	Drátové modely	28
3.3.2	Grafický návrh	30
3.4	Tvorba webové aplikace	31
3.4.1	Serverová část	32
3.4.2	Klientská část	33
4	Testování	37
4.1	Testování funkcionality	37
4.2	Testování použitelnosti	38
4.3	Testování aplikace	38
4.3.1	Specifický případ užití	39
4.3.2	Analýza úkolů	39
4.3.3	Případ užití aplikace	41

5 Závěr	42
Literatura	45

Kapitola 1

Úvod

Díky nárustu digitálních zařízení se zvyšuje i síťový provoz, což vyšetřovatelům výrazně ztěžuje extrahování relevantních informací k identifikaci potencionálních hrozeb, šetření spáchaných kybernetických útoků nebo činům, které porušují zákon. Vizualizační nástroje umožňují vyšetřovatelům zobrazovat relevantní data v přehledném a srozumitelném formátu čímž lze jednodušeji a rychleji zobrazit hledaná data a zabránit útoku nebo rychleji identifikovat útočníka. Vývojem těchto nástrojů tak lze bojovat proti kybernetickým útočníkům a jiným pachatelům, kteří konají proti zákonům.

Cílem této práce je prozkoumat již existující nástroje vizualizující síťovou komunikaci a na základě získaných poznatků navrhnout nový nástroj, v podobě webové aplikace, která by měla přínos v oblasti vizualizace forenzně síťových dat. Webová aplikace obsahuje nástroje pro filtrování a třídění dat, které umožňují vyšetřovatelům se lépe orientovat v zobrazených informacích a rychleji analyzovat zachycená data. Dalším přínosem aplikace je možnost zobrazení agregovaných dat nad kolekcí zachycených zpráv. Tato agregovaná data jsou zobrazena v podobě grafů. Jelikož je nástroj pod licencí open-source, tak byl navržen tak, aby byl jednoduše modifikovatelný a rozšířitelný do budoucna. Cílovou skupinou ovšem nemusí být pouze vyšetřovatelé, ale například i administrátoři sítí.

Úvod práce je věnován forenzní síťové analýze jako takové. Je zde představena definice forenzní síťové analýzy, následuje část o síťových důkazních materiálech, popisující typy získávání síťového důkazního materiálu (zachycování veškeré komunikace, relační data, „*alert data*“ a statistická data) a zdroje, ze kterých lze tato data získat (drátová/bezdrátová komunikace, přepínače, směrovače DHCP, DNS, autentizační a proxy servery). Další část je zaměřena na nástroje, které vyšetřovatelé běžně používají při práci s bojem proti kyberzločincům, ale i administrátoři při správě sítí. Tyto nástroje se lze rozdělit dle jejich využití a to na síťové forenzní nástroje pro analyzování, skenování, monitorování a posouzení zranitelnosti.

Následuje část zabývající se analýzou a porovnáním dvou vybraných nástrojů pro síťovou forenzní analýzu, konkrétně nástroj Xplico, který je implementován jako webová aplikace a desktopový nástroj NetworkMiner. U obou nástrojů je graficky zobrazen případ užití, demonstrující celkovou funkcionalitu nástroje a specifický případ užití, který zobrazuje použití nástroje při řešení konkrétního problému. Další kapitola je věnována návrhu a implementace, popisující návrh aplikačního rozhraní, objektů pro datový přenos, drátových a grafických modelů. Konec této kapitoly je věnován implementaci. Ta je rozdělena na klientskou a serverovou část. Poslední kapitola této práce je věnována testování aplikace, ve které je graficky vizualizován případ užití demonstrující celkovou funkcionalitu aplikace.

Kapitola 2

Síťová forenzní analýza

Síťová forenzní analýza hraje nezbytnou roli při vyšetřování kyberzločinů. Důkazní materiál, který mohou vyšetřovatelé získat a následně analyzovat může mít více forem. V této práci zmíním sběr celého datové toku (kompletní kopie všech přenášených dat), relační data (informace ohledně relacích mezi jednotlivými zařízeními), upozornění (vyvolané upozornění na předem definované události) a statistická data (agregovaná netflow data).

Vyšetřovatelé mohou důkazní materiál získat vícero způsoby. V této práci budu popisovat získání dat z drátové/bezdrátové komunikace, ze záznamů serverů a síťových stanic, směrovačů, přepínačů, DHCP serverů, DNS serverů a proxy serverů.

Následně se zmíním o nástrojích, které se běžně používají v síťové forenzní analýze a uvedu jejich využití. Pro některé z nich jsem vytvořil případy užití. Většina těchto nástrojů má open-source licenci, takže je možné si tyto nástroje vyzkoušet, případně modifikovat dle vlastních potřeb.

2.1 Definice

Síťová forenzní analýza je jedním z odvětví forenzní vědy, zaměřující se na počítačové sítě [14]. Zabývá se daty síťového provozu získanými aktivními nebo pasivními síťovými zařízeními. Pasivní síťová zařízení, jako například síťové odposlechy, umožňují získávání síťových dat bez rušení provozu v síti. Zdroje síťových důkazů jsou popsány v kapitole 2.2.2 [2].

Jednotlivá odvětví digitální forenzní analýzy (počítačová, mobilní, ...) spolupracují na získávání důkazních materiálů. Síťová forenzní analýza může být použita k identifikaci a předcházení útoků na síťová zařízení, jako jsou například přepínače a směrovače. Je to věda zabývající se zachycováním, zaznamenáváním síťových dat a jejich analyzováním za účelem detekce narušení stavu či jiných aktivit odpovídajícím zákonům a jejich následnému šetření [16]. Analyzovaná data poté slouží jako důkazní materiál pro následné usvědčení osob, které provozovaly nelegální aktivity, nebo je stále provozují. Síťová forenzní analýza může být také využita k identifikaci uživatelských aktivit a monitorování používání síťových aplikací, což může být užitečné pro organizace při plánování a řízení svých zdrojů. S rozvojem technologií a nových typů sítí, jako jsou například IoT sítě, je síťová forenzní analýza stále důležitější pro ochranu sítí a identifikaci hrozob.

Síťová forenzní analýza může být složitá a vyžaduje speciální znalosti a dovednosti, včetně schopnosti rozpoznat podezřelou síťovou aktivitu a analyzovat velké objemy dat. Síťová forenzní analýza také hraje klíčovou roli při vyšetřování kybernetických útoků a pomáhá úřadům a bezpečnostním službám identifikovat a usvědčit pachatele. Hlavním cílem forenzní analýzy je analyzovat síťový důkazní materiál a rozhodnout, zda došlo k nelegální aktivitě či nikoliv. Analýza síťového provozu může být také využita k identifikaci hrozob a útoků na firemní sítě, což pomáhá organizacím předcházet ztrátám dat a finančním ztrátám. Pokud k nelegální aktivitě

došlo, celý incident je vyšetřen a následně zdokumentován. Síťovou forenzní analýzu lze rozšířit o SDN, mobilní síť, „*cloud computing*“, IoT a jiné [2].

2.2 Síťový důkazní materiál

V této sekci se budu zabývat síťovým důkazním materiálem, dále se zmíním jaké existují síťové důkazní materiály a zdroje k jeho získávání. Jedná se o data, která mohou být použita například k identifikaci útočníka v případě síťového útoku nebo jiné nelegální aktivity. Cílem je získat, co nejvíce informací o útoku a útočníkovi, aby byl celý případ řádně vyšetřen a útočník identifikován a rádně potrestán.

Typy získávání důkazního materiálu se mohou lišit v závislosti na prostředí a případu. Mezi nejčastěji používané metody patří zachycování veškeré komunikace, sběr relačních dat, „*alert dat*“ a statistických dat. Tyto metody mohou být kombinovány k dosažení nejlepších výsledků a k případné identifikaci a usvědčení podezřelých osob.

Existuje několik zdrojů, ze kterých mohou být důkazní materiály získány. Tyto zdroje zahrnují drátovou komunikaci, bezdrátovou komunikaci, přepínače, směrovače, DHCP servery, DNS servery, autentizační servery a webové proxy servery a další. Forenzní vyšetřovatelé využívají různé techniky a nástroje pro získání dat z těchto zdrojů a následnou analýzu získaných dat za účelem identifikace podezřelých aktivit a osob.

2.2.1 Typy získávání důkazního materiálu

Síťový důkazní materiál hraje klíčovou roli ve forenzním vyšetřování týkajícím se síťových útoků nebo jiných nelegálních aktivit v oblasti počítačových sítí. Jsou různé typy síťových důkazních materiálů, ovšem vyšetřovatelé se zpravidla snaží získat všechny a na základě prostředí a případu jim přidělí prioritu, dle které je následně zpracovávají.

V této sekci popíšu čtyři všeobecně známé typy síťového důkazního materiálu, které jsou často zkoumány v rámci forenzního vyšetřování. Jedním z nejdůležitějších typů síťového důkazního materiálu jsou zachycené komunikace. Jedná se o zaznamenání veškeré síťové komunikace včetně elektronické pošty, VoIP hovorů, souborů, atd. Dalším důležitým typem síťového důkazního materiálu jsou relační data. Tato data obsahují informace o všech spojeních mezi síťovými prvky (přepínače, směrovače, ...), včetně informací o zdrojové a cílové adrese nebo použitém protokolu. „*alert data*“ jsou dalším důležitým typem síťového důkazního materiálu. Tato data jsou generována systémy a vytváří upozornění na předem definované situace. Posledním typem síťového důkazního materiálu, který v této sekci zmíním jsou statistická data. Tato data poskytují celkový přehled o síťové aktivitě. Nástroje k jejich získání jsou popsány v kapitole 2.3.1.

Zachycování veškeré komunikace

Zachycování veškeré komunikace neboli „*full content data*“ je proces zaznamenávání a analýzy všech paketů, které putují po síti. Tento proces je dosažen pomocí speciálních nástrojů, zaznamenávajících komunikaci na daném rozhraní a duplikujících procházející pakety, tak aby nenarušili síťový provoz.

Zachycená data obsahují veškerý provoz, včetně hlasu a videa což z nich činí cenný zdroj informací pro analýzu výkonu sítě, identifikaci potenciálních bezpečnostních incidentů a vyšetřování provedených útoků. Tato data bývají většinou šifrována a pro jejich analýzu je nutné data dešifrovat. Proces zachycování dat může být v některých případech problematický z hlediska ochrany soukromí a je důležité dodržovat příslušné právní předpisy, jako například GDPR. Z bezpečnostního hlediska by bylo vhodné ukládat veškerý síťový provoz, avšak velikost těchto dat může být problematická při jejich zaznamenávání na datové médium. Vzhledem k rostoucímu množství síťového provozu a počtu sítí je stále důležitější zajistit, aby síťová forenzní analýza

byla prováděna efektivně a účinně. Z tohoto důvodu se využívá vzorkování, kdy se zachycuje pouze určité procento z celkového provozu [17].

Zachycená data bývají často ukládána ve formátu z rodiny *.pcap*¹ (libpcap, pcap-ng, ...). Výhodou těchto formátů je, že jej většina analyzátorů a dalších sítových nástrojů podporují. Nevhodou *.pcap* formátů je, že postrádají integritní zabezpečení a šifrování, avšak soubor lze zašifrovat a integritně zabezpečit programy jako gpg nebo openssl [14].

Ukázku tzv. „*full content*“ dat můžete vidět ve výpisu 2.1. Analyzování velkých objemů dat může být časově náročné a vyžaduje speciální nástroje a dovednosti, jako jsou například analyzátoři sítového provozu a softwary pro vizualizaci dat. V poslední době se také objevují nové technologie a nástroje z oblasti strojového učení a umělé inteligence, které mohou být využity k zlepšení sítové forenzní analýzy a identifikaci nových hrozeb.

```
No. Time Source Destination Protocol Length Info
1 0.000000 192.168.0.136 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
```

```
Frame 1: 217 bytes on wire (1736 bits), 217 bytes captured (1736 bits)
Encapsulation type: Ethernet (1)
Arrival Time: Jan 10, 2023 10:09:40.348669000 Central Europe Standard Time
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1673341780.348669000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 217 bytes (1736 bits)
Capture Length: 217 bytes (1736 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:udp:ssdp]
[Coloring Rule Name: UDP]
[Coloring Rule String: udp]
```

Výpis 2.1: Zkrácená ukázka zachyceného paketu pomocí jednoho z nejrozšířenějších nástrojů Wireshark.

Relační data

Relační data neboli „*session data*“ jsou jedním z nejužitečnějších zdrojů informací pro sítovou forenzní analýzu, jelikož poskytují podrobné informace o jednotlivých spojeních a komunikaci mezi zařízeními. Na rozdíl od zachycování veškerého provozu se zachycují informace pouze o jednotlivých spojení a obvykle se zaznamenává komunikace mezi dvěma zařízeními. Demonstraci zobrazuje výpis 2.2, na kterém lze vidět počátek a konec spojení, zdrojová a cílová IP adresa a číslo portu, použitý protokol a množství přenesených dat.

Vyšetřovatelé mohou pomocí těchto dat zjistit, která zařízení byla do události zapojena a jakým způsobem. Tyto informace mohou být také velice užitečné k detekování podezřelé aktivity, například, že zaměstnanec odesílá velké množství dat což může znamenat únik informací nebo vysoký počet dotazů z určitého zařízení, což může signalizovat útok [14]. Relační data mají značně menší velikost než zachycování veškeré komunikace, přesto však mohou být stále velice užitečná, například při detekci nelegální aktivity nebo rekonstrukci konkrétních událostí.

```
Start time: 2019-08-01 09:12:32
End time: 2019-08-01 09:13:32
Source IP: 192.168.1.100
Destination IP: 192.168.1.200
```

¹pcap - packet capturing - zachycování paketů

```
Source port: 12345
Destination port: 80
Protocol: TCP
Bytes sent: 1234
Bytes received: 5678
```

Výpis 2.2: Příklad relačních dat

Upozorňovací data

Upozorňovací data neboli „*alert data*“ jsou data generována systémy NIDS a jsou klíčová k rychlému detekování podezřelé aktivity v síti. Systémy NIDS jsou nezbytným nástrojem pro ochranu sítě a poskytují rychlou a spolehlivou detekci možných hrozob. Systém je nastaven, aby monitoroval specifický stav nebo více stavů na síti, například správcem sítě. Pokud na síti dojde k definovanému stavu systém vytvoří upozornění, které následně odešle zodpovědné osobě, typicky správci sítě. Tato osoba celý incident prověří a vyhodnotí situaci, zda došlo k porušení pravidel a incident byl vyvolán oprávněně nebo zda došlo k planému upozornění a žádné hrozby systému nehrozí [14].

Při správném nastavení mohou systémy NIDS detektovat a reagovat na nebezpečné situace automaticky, bez nutnosti zásahu síťového administrátora. Typickým příkladem může být například DNS tuneling. Systém monitoruje síť a ve chvíli kdy zaznamená častý dotaz na doménu, která není obecně známá (google.com, facebook.com, cnn.com, ...) ve spojení s velikostí dotazu/odpověď vyvolá upozornění na podezřelou aktivitu. Systémy NIDS jsou schopny detektovat různé druhy hrozob, jako jsou například útoky typu DDoS, SQL injection, phishing a mnoho dalších [17]. Upozornění generovaná systémy NIDS jsou důležitým prvkem při vyšetřování incidentů a mohou poskytnout cenné informace o původu útoku a jeho charakteristikách.

Statistická data

Statistická data jsou důležitým zdrojem informací pro síťovou forenzní analýzu, jelikož poskytují detailní pohled na různé aspekty sítě a umožňují tak rychlé odhalení anomalií. Tato data obsahují informace jako například začátek a konec síťové komunikace, počet použitých protokolů, průměrnou velikost paketů, nejvíce aktivní uzly, a další [14]. Pomocí statistických dat mohou být detekovány různé druhy anomalií, jako jsou například síťové útoky, chyby v konfiguraci sítě nebo neobvyklé chování uživatelů.

2.2.2 Zdroje získávání důkazního materiálu

Existuje řada zdrojů, které mohou poskytnout síťový důkazní materiál. Každé prostředí je jiné a tak zdroj nalezení těchto informací se může vždy lišit. Vyšetřovatelé se většinou snaží získat důkazy z co nejvíce zdrojů a získat tak co nejvíce informací o útoku a útočníkovi. To jaká zařízení budou vyšetřovatelé zkoumat a jaká data budou hledat záleží na prostředí a daném případu. Nalezené informace se mohou z různých zdrojů překrývat čímž lze určit korelace těchto dat.

V této části, popíšu některé z nich a zároveň jakou mají hodnotu pro forenzního vyšetřovatele.

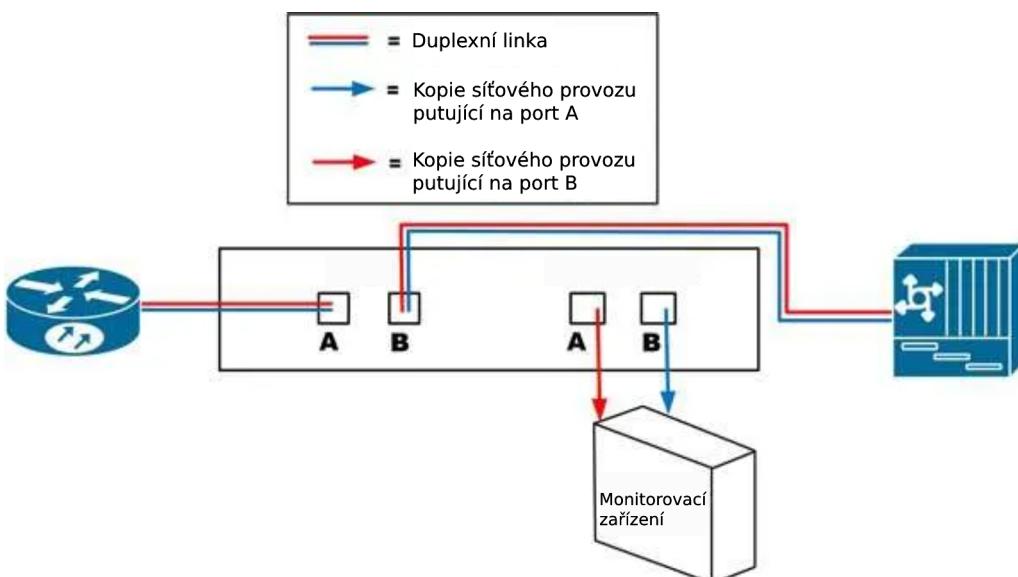
Jedním z hlavních zdrojů je drátová komunikace. Ta zahrnuje veškerou komunikaci, která probíhá přes kabelové připojení, jakým je například ethernet nebo optické kably. Dalším cenním zdrojem je bezdrátová komunikace, kterou zastává například Wi-Fi nebo Bluetooth. Dále se zmíním o získávání důkazního materiálu z přepínačů a směrovačů, jenž mohou poskytnout informace o připojených zařízeních, použitých protokolech, atd. Poté se budu věnovat DNS, DHCP servery a následně autentizačními servery, které slouží k ověření identity uživatele a poskytují informace o přihlašování. Posledním zdrojem důkazního materiálu, kterým se v této práci budu zabývat, jsou proxy servery, které zmíním na konci této sekce.

Drátová komunikace

Drátová komunikace se používá k propojení zařízení v různých typech sítí. Propojují se směrovače, routery, počítače a další zařízení. Přenos dat probíhá na základě změny napětí (kroucená dvojlinka, koaxiální kabel) nebo v přítomnosti/absenci fotonů (optické vlákno). Obě technologie pracují s digitální signalizací [18]. Následuje popis některých typů zařízení, se kterými lze odposlouchávat drátové komunikační médium. Při jakémkoliv manipulaci s drátovým médiem lze poškodit jeho komunikační vlastnosti.

- **Vnitřní síťové kohouty** — neboli „*inline network tap*“ je hardwarové zařízení, které slouží k přímému odposlechu síťového provozu. Tento typ zařízení se často používá v rámci síťového monitorování nebo forenzního vyšetřování, aby bylo možné získat podrobné informace o síťovém provozu a identifikovat případné problémy nebo nelegální aktivity. Jedná se o zařízení, které obsahuje nejčastěji 4 porty, které slouží k připojení síťových zařízení a ke snímání síťového provozu.

Řekněme, že máme dva směrovače na přímo propojeny koaxiálním kabelem. Abychom mohli vložit odposlouchávací zařízení mezi tyto směrovače, je nutné přerušit spojení a přetnout komunikační médium. Následně se na oba konce média vloží konektor (typicky RJ-45²) a zapojí se do portů odposlouchávacího zařízení. Nyní je komunikace opět navázána a směrovače mohou opět komunikovat. Nicméně veškerá komunikace nyní prochází přes odposlouchávací zařízení, které duplikuje signál do zbývajících portů, které lze zaznamenávat například připojeným počítačem. Grafické znázornění takového případu, můžete vidět na obrázku 2.1 [13].



Obrázek 2.1: Demonstrace použití odposlouchávacího zařízení „*inline network tap*“ na drátovém komunikačním médiu mezi dvěma koncovými zařízeními.

- **„Upíří kohouty“** — neboli „*Vampire taps*“ jsou zařízení, které na rozdíl od předešlého zařízení nevyžadují přerušení spojení a přetnutí komunikačního média. Reálné zařízení vizualizuje obrázek 2.2. Toto zařízení obsahuje ostré kovové cvočky, které se skrz ochranou vrstvu a vrstvu stínění dostanou přímo na komunikační médium (typicky měď). Odposlouchávací zařízení je následně připojeno a může odposlouchávat komunikaci, aniž by se komunikace přerušila. Toto zařízení zpravidla vyvádí odposlechnutý signál pomocí portu do jiného zařízení [13].

²Koncovka k zapojení síťových kabelů. Může mít samčí a samičí provedení.

- **Vstupy optických vláken** — neboli „*Fiber optic taps*“ odposlouchávací zařízení pro optická vlákna fungují na stejném principu jako již popsané „*Inline network taps*“ zařízení.



Obrázek 2.2: Ukázka použití odposlouchávacího zařízení „*Vampire tap*“ na drátovém komunikačním médiu. Převzato ze stránek www.wikimedia.org⁴.

Bezdrátová komunikace

Bezdrátová komunikace je nejčastěji zprostředkována pomocí rádiových vln, jelikož dokáží procházet skrz dřevo, cihly, sklo, tenkou vrstvu betonu, a skrz další materiály. Všechny materiály přes které vlny prochází, snižují sílu signálu. Jako komunikační médium lze použít i světlo, například infračervené avšak daleko více používané jsou rádiové vlny [18].

Výhodou bezdrátové komunikace je, že není zapotřebí žádné drátové spojení a každý je tedy schopen komunikovat. Toto je však i nevýhodou a proto je zapotřebí, aby komunikace byla zašifrovaná. Nicméně pokud například útočník získá kryptovací klíč či prolomí šifru, má přístup k veškeré komunikaci a nikdo nezjistí, že komunikace je odposlouchávána. I přes to, že je komunikace prostřednictvím zabezpečené Wi-Fi sítě šifrována, lze z ní získat užitečné informace pro účely síťového forenzního vyšetřování [13]. Například lze zjistit přítomnost přístupového bodu, jeho název, MAC adresy všech připojených zařízení a další detaile o síti.

Přepínače

Přepínače jsou zařízení, která slouží ke směrování síťového provozu v LAN sítích pomocí MAC adres. Každý přepínač obsahuje CAM neboli mapovací tabulkou portů na MAC adresy síťových karet zařízení v síti. Tuto tabulkou lze použít k vyhledání konkrétního zařízení v síti. Některé přepínače také umožňují tzv. „*mirror traffic*“, což znamená duplikovat komunikaci z jednoho nebo více portů na jiný port, čímž je možné odposlouchávat komunikaci. [13].

Směrovače

Směrovače propojují jednotlivé podsítě a síť a tím umožňují komunikaci v rámci různých síťových segmentů. Aby směrovač mohl směrovat pakety, musí mít směrovací tabulkou, ve které

⁴Zdroj — <https://upload.wikimedia.org/wikipedia/commons/thumb/7/72/VampireTap.jpg/440px-VampireTap.jpg> [5. května 2022]

se mapují porty na směrovači k jeho dostupným sítím, ke kterým má přímý přístup, ale i ke vzdáleným sítím. Směrovače si vzájemně sdílí směrovací tabulky (protokol OSPF , RIP) [19]. Směrovače dokáží filtrovat pakety dle IP adresy či portu a tím například omezovat komunikaci. Dále také některé směrovače umí zaznamenávat komunikaci případně zamítnutých pokusů o komunikaci a posílat tyto záznamy na jiné zařízení [13].

DHCP servery

DHCP je mechanismus k dynamickému přidělování IP adres. Pokud je v síti DHCP server může v různém čase přidělit stejnou IP adresu různým zařízením. DHCP server si vede záznamy o přidělených adresách v čase k určitým zařízením, které identifikuje pomocí jejich MAC adresy [18]. Tyto záznamy jsou vyšetřovatelům užitečné při pátrání po fyzickém zařízení s určitou IP adresou. Pokud je například zaznamenána neobvyklá aktivita na určité IP adresě, mohou vyšetřovatelé použít DHCP záznamy k identifikaci fyzického zařízení, které tuto IP adresu v daném čase používalo [17].

DNS servery

Služba DNS je klíčovou součástí počítačových sítích a jedním z jejích úloh je mapování doménových jmen (www.vutbr.cz) na IP adresy (147.02.23.142). Když uživatel zadá do prohlížeče dotaz na určitou doménu, prohlížeč vytvoří dotaz na DNS server, který odpoví s příslušnou IP adresou. Pokud je odpověď DNS serveru uložena v cache paměti, odpověď je vrácena okamžitě bez nutnosti kontaktovat další DNS server. DNS servery si mohou vést záznamy o jednotlivých dotazech v čase a zaznamenávat tak aktivitu jednotlivých uživatelů. To může být zajímavou informací pro vyšetřovatele, jelikož tak mohou vědět na jaké webové stránky se uživatel dotažoval [19].

Autentizační servery

Autentizační servery slouží k centralizované správě uživatelských účtů a autentizaci uživatelů, čímž organizace eliminuje potřebu mít v rámci jedné organizace více autentizačních systémů a veškerou odpovědnost a režii s autentizací spojenou centralizuje na jedno místo. Hlavním úkolem těchto serverů je ověření totožnosti uživatele typicky uživatelským jménem a heslem, jenž se dnes zpravidla posílá šifrovanou komunikací zajištěnou protokolem SSL/TLS. Tyto servery zaznamenávají jednotlivé události jako je pokus o přihlášení. Díky těmto záznamům lze identifikovat podezřelé aktivity, jakými jsou například pokus o přihlášení z podezřelé lokace, pokus o přihlášení v podezřelou hodinu, pokusy o přihlášení hrubou silou, atd [13].

Webové proxy servery

Proxy server je komunikační prostředník mezi koncovými uzly. Ve společnostech se proxy server nejčastěji využívá k ukládání webových stránek do paměti cache a k zaznamenávání komunikace. Pokud uživatel vytvoří dotaz na webovou stránku (www.vutbr.cz), dotaz je odeslán na proxy server, který dotaz povolí, případně zamítne. Důvodem k zamítnutí může být dotaz na stránku nacházející se na černé listině pochybných webových stránek. V případě, že proxy server povolí stránku, je uživateli vrácena požadovaná odpověď [18].

Záznam komunikace procházející přes proxy server je pro vyšetřovatele cenným zdrojem informací, jelikož tak, lze získat webovou aktivitu například všech zaměstnanců ve společnosti. Pomocí těchto záznamů, lze tak zjistit například oběť útoku zvaným „*phishing*“ nebo identifikovat zdroj malware viru. Pokud si proxy server ukládá webové stránky do paměti cache, je možné vidět obsah, který byl zobrazen oběti útoku [13]. Dalším využití proxy serverů je:

- **Ochrana soukromí** — když klient použije proxy server, jeho IP adresa není viditelná pro cílový server, což může chránit soukromí uživatele.

- **Změna geolokace** — proxy servery umožňují překrýt geolokaci klienta a tak mu zpřístupnit obsah, který je geolokalizačně omezen.

2.3 Nástroje k získávání forenzních síťových dat

Síťové forenzní nástroje jsou nezbytným nástrojem pro každého, kdo se zabývá bezpečností a správou počítačových sítí. Tyto nástroje umožňují detektovat a analyzovat útoky na síť a identifikovat bezpečnostní problémy. Síťové forenzní nástroje nám umožňují monitorovat síť a získávat informace o provozu na síti. Tyto nástroje jsou nedílnou součástí vyšetřování a pomáhají vyšetřovatelům analyzovat již spáchané útoky nebo naopak detektovat nadcházející [17]. Některé síťové forenzní nástroje umožňují také provádět „*incident response*“, což je proces rychlé reakce na bezpečnostní incidenty a minimalizace škod způsobených útokem. Vyšetření a objasnění útoku probíhá z mnoha důvodů, uvedu 3 základní.

- **Jak se útočník dostal do systému?** — toto je velice důležitá otázka pokud dojde k útoku. Pokud útočník nějakým způsobem obejde zabezpečení systému je nutné zjistit jakým způsobem prolomil systémová zabezpečení a systém patřičně upravit aby nedošlo k dalšímu útoku.
- **Co útočník v systému dělal?** — otázka, kterou si položí většina lidí a hlavně majitel onoho systému. Je důležité vědět zda došlo například k infiltraci dat ze systému ať už to jsou osobní informace zaměstnanců či jiná citlivá data. Takto infiltrovaná data mohou být zveřejněna, použita k vydírání atd.
- **Kdo je útočník?** — zjištění pravé identity útočníka k jeho následnému dopadení.

Tyto nástroje jsou využívány nejen vyšetřovateli pro forenzní analýzu ale také administrátory sítí k odstraňování problémů na síti, vývoji nových programů, komunikačních protokolů atd. Například mohou být použity k identifikaci slabých míst v síti, zlepšení výkonu sítě a optimalizaci použití sítě. V neposlední řadě, některé z nich jsou používány také studenty ke studijním účelům [14]. Síťové forenzní nástroje umožňují také vytvořit detailní logy síťového provozu, což může být užitečné pro další analýzu a zpracování dat. Tyto logy mohou být také použity k vytvoření zpráv pro management nebo jako důkazy v soudních procesech.

Níže uvedené nástroje jsem analyzoval z hlediska UX a UI. Cílem této analýzy bylo zjistit jak jsou nástroje implementovány. Získané znalosti jsem poté využil při návrhu mé aplikace, která je hlavním předmětem této práce. Analýzu některých nástrojů jsem prováděl přímo v aplikaci, zatímco jiné jsem analyzoval pouze na základě snímků nebo videí.

2.3.1 Síťové forenzní nástroje pro analyzování

Síťové forenzní nástroje pro analyzování (NFAT) slouží zejména k analyzování a ukládání síťových dat jakožto důkazních materiálů k jejich následnému šetření. Mohou být jak softwarové tak hardwarové. Existují jak proprietární, tak i open-source nástroje. Většina open-source nástrojů je vytvořena pro prostředí Linux. Jelikož se jedná o open-source licenci, lze tyto nástroje libovolně modifikovat, doprogramovávat funkcionalitu a přizpůsobit si nástroj konkrétním potřebám [17].

Síťoví analytici a vyšetřovatelé běžně analyzují odposlechnuté pakety za účelem identifikace paketů, které mají specifický význam pro vyšetření případu. Jelikož na síti může být velký provoz, jednotlivé pakety lze filtrovat podle komunikačního protokolu, čísla portu, adresy odesílatele/příjemce, případně podle obsahu paketu a jiných kritérií k identifikování konkrétních paketů.

NetDetector

NIKSUN NetDetector je systém pro analyzování síťového provozu postavený na NIKSUN architektuře. Poskytuje zachycování paketů, generování metadat, IDS/IPS systémy, rekonstrukci jednotlivých relací a jiné [21]. Díky těmto funkcím může NIKSUN NetDetector poskytnout uživatelům velmi podrobné informace o provozu v jejich sítích. Niksun NetDetector má dvě varianty:

- NetDetector — desktopová aplikace poskytuje uživatelům vysokou rychlosť a efektivitu při analýze síťového provozu.
- NetDetector Live — webová aplikace je pak ideální pro uživatele, kteří potřebují přístup k systému z různých zařízení a míst.

NetDetector je využíván více než tisíci společnostmi včetně společností Fortune 500, vládních agentur a internetových providerů [15]. Díky této popularitě se jedná o osvědčený a spolehlivý nástroj pro analýzu síťového provozu.

Xplico

Xplico je open-source nástroj pro forenzní analýzu paketů na UNIX systémech. Podporuje celou řadu protokolů, včetně HTTP, SIP, IMAP, POP, SMTP, TCP, UDP, IPv6, Facebook, MSN, RTP, IRC, Paltalk, a dalších. Cílem aplikace Xplico je extrahat aplikační data ze zachyceného provozu [8]. Detailní popis tohoto nástroje naleznete v kapitole 2.4.

NetworkMiner

Dalším velice populárním nástrojem pro odposlech a analýzu je NetworkMiner, který má také open-source licenci, avšak lze zakoupit i profesionální licence s rozšířenou funkcionalitou např. export XML, extrakce audio hovorů atd. Byl vyvinut v roce 2007 a od té doby se stal velice populárním nástrojem používaným společnostmi po celém světě. Primárně byl vyvinut pro operační systém Windows, ale lze jej použít i na systému Linux a MacOS. Detailní popis tohoto nástroje naleznete v kapitole 2.4.

2.3.2 Síťové programy pro skenování

Existuje celá řada nástrojů pro síťové skenování, které mají různé funkce a vlastnosti. V této sekci uvedu Angry IP Scanner a Wireless Network Watcher. Síťové skenování může být užitečné pro správce sítě, kteří potřebují získat přehled o aktivních zařízeních a službách v síti. Vyhledávání probíhá za pomocí „ping sweepování“, které vráci informace o aktivních uživatelích v síti a port jejich nabízené služby. Je možné invertovat tento proces za účelem vyhledání, která IP adresa patří aktivnímu uživateli [17].

Angry IP Scanner

Angry IP Scanner, také zvaný IPScan je multiplatformní nástroj s open-source licencí s více než 29 miliony stažení i přestože instalace není nutná. Angry IP Scanner lze používat přímo v prohlížeči a jedinou prerekvizitou je Java Runtime Environment. IPScan dokáže skenovat IP adresy a jejich porty jakékoli vzdálenosti. Naskenované výsledky lze uložit do .csv, .txt, .xml souborů. Angry IP Scanner skenuje síť více vláknovým přístupem za účelem zvýšení rychlosti skenování [3].

Wireless Network Watcher

Wireless Network Watcher je „*freeware*“ nástroj vyvinutý společností NirSoft Inc. Skenuje bezdrátové sítě a pro každý aktivní uzel zajistí IP adresu, MAC adresu, NIC⁵ výrobce, název zařízení, datum prvního spuštění a status [17]. Podporuje exportování vyhledaných výsledku do *.xml*, *.txt*, *.html* a dalších souborů.

2.3.3 Síťové nástroje pro monitorování

Nástroje pro monitorování sítě konstantně monitorují síť a upozorní na anomálie administrátora pokud se síť vyskytne v předem definovaném stavu [14]. Typickým stavem pro vytvoření upozornění, je například nedostupnost nějaké služby, způsobená pádem serveru. Síťové monitorování je v podstatě kolekce jednoduchých nástrojů a příkazů operačního systému. Typicky se při monitorování sleduje výkonost, QOS, doba odezvy, šířka pásma, atd [17].

Wireshark

Jedním z nejznámějších a nejpoužívanějších síťových nástrojů pro odposlech a analýzu paketů je právě Wireshark, původně pojmenovaný Ethereal. Wireshark je open-source dostupný na GNU/Linux, Windows, MacOS a ostatních UNIX systémech. Pro odposlech paketů Wireshark používá libpcap⁶ pro odposlechnutí paketů ze sítě. Sítě, které nepodporují libpcap, Wireshark nedokáže odposlouchávat [17].

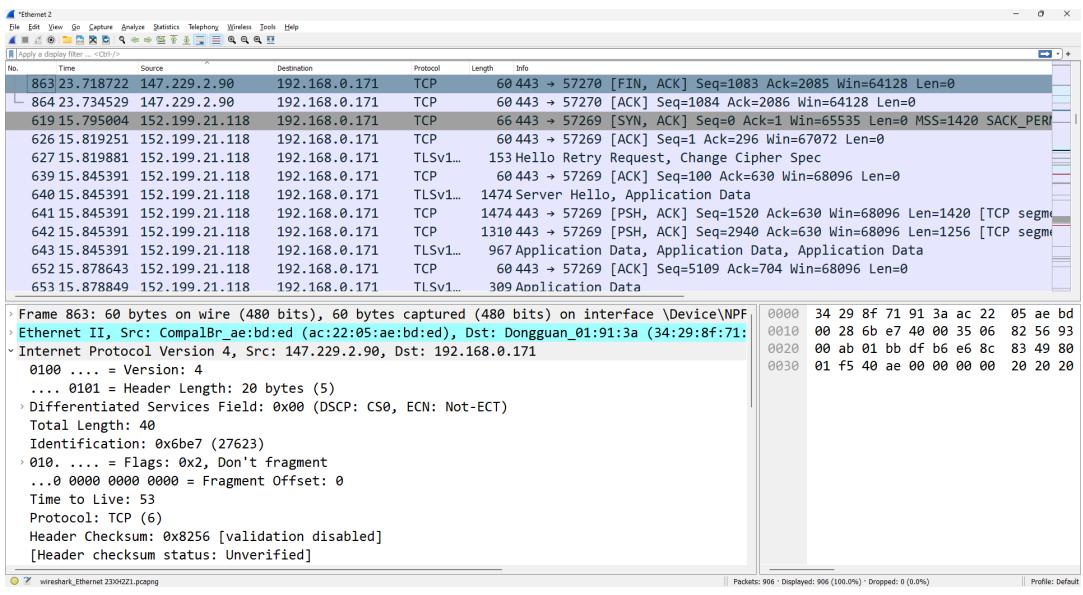
Wireshark analyzuje pakety pomocí nástroje tshark, který je součástí Wiresharku. Pomocí nástroje Wireshark lze také analyzovat předem odposlechnutou komunikaci uloženou ve formátu *.pcap* a poskytnout základní informace o paketech. Poskytuje jak uživatelsky přívětivé grafické prostředí, tak i ovládání pomocí příkazové řádky. Wireshark nabízí filtrování irrelevantních paketů dle uživatelských preferencí. Existuje více než 105000 různých kombinací filtrů, které může uživatel zvolit a tím zpřesnit zobrazení paketů, které uživatele zajímají.

Nástroj Wireshark také využívá Ngrep⁷ avšak tento nástroj prohledává vyhledávaný obsah v každém paketu zvlášť. Což znamená, že pokud je právě vyhledávaný obsah rozprostřen do více paketů, tak jej Ngrep nenajde [13].

⁵Network interface controller - síťová karta, která připojuje zařízení k počítačové síti

⁶Multiplatformní knihovna, poskytující aplikační rozhraní k přijímání paketů ze sítě [1]

⁷Nástroj postavený nad knihovnou libpcap k identifikování paketů obsahující či neobsahující vyhledávaný řetězec či binární sekvenci kdekoli v paketu [13]



Obrázek 2.3: Ukázka zachycené komunikace v programu Wireshark.

VisualRoute

VisualRoute je diagnostický nástroj, dostupný na Windows a MacOS systémy, který vizuálně zobrazuje výsledky „traceroute“ programu. Mezi klíčové nástroje tohoto programu patří nepře-tržité trasování, reverzní trasování, skenování sítě, grafické znázornění doby odezvy atd.

Ntop

Ntop je open-source nástroj, který lze používat jak ve webovém prohlížeči tak pomocí příkazové řádky. Při použití webového prostředí Ntop vykresluje grafy a poskytuje statistické hodnoty z výsledků. Nástroj Ntop umí analyzovat komunikační provoz na síti, filtrovat výsledky a geolo-kalizovat hosty [17].

2.3.4 Nástroje pro posouzení zranitelnosti systému

V dnešní době, kdy jsou informační systémy často napadány a zneužívány neoprávněnými oso-bami, je důležité mít k dispozici nástroje, jenž posoudí zranitelnost těchto systémů. Mezi jedny ze známějších patří Metasploit a Nessus. Tyto nástroje skenují systém a hledají známé systémové bezpečnostní problémy. V některých případech mohou připomínat útok na systém za účelem vyhledání bezpečnostních dér v systému.

Metasploit

Metasploit je nástroj pro testování průniknosti systémů, dostupný na Windows, Linux a Ma-cOS. Na operačním systému Kali Linux je Metasploit předinstalovaný. Přichází ve dvou verzích Metasploit Framework (open-source) a Metasploit Pro (komerční).

- **Metasploit Framework** — platforma poskytující psaní, testování a spouštění tzv. ex-ploitů programů. Obsahuje nástroje, se kterými lze testovat bezpečnostní zranitelnosti, pro-vádět útoky atd. MSFconsole poskytuje rozhraní pro práci s Metasploit frameworkem [4].
- **Metasploit Pro** — na rozdíl od Metasploit Frameworku, Metasploit Pro přichází s webo-vým rozhraním. Dále také nabízí exploit programy, průvodce standardními audity, testo-vání webových aplikací proti top 10 OWASP bezpečnostních pochybností a další funkci-onality [5].

Nessus

Nessus vlastněný společností Tenable Network Security je nástroj dostupný na Windows, Linux, Solaris, FreeBSD a MacOS. Přichází ve dvou variantách Nessus Expert a Nessus Professional a obě tyto verze jsou komerční. Nessus disponuje nejnižším počtem falešně pozitivních výsledků (0.32 zmetků při milionů skenování) a má nejširší pokrytí ze všech nástrojů v tomto odvětví [6].

2.4 Analýza nástrojů Xplico a NetworkMiner

Nástroje Xplico a NetworkMiner byly již stručně popsány v předešlé kapitole 2.3.1. V této kapitole se budu zabývat analýzou těchto forenzních nástrojů detailněji. Pokusím se uvést silné a slabé stránky těchto nástrojů a na závěr každého z nástrojů uvedu případy užití, které slouží k demonstraci jejich funkcionality.

NetworkMiner a Xplico patří mezi předně používané nástroje a oba disponují open-source licencí. Nástroje slouží k analyzování a následné vizualizaci forenzně sítových dat. Jsou navrženy tak, aby usnadňovaly práci vyšetřovatelům a odborníkům v oboru informační bezpečnosti. Usnadnění práce následně vede k významnému urychlení řešených případů, což může vést ke snížení kybernetických útoků. Oba tyto nástroje lze také využít ke studijním účelům nebo pro monitorování a diagnostiku sítových problémů. Získané poznatky z těchto nástrojů budou aplikovány při tvorbě nástroje pro vizualizaci forenzně sítových dat, který je hlavním záměrem této práce.

2.4.1 Xplico

Sítový forenzní nástroj pro analýzu Xplico je nástroj vyvinutý pouze pro operační systém Linux a k zobrazení analyzovaných dat a interakci s uživatelem používá webové rozhraní. Xplico provádí analýzu dat nad již zachycenými daty, které lze nahrat v souboru s formátem patřícím do rodiny formátů *.pcap* a zároveň lze pomocí tohoto nástroje zachytávat síťovou komunikaci v reálném čase. Analyzování a odchytávání paketů se provádí nad knihovnou libpcap. Xplico podporuje mnoho protokolů (vypsány v kapitole 2.3.1).

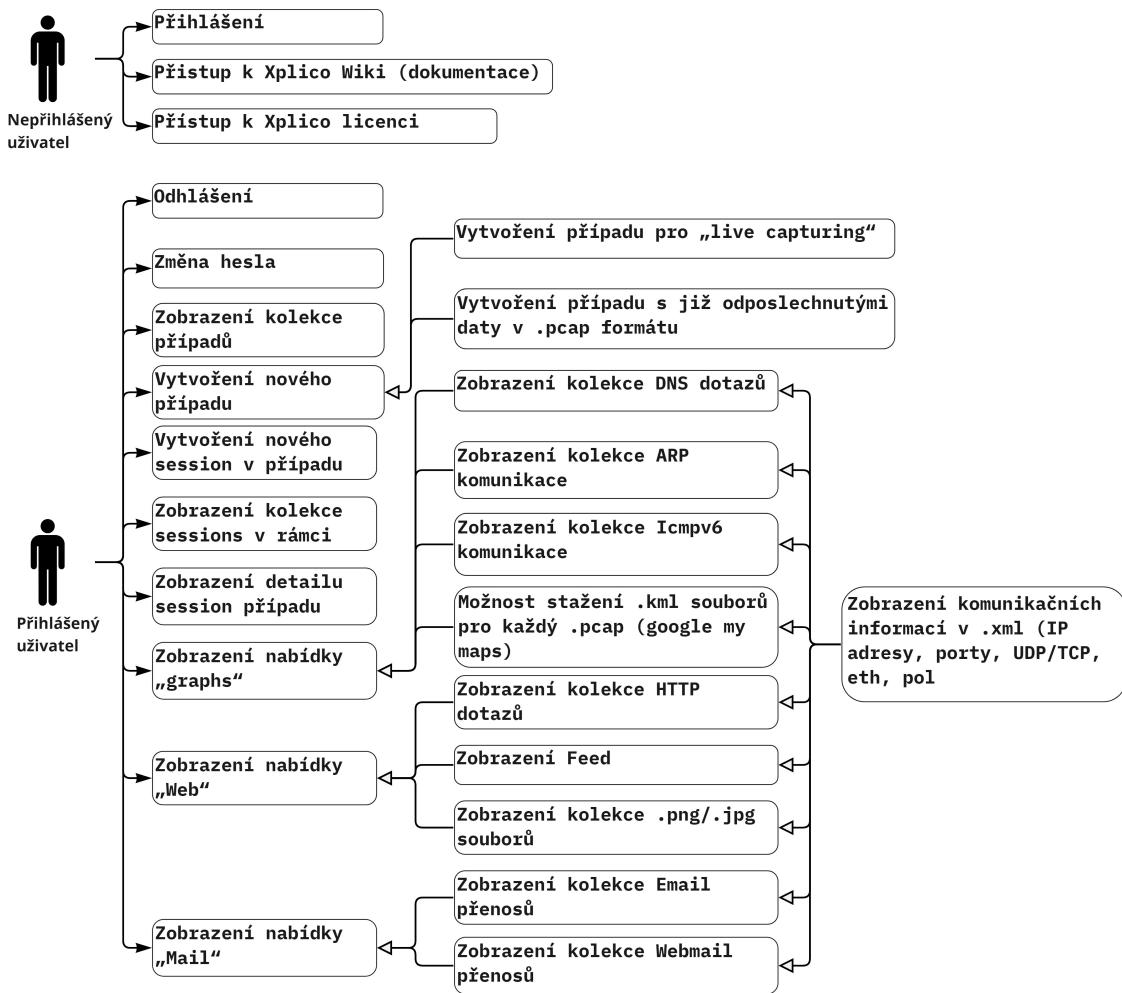
Xplico podporuje přístup k jednomu případu více uživatelům, což může být vhodné například pro vyšetřovatele, pracující na stejném případu. Lze využít databázi SQLite, MySQL či PostgreSQL pro zachování persistence dat [8]. Dále umožňuje filtrovat a procházet zachycená data podle jistých kritérií, jako jsou například čísla portů, použitý protokol a jiné. Analyzovaná data je možné exportovat do formátu *.xml* a používat dále v jiných nástrojích.

Xplico poskytuje dokumentaci popisující jak nástroj nainstalovat a základní použití, avšak u komerčních nástrojů bývá dokumentace rozsáhlější. Grafické prostředí není zcela uživatelsky přívětivé, na některých stránkách chybí popis zobrazených dat a uživatel tak neví, co zobrazená data znamenají⁸. Dalším velkým nedostatkem je chybějící agregace dat dle nějakého kritéria nebo agregace kolekce zachycených zpráv v podobě grafů. V následující části této kapitoly předvedu jeden běžný případ užití popisující základní funkcionality nástroje Xplico a jeden specifický soutěžní, který popisuje práci s nástrojem Xplico při řešení konkrétního případu.

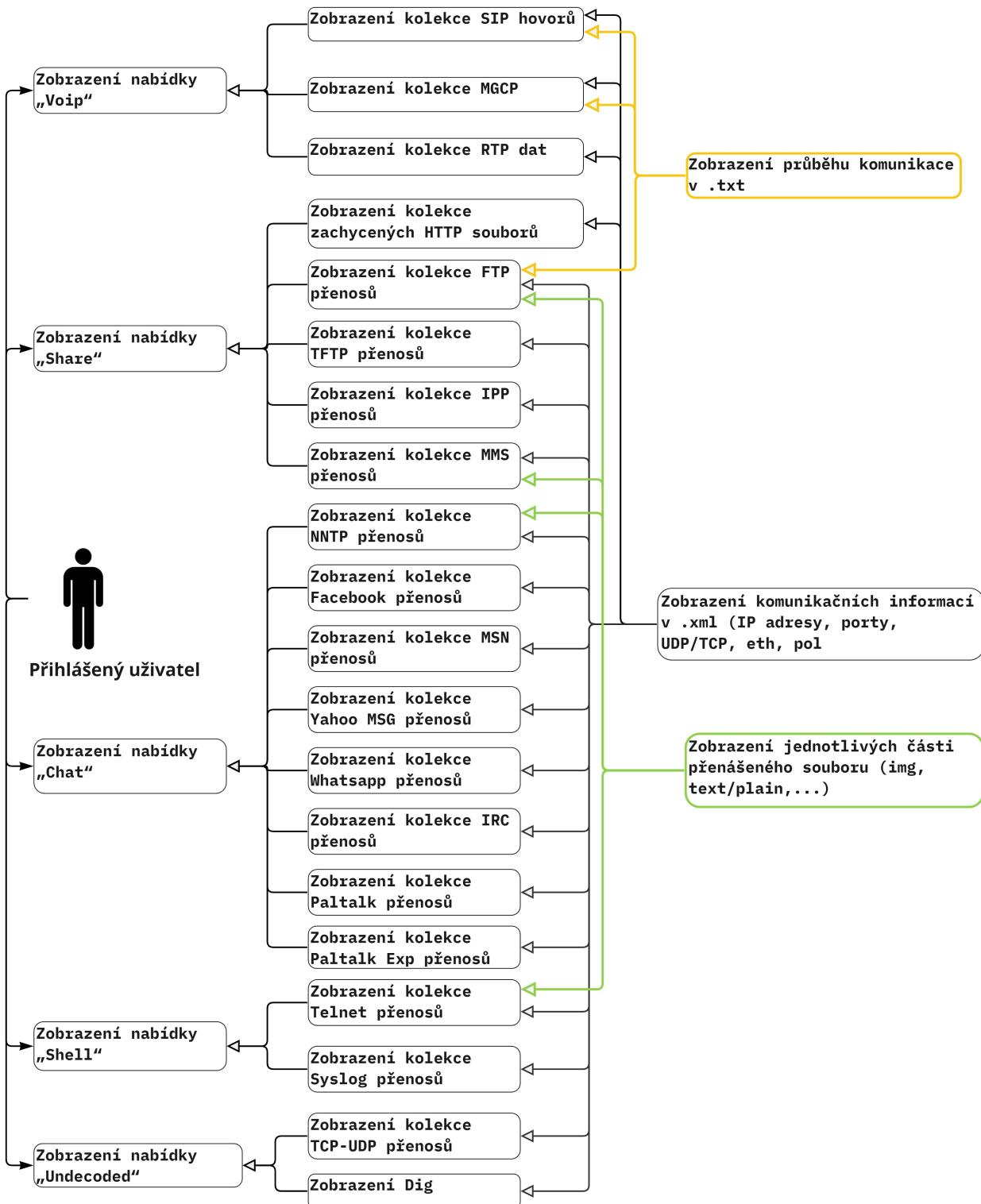
Běžný případ užití

Následující diagram užití slouží k vizuální reprezentaci různých scénářů, které lze provádět s tímto nástrojem. Tento diagram zahrnuje všechny funkcionality nástroje Xplico. Diagram je zde zobrazen k porovnání funkcionalit s nástroji NetworkMiner a mnou vytvořené Blazor aplikace.

⁸Například při zobrazení detailní stránky zachyceného emailu, Xplico zobrazí dvě emailové adresy a není u nich popis, kdo je odesílatel a kdo adresát.



Obrázek 2.4: První část případu užití síťového forenzního nástroje pro analyzování Xplico.



Obrázek 2.5: Druhá část případu užití síťového forenzního nástroje pro analyzování Xplico.

Specifický případ užití nástroje Xplico

Při vytváření úloh pro tento případ užití jsem se inspiroval soutěžním⁹ zadání specifických incidentů společně se zachycenou komunikací. V nástroji Xplico jsem prováděl jednotlivé úlohy

⁹<https://forensicscontest.com/2009/10/10/puzzle-2-ann-skips-bail> — Kompletní zadání včetně odkazu ke stažení zachycené komunikace. Citováno dne 7. 2. 2023.

a popsal jak získat pomocí tohoto nástroje hledaná data, to zobrazuje tabulka 2.1. Cílem případu užití je zjistit informace o jisté Anně a jejím milencem z elektronické pošty.

1	Jaká emailová adresa náleží Anně?
	Zobrazení kolekce emailů v záložce „Mail“. Po otevření detailu emailové zprávy lze u emailového jména „ <i>sneakyg33k@aol.com</i> “ vidět jméno Anny.
2	Jaká je emailová adresa milence Anny?
	Zobrazením kolekce emailů v záložce „Mail“ a analyzováním jednotlivých zpráv zaslaných Annou lze určit adresáta.
3	Jaké dva předměty řekla Anna svému milenci aby přinesl?
	Zobrazení kolejce emailů v záložce „Mail“. Vyhledání zprávy pro adresáta „ <i>mister-secretx@aol.com</i> “ a otevření detailu zprávy.
5	Kolik příloh odeslala Anna celkem?
	Xplico nezobrazuje celkový počet příloh konkrétního odesílatele. Je tedy nutné otevřít každou elektronickou zprávu a manuálně je spočítat.
4	Název přílohy, kterou Anna poslala svému milenci?
	Zobrazení kolejce emailů v záložce „Mail“. Vyhledání zprávy pro adresáta „ <i>mister-secretx@aol.com</i> “. Po otevření detailu zprávy lze vidět jednotlivé přílohy připojené ke zprávě.

Tabulka 2.1: Specifický případ užití nástroje Xplico.

2.4.2 NetworkMiner

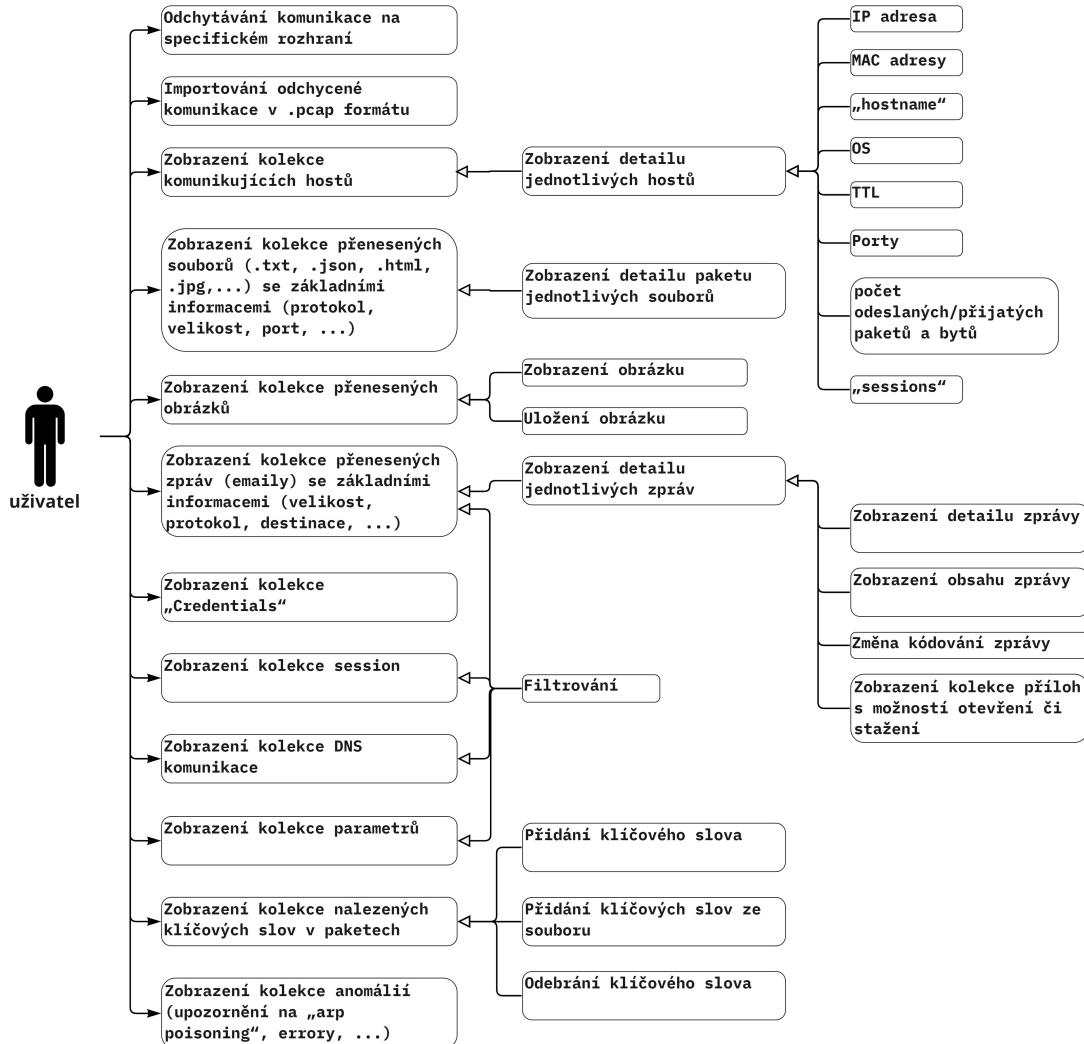
Síťový forenzní nástroj pro analýzu NetworkMiner je multiplatformní nástroj, který přichází ve dvou variantách a těmi jsou „*NetworkMiner Free Edition*“, která má open-source licenci a „*NetworkMiner Professional*“. Verze NetworkMiner Professional je placenou verzí, která nabízí přidanou funkcionality (export do *.xml*, *.csv*, geolokalizaci IP adresy, extrahování audio záznamu VoIP hovorů a další).

NetworkMiner poskytuje jak analýzu již zachycené komunikace, tak aktivní odposlech síťové komunikace ze síťového rozhraní. Lze jej také použít pro extrahování souborů, emailů a hesel ze zachycené komunikace v souborech *.pcap* [7]. NetworkMiner shromažďuje data spíše o uživateli než data o provozu na síti tzn. IP adresu, hostname, operační systém, porty, přihlašovací údaje zadáné uživatelem, atd [17]. Přestože je NetworkMiner velice intuitivní aplikací, chybějící dokumentace tohoto nástroje může být pro některé uživatele problémem. Dalším velkým nedostatkem je chybějící agregace dat dle nějakého kritéria nebo agregace kolejce zachycených zpráv v podobě grafů. V následující části této kapitoly předvedu jeden běžný případ užití popisující základní funkcionality nástroje NetworkMiner a jeden specifický soutěžní, který popisuje práci s nástrojem NetworkMiner při řešení konkrétního případu.

Běžný případ užití

Následující diagram užití slouží k vizuální reprezentaci různých scénářů, které lze provádět s tímto nástrojem. Tento diagram zahrnuje všechny funkcionality nástroje NetworkMiner. Di-

agram je zde zobrazen k porovnání funkcionalit s nástroji Xplico a mnou vytvořené Blazor aplikace.



Obrázek 2.6: Případ užití síťového forenzního nástroje pro analyzování NetworkMiner.

Specifický případ užití

Pro tento případ užití jsem využil stejné úlohy jako při analyzování nástroje Xplico viz sekce 2.4.1 v této kapitole. V následující tabulce 2.2 můžete vidět jak lze jednotlivé úlohy řešit pomocí nástroje NetworkMiner.

2.4.3 Funkční a nefunkční požadavky na nový systém

Na základě analýzy všech předešlých nástrojů a důkladné analýzy nástroje Xplico a NetworkMiner jsem vytvořil funkční a nefunkční požadavky na výslednou aplikaci. Důvodem vytvoření těchto požadavků je definování vlastností a funkcionality, kterou bude aplikace disponovat. Zároveň je vhodné si definovat tyto požadavky z hlediska plánování vývoje a následného testování.

1	Jaká emailová adresa náleží Anně?
	Zobrazení kolekce zpráv v záložce „Messages“. U jména Anny lze vidět emailovou adresu a tedy odpověď: „sneakyg33k@aol.com“
2	Jaká je emailová adresa milence Anny?
	Zobrazení kolekce emailů v záložce „Mail“. Analyzováním jednotlivých zpráv zaslaných Annou lze určit adresáta.
3	Jaké dva předměty řekla Anna svému milenci aby přinesl?
	Zobrazení kolekce emailů v záložce „Messages“. Vyhledání zprávy pro adresáta „mister-secretx@aol.com“. V detailu zprávy lze v obsahu zprávy vyčíst odpověď.
4	Kolik příloh odeslala Anna celkem?
	NetworkMiner nezobrazuje celkový počet příloh konkrétního odesílatele. Je tedy nutné otevřít každou elektronickou zprávu a manuálně je spočítat.
5	Název přílohy, kterou Anna poslala svému milenci?
	Zobrazení kolekce emailů v záložce „Mail“. Vyhledání zprávy pro adresáta „mister-secretx@aol.com“. Po otevření detailu zprávy lze vidět jednotlivé přílohy připojené ke zprávě.

Tabulka 2.2: Specifický případ užití nástroje NetworkMiner.

Funkční požadavky

Funkční požadavky aplikace zahrnují vlastnosti a funkce, kterými by měla výsledná aplikace disponovat. Jsou to klíčové prvky pro návrh a specifikaci systému a musí být jasně definovány, jelikož na základě těchto požadavků se aplikace implementuje a následně testuje. Funkční požadavky na výslednou aplikaci:

- Zobrazení kolekce zpráv — aplikace musí poskytovat jednoduchý způsob pro zobrazení kolekce dat. Pro každou kolekci dat by měla existovat separátní stránka tak, aby uživatel neztratil přehled v aplikaci a zobrazovala se pouze uživatelem zvolená relevantní data.
- Filtrování nad kolekcí zpráv — každá rozsáhlejší kolejce zpráv by měla obsahovat filtrování nad daty, aby bylo možné zobrazit pouze relevantní data v rámci zvolené kolejce.
- Agregace kolejce zpráv — agregace dat by se měla vyskytovat u každé kolejce dat alespoň pomocí grafů. U kolejek týkajících se elektronických zpráv by měla být implementována aggregace dle uživatele (počet odeslaných zpráv daným uživatelem, atd.)
- Zobrazení detailních informací o zprávě — výsledná aplikace musí umožnit zobrazit detailní informace o konkrétní zprávě uživateli, pokud již veškeré informace nezobrazuje.
- Výběr případu — výsledná aplikace musí uživateli umožnit výběr případu se zachycenou komunikací (je možné, že jeden případ má více zachycených komunikací), který bude zaznamenán a aplikace bude zobrazovat data nad daným případem.
- Stránkování — pokud stránka zobrazuje kolejci dat o vyšším počtu zachycených zpráv, musí obsahovat stránkování.

Nefunkční požadavky

Na rozdíl od funkčních požadavků, nefunkční požadavky popisují kvalitu a vlastnosti systému. Tyto vlastnosti nesouvisí s funkcionalitou, nýbrž popisují například spolehlivost, bezpečnost, rozšiřitelnost systému atd. Nefunkční požadavky na výslednou aplikaci.

- Rozšiřitelnost — výsledná aplikace musí být implementována s určitou abstrakcí a modularizací tak aby případná modifikace byla jednoduchou záležitostí.
- Spolehlivost — aplikace musí být spolehlivá, nesmí v běhu vyhazovat chybové hlášky a implementovaná funkcionalita musí fungovat.
- Výkonost — aplikace by měla jet rychle bez větších či zbytečných prodlev tak, aby ji uživatelé mohli používat snadno a rychle.
- Uživatelské rozhraní — systém by měl být intuitivní a uživatelsky přívětivý, aby bylo jednoduché se systémem pracovat.

Kapitola 3

Návrh a implementace

Tato kapitola se zaměřuje na návrh a implementaci vizualizačního nástroje s využitím různých technologií a knihoven, jenž by usnadnil práci vyšetřovatelům při šetření. Kapitola začíná představením použité technologie, která byla zvolena při vývoji včetně SPA, WebAssembly a framework od společnosti Microsoft Blazor. Vytvořením tohoto nástroje jakožto SPA poskytuje uživateli plynulý zázitek, zatímco technologie WebAssembly zajišťuje výkonnost téměř na úrovni nativního kódu. Webový framework Blazor umožňuje psát webové aplikace pomocí jazyka C# namísto jazyku javascript a umožňuje tedy, aby jak backend tak i frontend byl napsán ve stejném jazyce, což zjednoduší práci například při použití stejných objektů pro přenos dat atd.

Po detailním popsání výše zmíněných technologií, popíšu použité knihovny, které výrazně zjednodušili práci při vývoji. Knihovna MudBlazor, na které je celá Blazor aplikace postavena a poskytuje předem připravené UI komponenty, se kterými interaguje uživatel. Dále zmíním knihovnu ApexCharts, která je v aplikaci použita ke grafické vizualizaci dat v podobě grafů. Dále v této kapitole popíši jak jsem navrhl aplikační rozhraní, které slouží ke komunikaci mezi Blazor aplikací a serverem a popíšu postup při vytváření objektů pro datový přenos.

Následuje sekce, která se zabývá návrhem aplikace, nejprve byl vytvořen drátový model, který sloužil zejména na ucelení myšlenek, návrhu funkcionality na jednotlivých stránkách, tak aby se docílilo maximálního UX. Po vytvoření drátových modelů jsem také vytvořil grafické modely, které sloužily k rozvržení jednotlivých komponent na stránce k dosažení pěkného vzhledu.

V poslední sekci této kapitoly popisují implementaci webové aplikace. Aplikaci jsem rozdělil na server, Blazor aplikaci a sdílenou knihovnu obsahující objekty pro datový přenos. Všechny tyto celky důkladně popisuji, včetně generování testovacích dat k zaplnění aplikace.

3.1 Použité technologie

V rámci vytváření webové aplikace jsem použil moderní technologie. Mezi tyto technologie patří SPA, umožňující načítání obsahu stránky bez nutnosti dotazování serveru o HTML elementy. Dále byla použita technologie WebAssembly, pomocí níž je možné spustit binární kód v prohlížeči namísto javascriptu a zároveň přináší výkonnostní výhodu. Dále byl využit framework Blazor, který překládá C# kód do WebAssembly.

Kromě těchto technologií byly také použity knihovny ke zjednodušení práce a to zejména při vytváření UI. Na knihovně s připravenými UI komponentami MudBlazor je postavena celá aplikace. Dále je také využita knihovna ApexCharts k vizualizaci dat v podobě grafů. Poslední použitou knihovnou byla knihovna Bogus, pomocí níž jsem vytvářel falešná data k zaplnění a otestování funkcionality aplikace.

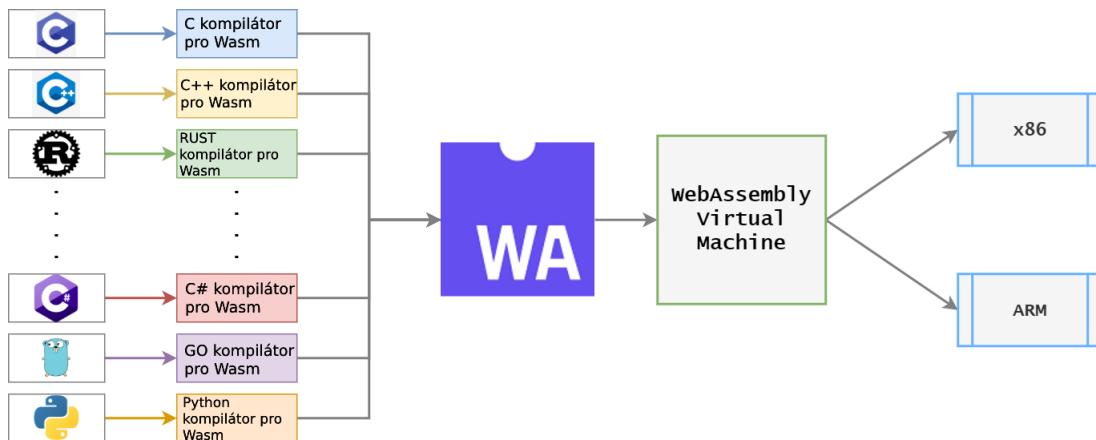
3.1.1 Jednostránková aplikace

Cílem SPA neboli jednostránkové aplikace je vytvořit rychlou webovou aplikaci s podobnou interaktivitou jako u mobilních aplikací. Proto jsou stále populárnější oproti klasickým webovým stránkám [20]. SPA si při prvotním načtení stáhne ze serveru celý webový obsah a poté pomocí zvolené technologie (Javascript, WebAssembly) mění zobrazený obsah na základě uživatelské interakce [9]. V případě klasických webových stránek se ze serveru stáhne vždy jen požadovaná stránka a v případě, že se uživatel odkáže na jinou stránku se proces opět opakuje [20].

Doba stažení stránky u klasických webových stránek je závislá na rychlosti připojení a velikosti stránky, typicky však stovky milisekund až několik vteřin. Při použití SPA, kdy se obsah pouze přerenderuje a není stahován ze serveru je doba značně kratší. Opět závisí na řadě faktorů (složitost stránky, zpracovávaná data, …), ovšem doba zobrazení je v rázech milisekund. Prvotní načtení SPA bývá delší jelikož se stahuje větší obsah a více dat. To se však dá snížit například pomocí „Lazy loading“ techniky.

3.1.2 WebAssembly

WebAssembly neboli WASM je binární instrukční formát. Je to výsledek komplikace programovacích jazyků, jenž lze nasadit na ve webových prohlížečích podporující tuto technologii [12]. To znamená, že WebAssembly nám umožňuje spouštět kód napsaný v jakémkoliv jazyce ve webovém prohlížeči, za předpokladu, že jej zkompilujeme do WebAssembly. Grafické znázornění na obrázku níže. Jedna ze zásadních výhod je jednoznačně rychlosť, kód lze spustit téměř stejnou rychlosťí jako nativní kód [12]. WebAssembly je webový standart podporovaný prohlížeče bez nutnosti instalace doplňkových balíčků a funguje ve všech moderních internetových prohlížečích (Chrome, Safari, Firefox, Edge, …) včetně mobilních [10].



Obrázek 3.1: Vztah programovacích jazyků k WebAssembly¹.

3.1.3 Blazor

Blazor je webový framework od společnosti Microsoft pro vytváření interaktivních aplikací. Pomocí Blazor frameworku lze vytvářet Blazor WebAssembly aplikace, což jsou aplikace, jedoucí na straně klienta a Blazor Server aplikace, které jedou na straně serveru, v obou případech se jedná o Blazor framework a mění se pouze, kde je aplikace hostována. Aplikace vytvořené pomocí tohoto frameworku jsou založené na komponentách, kdy každá komponenta je definována jako třída jazyka C#. Komponenta je prvek uživatelského rozhraní. Pod tím si lze představit

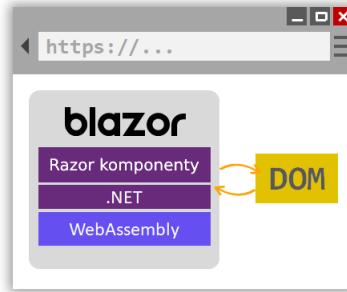
¹Vztah programovacích jazyků k WebAssembly — zdroj viz https://arghya.xyz/images/posts/webassembly/WebAssembly_compile.png [6. května 2023]

stránku, dialogové okno, formulář atd. Blazor komponenty se vytváří pomocí Razor syntaxe, která umožňuje vývojářům kombinovat značkovací jazyk HTML společně s programovacím jazykem C# [10]. Tyto komponenty se ukládají s příponou *.razor* jejichž výchozím jazykem je HTML a pomocí znaku „@“ se lze přepnout do jazyka C#.

Blazor WebAssembly

Blazor WebAssembly je SPA framework pro tvorbu interaktivních aplikací na straně klienta pomocí jazyka C#. Dle Microsoft dokumentace Blazor WebAssembly aplikace funguje následovně.

- C# kód a *.razor* soubory jsou zkompilovány
- Zkompilované soubory společně s .NET runtime² jsou staženy do prohlížeče
- Blazor WebAssembly v prohlížeči spustí .NET runtime jenž spustí aplikaci v javascriptem vytvořeném sandbox³ prostředí. Aplikace využívá Javascriptovou interoperabilitu k manipulování s DOM a voláním aplikačního rozhraní prohlížeče.



Obrázek 3.2: Grafické znázornění Blazor WebAssembly a DOM⁴.

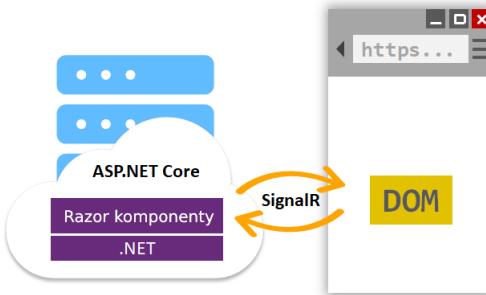
Blazor Server

Na rozdíl od WebAssembly běžící na straně klienta, Blazor Server je aplikace, běžící na straně serveru a uživatelské prostředí se mění pomocí SignalR spojení [10] ze serveru. SignalR je open-source knihovna, která slouží primárně k real-time webovým aplikacím. Mezi takové aplikace patří aplikace, které vyžadují frekventované aktualizace ze serveru, konkrétním příkladem jsou hry, mapy, aplikace vyžadující notifikace (sociální, chatovací, emailové, ...) a další [11]. .NET runtime se neposílá ze serveru do prohlížeče jako u Blazor WebAssembly, nýbrž zůstává na serveru, kde připraví uživatelské prostředí ve formě HTML na základě uživatelských interakcí, které klient posílá na server a poté pošle zpět klientovi HTML soubor.

²Virtuální stroj, který spouští zkompilovaný C# kód. Obsahuje JIT kompilátor, který zkompiluje zkompilovaný C# kód (*.dll* nebo *.exe* soubory) do nativního kódu, který je následně spuštěn počítačem.

³Vymezené prostředí ve kterém mohou programy jet aniž by měli dopad na vnější systém. Sandbox prostředí se využívá zejména k zajištění bezpečnosti. V prohlížeči se toto prostředí využívá aby se například útočník nedostal k heslům uloženým v prohlížeči, historii prohlížení, cookie souborům atd.

⁴Grafické znázornění Blazor WebAssembly a DOM — zdroj viz <https://learn.microsoft.com/cs-cz/aspnet/core/blazor/?view=aspnetcore-7.0> [6. května 2023]



Obrázek 3.3: Grafické znázornění Blazor Server a DOM⁵.

3.1.4 Použité knihovny

Vývoj aplikací se díky rostoucí poptávce po uživatelské přívětivosti, funkcionalitě a responzivitě stává stále náročnější. Abych dosáhl těmto požadavkům, rozhodl jsem se využít již zmíněné knihovny, které usnadňují vývoj aplikací čímž šetří čas a prostředky vývojářů. Knihovny, které jsem se rozhodl použít jsem zvolil k usnadnění práce s grafickou stránkou aplikace, čímž jsem získal časový prostor, který jsem mohl věnovat funkcionalitě klientské části. Aplikace je postavena na knihovně MudBlazor⁶ a k zobrazení složitějších grafů jsem použil knihovnu ApexCharts⁷.

MudBlazor

MudBlazor je knihovna již připravených UI komponent, pro Blazor. Komponentou může být myšleno tlačítko, modální okno, formulář, text, tabulka, atd. Každá komponenta má aplikační rozhraní, kterým lze komponentu jednoduše modifikovat a dosáhnout tak grafických či funkcionalních požadavků. S použitím MudBlazor knihovny lze vytvořit webovou aplikaci, která může být snadno použitelná a responzivní, aniž by vývojář psal veškerý HTML a CSS kód, čímž se stává mocným nástrojem při vývoji. Tato knihovna má open-source licenci a kdokoliv tedy může přispívat při vývoji.

ApexCharts

ApexCharts je open-source javascriptová knihovna pod licencí MIT, avšak lze ji použít i v Blazoru. V tomto případě se knihovna integruje do aplikace prostřednictvím interop funkcionality. Jinými slovy by se dalo říci, že kód napsaný v Blazoru zavolá javascriptový kód, který se následně spustí v prohlížeči.

Tato knihovna slouží k zobrazení dat v podobě grafů. Data lze vizualizovat pomocí čárových, sloupcových, plošných, koláčových, kuličkových, svíčkových a mnoho dalších grafů. Dále také poskytuje v těchto grafech určitou funkcionalitu jakou je například zvětšování, zmenšování, pohyb v grafu či stažení grafu v podobě obrázku ve formátu například *.png*. Tyto grafy v podobě komponent mají jednoduché rozhraní, kterým lze upravovat použité barvy či pozadí a dosáhnou tak například lepší čitelnosti nebo aplikace se změnou barevného motivu.

Bogus

Bogus je open-source knihovna, která slouží ke generování realistických falešných dat v .NET a tedy jazyky C#, F# a VB.NET. Knihovna je vysoce konfigurovatelná a lze ji tak použít pro

⁵Grafické znázornění Blazor Server a DOM — zdroj viz <https://learn.microsoft.com/cs-cz/aspnet/core/blazor/?view=aspnetcore-7.0> [6. května 2023]

⁶<https://mudblazor.com/>

⁷<https://github.com/apexcharts/Blazor-ApexCharts>

různé specifické užití. Poskytuje snadno použitelné aplikační rozhraní pro generování falešných dat z různých oblastí. Celý seznam dat, které lze pomocí této knihovny vygenerovat je na v souboru Readme.md na githubu⁸, zde uvedu pár dat, která lze s pomocí této knihovny snadno vygenerovat:

- Adresa — město, ulice, stát, směrovací číslo, číslo domu, zeměpisnou šířku a délku, ...
- Internet — emailová adresa, username, IP adresa, číslo portu, url adresa, heslo, ...
- Lidé — Jméno, příjmení, název práce, popis práce, typ práce, ...
- Systém — název souboru, typ souboru, cesta k souboru, verze, Apple Push Token, ...

3.2 Návrh definice aplikačního rozhraní

Pro zajištění bezproblémové komunikace mezi serverem a klientem bylo nutné vytvořit aplikační rozhraní a přesné definice objektů pro datový přenos, které jsou použity k přesunu informací ze serveru na klienta a k jednoduché manipulaci s daty při následné vizualizaci. Jelikož cílem této práce je pouze data vizualizovat a nikoliv modifikovat, přidávat či mazat, tak jediné dotazy, které server zpracovává jsou dotazy typu GET.

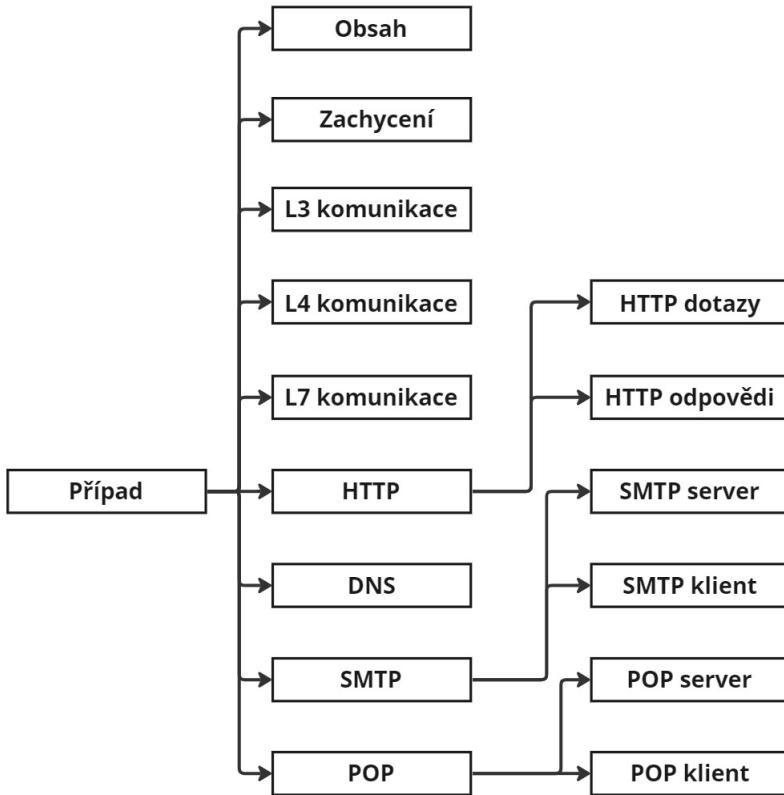
Objekty pro datový přenos jsou navrženy pro přesun dat ze serveru na klienta s ohledem na co největší efektivitu a jednoduchost zpracování na straně klienta. Při návrhu objektů pro datový přenos jsem se inspiroval u již existujících nástrojů, u kterých jsem analyzoval jaká data a v jaké podobě se zobrazují na konkrétních pohledech. Po následné konzultaci s vedoucím práce jsem došel k finální struktuře aplikačního rozhraní a vytvořeným návrhem objektům pro datový přenos, jejichž tvorbu popisuji v následujících subsekčích.

3.2.1 Struktura aplikačního rozhraní

Nástroj Xplico umožňuje uživateli vytvářet případy obsahující zachycenou komunikaci. Tímto jsem se inspiroval a rozhodl se vytvořit stejnou funkcionality také. Vytvoření případů umožní uživatelům snadněji a efektivněji analyzovat zachycenou komunikaci v rámci jednoho vyšetřovacího případu. Pro dosažení maximální efektivity při přenosu dat mezi serverem a klientem bylo nutné pečlivě navrhnout aplikační rozhraní. Základní strukturu aplikačního rozhraní vizualizuje obrázek 3.4, který popisuje základní navržení aplikačního rozhraní, nikoliv konkrétní koncové body. Koncových bodů je v rámci aplikačního rozhraní více (např. kvůli filtrování dat).

Případ slouží ke sdružení veškerých zachycených spolu souvisejících dat. To může být zejména vhodné pokud má vyšetřovatel v rámci jednoho vyšetřujícího případu více zachycených komunikací, které chce analyzovat zároveň. Koncový bod „Obsah“ poskytuje informace o případu (datum první a poslední zachycené komunikace, počet zachycených komunikací, atd.). Zachycení reprezentuje konkrétní zachycenou komunikaci a informace o ní. Zobrazenou komunikaci v rámci celého případu jsem se rozhodl zobrazovat dle vrstev referenčního modelu ISO/OSI, konkrétně na vrstvách L3, L4 a L7 a také dle aplikačních protokolů HTTP, DNS, SMTP a POP. Zachycenou komunikaci nad protokolem HTTP jsem dále rozdělit na HTTP odpovědi a HTTP dotazy a komunikaci nad protokolem SMTP a POP na komunikaci ze serveru a od klienta.

⁸<https://github.com/bchavez/Bogus>



Obrázek 3.4: Navržená základní struktura aplikačního rozhraní.

3.2.2 Objekty pro datový přenos

V této subsekci popíšu jak jsem modely vytvářel. Pro zobrazování kolekcí zpráv stejného typu (např. kolekce všech případů) jsem vytvořil „List“ model, který obsahuje pouze některé informace o konkrétní zprávě. Tento přístup jsem zvolil, abych uživateli vytvořil možnost se postupně zanořovat v aplikaci a zobrazovala se mu pouze ta data, o která má zájem. Zároveň tak šetřím data, která je nutná přeposlat ze serveru na klienta a datový přenos je tak rychlejší.

K lepšímu UX jsem se rozhodl implementovat stránkování při zobrazování kolekce vizualizovaných dat. Stránkování dat umožňuje uživateli rychlý a efektivní přístup k velkým kolekcím dat. Z tohoto důvodu jsem potřeboval uchovávat informaci, na které stránce se uživatel nachází a také kolik dat se má na stránce zobrazit, jelikož to je jedna z možností, které si může uživatel určit. To jsem vyřešil vytvořením objektů pro datový přenos se suffixem „*PageQueryResultDTO*“, který slouží jako obálka obsahující tyto informace o stránkování a kolekci všech zpráv stejného typu.

K zobrazení jedné konkrétní zprávy jsem vytvořil „Detail“ model, jenž obsahuje všechna dostupná data o zprávě. Tento model už se ze serveru na klienta odesílá samostatně a tedy nedochází ke zbytečnému zahlcení sítě, které by vznikalo pokud bych některá přijatá data ze serveru nezobrazoval. Tyto modely jsem použil například při zobrazení zpráv na vrstvě L3, L4 a L7.

3.3 Návrh uživatelského prostředí

Po navržení aplikačního rozhraní, jsem byl schopen přejít na návrh samotného uživatelského prostředí a vizualizovat jednotlivé stránky aplikace, celkový vzhled a jak bude možné aplikaci ovládat. Nejprve jsem vytvořil jednoduché drátové modely, které mi sloužili k rozvržení aplikace tak aby byla pro uživatele srozumitelná, jednoduše použitelná a nápomocná při čtení

zobrazených dat. Po vytvoření těchto drátových modelů jsem vytvořil grafický návrh, který už reprezentoval vzhled výsledné aplikace.

Návrh drátových a grafických modelů před samotným vývojem je velice běžné jelikož šetří čas a zdroje potřebné k vývoji aplikace. Vytvoření těchto modelů není tak časově náročné jako samotný vývoj. Lze tak rychle vizualizovat vzhled a použitelnost výsledného celku a zjistit tak případné nedostatky či chyby. Modely je možné následně předělat či upravit tak, aby vyhovovaly veškerým požadavkům a aplikace byla pro koncového uživatele jednoduše ovladatelná a uživatelsky přívětivá.

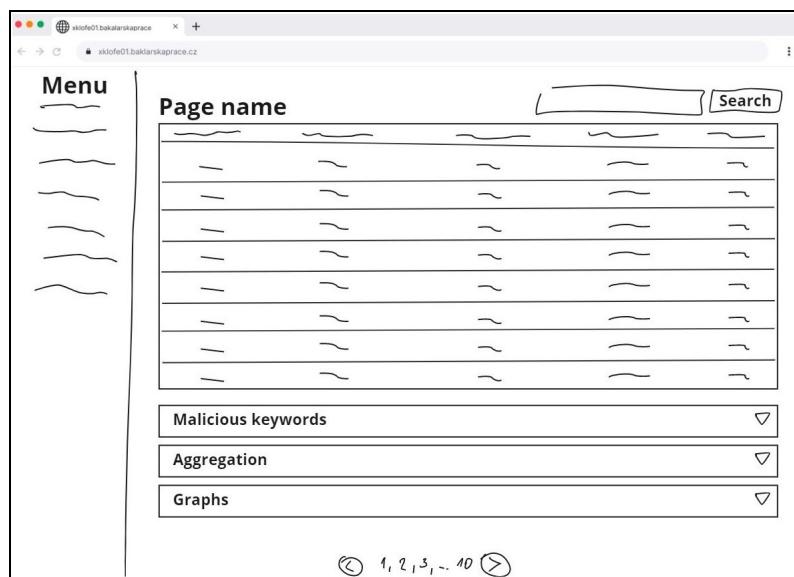
3.3.1 Drátové modely

Jedná se o velice jednoduchý model, který by měl být tvořen pouze pomocí čar a textu. Účelem je rychle vizualizovat rozložení prvků aplikace, které pak slouží grafikům při návrhu grafických UI modelů, programátorům při vytváření znovupoužitelných komponent pro zjednodušení vývoje aplikace, atd. Při tvorbě těchto modelů často vznikají nové nápady na vylepšení aplikace. Tyto modely mohou být také součástí smlouvy, aby se zabránilo případným neshodám při dodání aplikace zákazníkovi.

Pro tvorbu drátových modelů jsem použil nástroj Miro. Vytvořil jsem si šablony webového prohlížeče a poté pomocí elektronické tužky jsem do těchto šablon kreslil. Drátové modely jsem navrhoval na základě již vytvořeného aplikačního rozhraní a zároveň jsem se inspiroval jinými nástroji (Xplico, NetworkMiner, ...). Chtěl jsem vytvořit nástroj, který bude zobrazovat data ze zachycené komunikace a zároveň uživateli poskytnut možnost agregace vybraných dat a poskytnut uživateli agregované hodnoty ze všech právě zobrazených dat na stránce v podobě grafů.

Abych ušetřil čas při vytváření drátových modelů, rozhodl jsem se je vytvořit genericky. Namísto vytváření drátového modelu pro každou stránku jsem vytvořil generickou verzi pro stránku zobrazující konkrétní zprávu, kolekci všech zpráv a další informace o zobrazené kolekci (grafy, agregovaná data a podezřelá slova v obsahu zprávy).

Na obrázku 3.5 je konečná verze drátového modelu pro zobrazení kolekce zpráv. Obsahuje tabulkou, v níž budou data zobrazena společně s možností vyhledávání. Pod tabulkou se nachází otevíratelné panely, obsahující dodatečnou funkcionality a informace nad kolekcí.

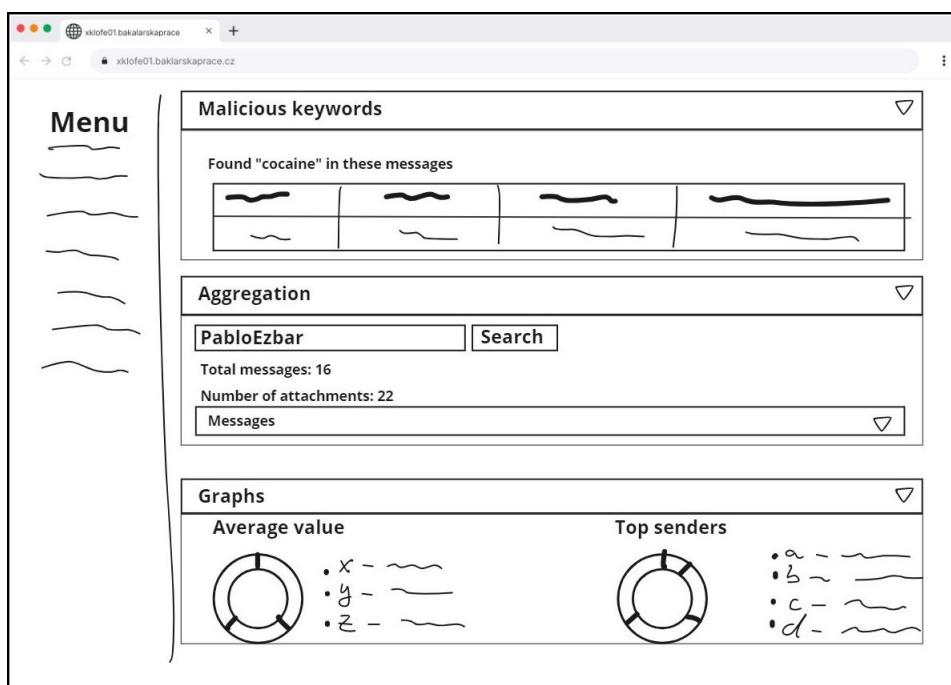


Obrázek 3.5: Drátový model vizualizující rozložení stránky zobrazující kolekci zachycených zpráv, včetně možnosti agregace dat, vyhledávání a stránkování.

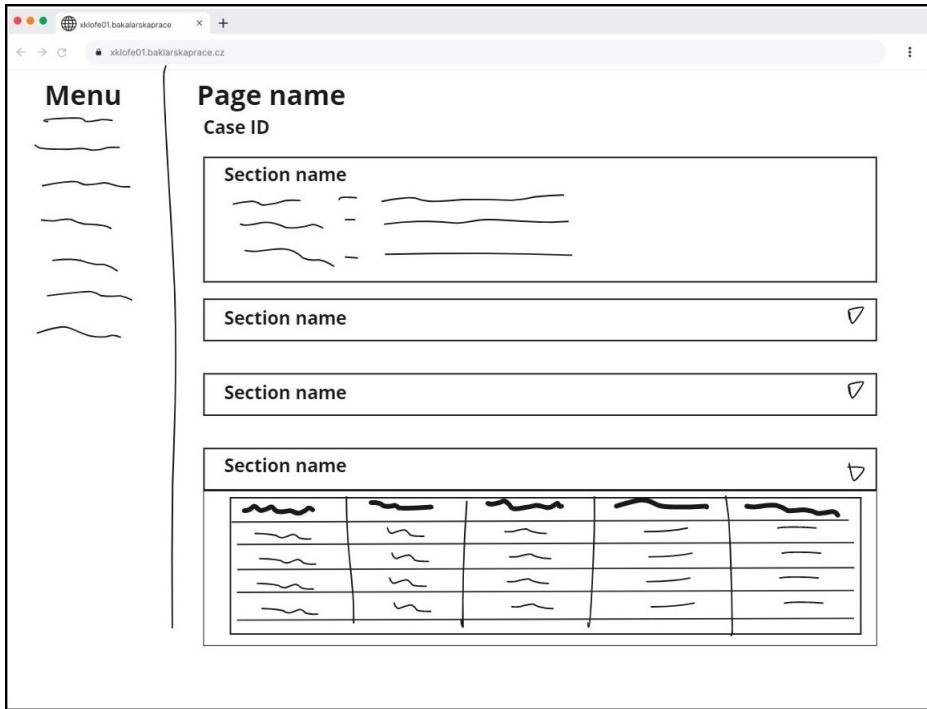
Otevřené panely lze vidět na obrázku 3.6. V panelu s názvem „*Malicious keywords*“ se nachází seznam zpráv, které ve svém obsahu mají podezřelá slova. Tato funkciálnita může vyšetřovati nejen ušetřit čas při prohledávání zpráv, ale zejména ho upozorní na existenci zprávy s podezřelým obsahem.

Další panel jenž nese nadpis „*Aggregation*“ slouží k agregaci dat podle odesílatele. Při zadání jména do pole pro vkládání textu se agregují informace o daném uživateli z kolekce právě zobrazených zpráv. Takovými informacemi jsou například celkový počet zpráv, které uživatel napsal, odeslané přílohy, adresáti atd. Poslední panel s nadpisem „*Graphs*“ obsahuje grafy vytvořené nad právě zobrazenou kolekcí.

Poslední drátový model 3.7, který zde zmíním je model pro zobrazení konkrétní zprávy. Na úvod stránky jsem vložil základní informace o zprávě a poté jsem jednotlivé části rozdělil a vložil do otevíracích panelů. Takto se uživatel může jednoduše zanorit a nezobrazuje se mu informace, které nechce vidět. Konkrétním příkladem je například obsah emailové zprávy, který může být značně rozsáhlý a zabírat tak poměrně velkou část celé stránky.



Obrázek 3.6: Drátový model reprezentující rozložení stránky zobrazující kolekci zachycených zpráv s otevřenými záložkami k zobrazení grafů, agregaci dat konkrétního uživatele a zpráv obsahujících podezřelé slova.



Obrázek 3.7: Drátový model reprezentující rozložení stránky zobrazující detailní informace konkrétní zprávy v otevíratelných panelech.

3.3.2 Grafický návrh

Poté, co jsem vytvořil drátové modely, jsem si vytvořil grafické modely jednotlivých stránek pomocí nástroje Figma⁹. Stáhl jsem si předlohu prohlížeče a začal tvořit pro každou stránku samostatný model, který reprezentoval konečný vzhled stránky.

K urychlení práce jsem si vytvářel komponenty, ze kterých jsem tvořil instance a ty vkládal do jednotlivých stránek. To vytvořilo časovou úsporu, jelikož kdykoliv jsem potřeboval některou z komponent změnit, stačilo změnit pouze rodičovskou neboli referenční a všechny její instance se automaticky změnily také. Podle těchto modelů jsem následně programoval aplikaci. Přestože jsem však vytvářel modely tak aby vizualizovali výslednou podobu aplikace, při její implementaci jsem se občas od modelů odchylil, důvodem bylo zjednodušení implementace.

Na obrázku 3.8 můžete vidět UI model, podle kterého jsem vytvářel stránku pro zobrazení detailních informací o zachycené komunikaci elektronické pošty nad protokolem SMTP. Informace jsou členěny do sekcí z důvodu lepší orientace uživatele na stránce - obálka („Envelope“), zpráva („Email“) a přílohy („Attachments“). Obrázek 3.9 vizualizuje návrh stránky pro zobrazení kolekce zachycených zpráv na vrstvě L7. Na místo konkrétních hodnot jsem při vytváření těchto modelů použil pouze popis reprezentující datový typ hodnoty, která bude na daném místě.

⁹Figma je vektorový grafický editor zaměřující se na UX a UI

SMTP - Client message
session Id: 3fa85f64-5717-4562-b3fc-2c963fb6afab

Mail path: <string>
Timestamp: <int>

Envelope

Date ticks: <0>
Subject: <string>
Address from:
Sender: <string>
Reply to:
Addresses to:
CC:
BCC:
In reply to: <string>
Message id: <string>

Email

Headers:
Payload: <string>

Attachments

Search

Path	Filename	Content type	Content encoding
<string>	<string>	<string>	<string>
<string>	<string>	<string>	<string>
<string>	<string>	<string>	<string>
<string>	<string>	<string>	<string>

Obrázek 3.8: UI Model zobrazující stránku s detailními informacemi o zachycené elektronické zprávy nad protokolem SMTP.

L7 - Conversation statistics

AddressA	AddressB	PortA	PortB	Protocol L3	Protocol L4	Protocol L7
<string>	<string>	<int>	<int>	<enum>	<enum>	<enum>
<string>	<string>	<int>	<int>	<enum>	<enum>	<enum>
<string>	<string>	<int>	<int>	<enum>	<enum>	<enum>
<string>	<string>	<int>	<int>	<enum>	<enum>	<enum>
<string>	<string>	<int>	<int>	<enum>	<enum>	<enum>
<string>	<string>	<int>	<int>	<enum>	<enum>	<enum>

Total 85 items < 1 ... 4 5 6 7 8 ... 20 > 10 / page Go to

Obrázek 3.9: UI Model zobrazující stránku s kolekcí zachycené komunikace na vrstvě L7.

3.4 Tvorba webové aplikace

Následující část popisuje samotnou implementaci, jejichž úspěšný výsledek závisí na předešlé práci a to zejména na nastudované teorii, drátových a grafických modelech ale také na praktických zkušenostech nabitych v dosavadním studiu.

Implementaci jsem rozdělil do 3 částí. Sdílená část, která slouží k uchování modelů, které jsou společné jak pro serverovou, tak pro klientskou část. Serverová část, byla vytvořena k otestování komunikace s klientskou částí a zároveň jako zdroj dat. Jelikož se tato práce zabývá vizualizací dat a tedy klientské části, vytvořil jsem serverou část pouze tak, aby odpovídala

na dotazy a nedbal jsem zvýšené pozornosti na její rozšířitelnost v budoucnu. Poslední částí je klientská část. Tato část už se zabývá samotnou vizualizací dat. Bylo zapotřebí vytvořit jednotlivé stránky a znovupoužitelné komponenty, abych zbytečně nekopíroval stejný kód. Při vytváření komponent a stránek jsem použil knihovny, abych ušetřil čas psaním HTML a CSS kódu a mohl se věnovat funkcionalitě a zlepšení UX aplikace. Použité knihovny jsou detailně popsány v předešlé sekci 3.1.4.



Obrázek 3.10: Vizualizace struktury projektu, šipky ukazují závislost.

3.4.1 Serverová část

Serverová část byla vytvořena pouze k otestování správné funkčnosti komunikace v klientské části. Jelikož serverová část není hlavním tématem této práce, nebyl z tohoto důvodu kláden důraz na případnou rozšířitelnost v budoucnu. Napsaný kód tedy nemusí splňovat některá doporučení při psaní čistého kódu.

Serverovou část jsem vygeneroval z již vytvořeného návrhu aplikačního rozhraní pomocí nástroje Swagger. Tento nástroj vygeneruje jednoduchou kostru skládající se z kontrolerů zastávající komunikaci se serverem a objekty pro datový přenos. Vygenerovaný kód vyžaduje úpravy ke správné funkčnosti serveru, avšak dříve než jsem začal upravovat tento vygenerovaný kód jsem začal vytvářet testovací data jejichž tvorba je popsána níže v této subsekci.

Vygenerované kontrolery obsahují funkce, které nevrací žádná data. Z tohoto důvodu bylo nutné vygenerovaný kód upravit tak, aby jednotlivé funkce vraceły patřičná data. Dalším krokem byla úprava objektů pro datový přenos. Jelikož jsou modely použity i v rámci klientské části, rozhodl jsem se je vyjmout z vygenerovaného projektu a vytvořit pro ně nový projekt, na který jsem poté vytvořil závislost jak ze serverové, tak z klientské části.

Testovací data

Pro každý typ komunikace (POP server, POP client, HTTP Response, atd.) jsem vytvořil samostatnou statickou třídu, ve které jsem vytvořil kolekci v podobě seznamu neboli listu. V této kolekci jsem následně uchovával jak ručně vytvořená, tak i vygenerovaná data. Některá data jsem vytvořil ručně, abych mohl replikovat testovací příklad použitý k vytvoření případů užití ke zmapování a prozkoumání konkurenčních nástrojů (Xplico, NetworkMiner). V tomto případě jsem si stáhl .pcap soubor testovacího případu (více v sekci 2.4), který jsem si otevřel v nástroji Wireshark a ručně vytvářel jednotlivé objekty pro datový přenos z vyčtených dat.

Poté, co jsem měl vytvořená potřebná data k replikaci testovacího příkladu, jsem použil knihovnu Bogus. Pomocí této knihovny jsem rychle vytvořil více než 1000 falešných dat, která slouží k zaplnění aplikace daty a otestování implementované funkcionality aplikace jako je například filtrování, agregace dat, zobrazení grafů atd.

Generování falešných dat (viz výpis 3.1) s nástrojem Bogus probíhá tak, že pro každou třídní proměnnou se nastaví kolekce ze které se pro tuto proměnnou bude náhodně vybírat hodnota. Nástroj Bogus má rozsáhlý výběr dat, která lze použít avšak v mé případě bylo k otestování aplikace nutné aby se některá data opakovala (IP adresy, emailové adresy, ID relací, atd.). Z tohoto důvodu jsem vytvořil jednu statickou třídu, jejichž účelem je uchování mnou vytvořených kolekcí dat (IP adres, emailových adres, ID relací, atd.), které jsem využíval při tvorbě všech generovaných falešných dat pro jednotlivé objekty pro datový přenos.

¹ var PopServerMessage = new Faker<PopServerMessageDTO>()
² .RuleFor(o => o.SessionId, f => f.PickRandom(SharedDTOsSeeds.GuidArray))

```

3 .RuleFor(o => o.Reply, f => f.Random.Enum<PopServerMessageDTO.ReplyEnum>())
4 .RuleFor(o => o.Description, f => f.Commerce.ProductDescription())
5 .RuleFor(o => o.Payload, f => f.Lorem.Paragraphs(randomNum.Next(1, 5)))
6 .RuleFor(o => o.Envelope, f => SharedDTOsSeeds.EmailEnvelopeFaker.Generate(1)[0])
7 .RuleFor(o => o.Email, f => SharedDTOsSeeds.EmailMessageFaker.Generate(1)[0])
8 .RuleFor(o => o.MailPath, f => f.PickRandom(SharedDTOsSeeds.EmailPath))
9 .RuleFor(o => o.Timestamp, f => (int)DateTime.UtcNow
10 .Subtract(f.Date.Between(new DateTime(1990, 1, 1), new DateTime(2022, 1, 1)))
11 .TotalSeconds)
12 .RuleFor(o => o.Attachments, f => SharedDTOsSeeds.EmailAttachmentFaker.Generate(
13     randomNum.Next(0, 15)));
14 PopServerMessageListSeed.Items = PopServerMessage.Generate(NumberOfPopClientMessages);

```

Výpis 3.1: Ukázka tvorby falešných dat pro objekt sloužící k datovému přenosu „PopServerMessageDTO“ pomocí nástroje Bogus.

3.4.2 Klientská část

Vytvoření vizualizační aplikace, která umožní uživatelům snadno a rychle pracovat s daty je hlavním tématem této práce a při vývoji jsem kladl důraz na čitelnost a udržitelnost kódu, abych zajistil snadnou rozšířitelnost aplikace v budoucnu. Pro dosažení tohoto cíle jsem vytvořil dva projekty: klientskou Blazor aplikaci a komunikační knihovnu. V komunikační knihovně je obsažena veškerá komunikační logika mezi aplikačním rozhraním na serveru a Blazor aplikací, zatímco klientská aplikace, se kterou interaguje uživatel se zaměřuje pouze na vizualizaci dat.

Oddělením komunikační logiky od klientské aplikace jsem zajistil, že každý projekt má pouze jednu odpovědnost a tedy jeden důvod k existenci. Klientská aplikace tak obsahuje pouze kód související s vizualizací dat a uživatelským rozhraním. Kromě výhod jako znovupoužitelnost, testovatelnost a udržitelnost, tento přístup také zajišťuje snadnější údržbu a modulárnost kódu, což v budoucnu umožní jednodušší rozšíření aplikace.

Komunikační knihovna

Komunikační knihovna zajišťuje veškerou komunikaci se serverem a je tedy komunikačním prostředníkem mezi klientskou aplikací a aplikačním rozhraním na serveru. Díky tomu, že veškerá komunikační logika je oddělena do samostatného projektu, lze jednoduše modifikovat a rozširovat kód bez zásadních změn na klientskou aplikaci. Komunikační knihovna vytváří abstrakci a chrání klientskou aplikaci před případnými změnami. Při vytváření komunikační knihovny jsem postupoval následovně:

- Vygenerování aplikačního klienta — aplikačního klienta jsem vygeneroval pomocí nástroje NSwagStudio¹⁰. Aplikační klient slouží ke komunikaci se serverem skrze jednotlivé koncové body aplikačního rozhraní. Vygenerovaný klient obsahuje funkci pro každý koncový bod a interně provádí serializaci a deserializaci dat. Díky tomu je komunikace se serverem velmi jednoduchá, stačí pouze zavolat patřičnou funkci a ta provede potřebné požadavky na serveru.
- Vytvoření fasád — vytvořením fasád mezi generovaným aplikačním klientem a samostatnou klientskou aplikací jsem mezi nimi zjednodušil komunikaci a skryl složitost aplikačního klienta před klientskou aplikací. Generovaný kód nemusí být lehce čitelný či modifikovatelný a z tohoto důvodu je vhodné vytvořit pomocí fasády rozhraní vyšší úrovňě. Dále

¹⁰NSwagStudio — open-source nástroj pro generování aplikačních klientů a serverového kódu na základě REST aplikačního rozhraní využívající Swagger (OpenAPI) specifikaci. Nástroj NSwagStudio má grafické rozhraní a umožňuje tak jednoduchou interakci s uživatelem. NSwagStudio umí generovat kód nejen v jazyce C# ale i v jazyce typescript.

vytvořením fasád, se vytvořila vrstva abstrakce, která usnadňuje případné změny v buďoucnu, které tak nemusí ovlivnit klientskou aplikaci. Pokud by byl aplikační klient nějakým způsobem modifikován či zcela nahrazen, jediným místem v kódu, na které by měla případná změna vliv, by byla právě ona fasáda. Pokud by tedy rozhraní fasády zůstalo po modifikaci stejná, klientská aplikace by nebyla třeba jakkoliv modifikovat.

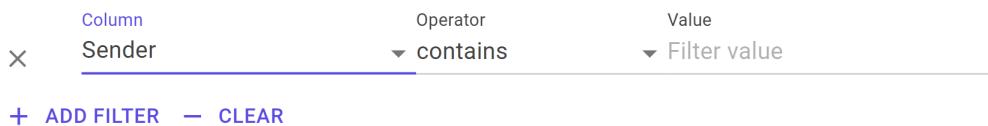
Blazor aplikace

Blazor aplikaci jsem vyvíjel pomocí grafických návrhů, které jsem měl již vytvořené. Nicméně přestože jsem strávil poměrně dost času tvorbou těchto návrhů, během vývoje jsem přicházel s nápady jak aplikaci vylepšit. Nyní popíšu jak jsem postupoval při tvorbě stránek a ukážu některé změny oproti grafickému návrhu.

Prvotním krokem při vytváření klientské aplikace byla instalace knihoven MudBlazor a ApexCharts viz 3.1. Instalace těchto knihoven proběhla velice snadno díky řádně zpracované dokumentaci na oficiálních stránkách. Na knihovně MudBlazor je postavena celá aplikace a z toho důvodu byla její instalace nutná hned na začátku. Dále jsem pokračoval vytvářením UI jednotlivých stránek a následně jejich funkcionality. Pro funkci jenž se opakovala na více stránkách a bylo to možné, jsem vytvořil samostatnou komponentu, abych nevytvářel stejný kód na více místech.

Díky již vytvořeným grafickým modelům a knihovnou s připravenými UI komponentami byla tvorba stránek velice jednoduchá. Nejprve jsem začal se stránkou pro výběr případů. Pokud uživatel nemá vybraný žádný případ, jeho přístup v aplikaci je limitovaný pouze na hlavní stránku a stránku s kolekcí případů. Pokud chce uživatel přejít na stránku, která vyžaduje vybraný případ, je odkázán na stránku s kolekcí případů, na které se objeví informační hláška, která uživatele informuje o problému.

Stránky obsahující kolekce zachycených zpráv Následně jsem začal vytvářet stránky obsahující kolekce zpráv. Všechny tyto stránky obsahují tabulku, jenž zobrazuje data ze serveru a nějakou formu agregovaných dat, nejčastěji v podobě grafů. V každé tabulce lze filtrovat právě zobrazená data díky jednoduchému rozhraní, které vizualizuje obrázek 3.11.



Obrázek 3.11: Rozhraní pro specifikování filtrování nad daty zobrazené v tabulce.

Ve většině tabulek v rámci celé aplikace lze vytvořit libovolný počet filtrů, přičemž si uživatel vybere sloupec nad kterým se bude filtr provádět, hledanou hodnotu a operátor který závisí na datovém typu sloupce (string, int, date, ...) a dle datového typu nabídne patřičné operátory. Tuto funkci vizualizuje obrázek 3.14, na kterém můžete vidět možnost výběru operátorů dle datového typu hodnoty ve vybraném sloupci. Z obrázku 3.14 lze vidět, že uživatel má rozsáhlé možnosti filtrování dat a tedy zobrazení pro uživatele relevantních dat.

string	int	date
contains	=	is
not contains	!=	is not
equals	>	is after
not equals	>=	is on or after
starts with	<	is before
ends with	<=	is on or before
is empty	is empty	is empty
is not empty	is not empty	is not empty

Obrázek 3.12: Kolekce operátoru, při vytváření nového filtru nad tabulkou, ze kterých si uživatel vybere operátor k filtrování dat v tabulce. Dle vybraného sloupce a datového typu hodnot v něm se uživateli zobrazí určitá kolekce operátorů.

Pro každou kolekci jsem vytvořil graf nad daty, jenž jsem považoval za vhodný ke grafickému zobrazení. Nejčastěji se jednalo o maxima, minima, nejčastější hodnoty, zobrazení všech nebo konkrétních zpráv v čase atd. Ukázku zobrazení zpráv v čase můžete vidět na snímku 3.14. Jedná se o komponentu z knihovny ApexCharts zobrazující zprávy v čase uživatelem vybraných odesílatelů. Graf lze stáhnout v podobě *.svg*, *.png* či *.csv*. Celkově jsem si uvědomoval, že grafické zobrazení dat může být pro uživatele mnohem srozumitelnější a názornější než pouhé zobrazení čísel. Proto jsem se snažil vytvořit grafy, které budou plnit svůj účel a zároveň budou esteticky příjemné a snadno čitelné.



Obrázek 3.13: Graf zobrazující zachycené zprávy v čase, uživatelem vybraných odesílatelů.

Na některých stránkách aplikace umožňuji uživatelům provádět agregaci dat, tedy zobrazit zprávy, které obsahují určitá podezřelá klíčová slova nebo informace o konkrétním odesílateli. Pro tyto účely jsem v aplikaci nastavil výchozí klíčová slova, která jsem považoval za nejvhodnější pro detekci problematických zpráv. Jedná se například o názvy nelegálních drog, zbraní,

slova týkající se násilí atd. Pokud některá ze zpráv obsahuje některé ze slov, zobrazí se v nové tabulce.

Pokud aplikace zachytí zprávu obsahující některé z těchto klíčových slov, zobrazí se uživateli v nové tabulce, aby mohl být upozorněn na potenciálně podezřelou zprávu. Uživatel má také možnost přidávat svá vlastní klíčová slova, která by mohla být pro něj relevantní a odstraňovat jak svá vlastní, tak i výchozí klíčová slova.

Tato funkce je pro vyšetřovatele velmi užitečná, protože umožňuje rychle identifikovat potenciálně podezřelé zprávy bez nutnosti procházet ručně celým seznamem zachycených zpráv. To může být velmi časově náročné, zejména v případech, kdy je k dispozici velké množství dat. Díky této funkci se tak proces vyšetřování může urychlit a zefektivnit.

Malicious key words			
Type malicious word			
Frozen			
	Hit		bacon
	Frozen		
Sent on	Malicious words	Subject	Message ID
16.03.2020 0:25:17	Hit, Frozen	Handcrafted Frozen Chips	ffbdd2f9-519a-464a-aab0-7a2ef1b52e85
12.12.1997 12:45:33	bacon	Handmade Wooden Bacon	79eb3825-208e-4fb8-9f6a-74f7390a368f
15.08.2018 22:15:13	Hit, Frozen	Intelligent Frozen Sausages	4c5e0418-61fe-4588-b6fb-b1f5142327d9

Obrázek 3.14: Graf zobrazující zachycené zprávy v čase, uživatelem vybraných odesíatelů.

Další funkcí aplikace je agregování dat pomocí odesílatele. Uživatel může do vyhledávacího pole zadat jméno odesílatele a aplikace mu následně zobrazí veškeré informace o tomto odesílateli, například počet odeslaných zpráv, seznam všech příjemců, seznam všech zaslaných příloh a další relevantní informace.

Tato funkce umožňuje uživatelům rychle identifikovat informace související s konkrétním odesílatelem a zjistit, jaké zprávy byly odeslány, kdy byly odeslány a komu byly zprávy poslány. To je užitečné zejména v situacích, kdy se vyšetřovatelé snaží rekonstruovat konkrétní události nebo sledovat komunikaci mezi určitými osobami.

Stránky zobrazující detail konkrétní zprávy Na stránkách, které zobrazují detail konkrétní zprávy, není k dispozici žádná funkčnost. Tyto stránky jsou navrženy jako statické a zobrazují pouze informace o dané zprávě, jako je například datum odeslání, odesílatel, příjemce, předmět a text zprávy.

Nicméně, abych zajistil co nejlepší UX, využil jsem rozšiřujících panelů k zobrazení dodatečných informací. Rozšiřující panely umožňují uživatelům zobrazit další informace o zprávě, které nejsou na první pohled viditelné, jako je text nebo hlavičky emailové zprávy. Rozšiřující panely také pomáhají udržovat stránku čistou a uživatelsky přívětivou.

Kapitola 4

Testování

Testování softwaru je klíčovým procesem v celém vývojovém cyklu, zajišťující kvalitu a spolehlivost výsledného produktu. Cílem testování je zajistit, aby software splňuje definované požadavky. Testování je proces, který by měl být prováděn průběžně během celého vývojového cyklu. K zajištění nejlepších možných výsledků je nutné použít automatizované testy, které lze libovolně spouštět v průběhu vývoje. Je vhodné tyto testy spouštět v pravidelných časových intervalech, aby se případná chyba identifikovala co nejdříve a také byla co nejdříve opravena. Při testování je důležité znát a rozumět případům užití aplikace a otestovat různé možnosti, které by mohl uživatel při používání aplikace vyzkoušet. Testování webové aplikace je provedeno ze dvou důvodů:

- Odhalit funkcionální a grafické nedostatky a chyby.
- Otestovat UX aplikace.

V této kapitole popíšu dva druhy testování - testování funkcionality a testování použitelnosti. Na závěr této kapitoly popíšu testování mnou vytvořené aplikace s nezávislým uživatelem. Testování probíhalo tak, že tento uživatel plnil v aplikaci sérií dílčích úloh a já jsem zaznamenával dobu trvání a další hodnoty. Následně zmíním testování aplikace na specifickém případu užití, na kterém jsem testoval i nástroj Xplico a NetworkMiner při získávání znalostí k vytvoření mé aplikace a na závěr vizualizuji případ užití, který demonstruje funkctionalitu aplikace.

4.1 Testování funkcionality

Jedním z nejdůležitějších typů testování je testování funkcionality. Toto testování ověřuje, zda software plní požadavky a specifikace stanovené při návrhu aplikace. Testování funkcionality jsem testoval pouze manuálně. To mi umožnilo ověřit správnost a funkčnost jednotlivých prvků aplikace. Testování jsem prováděl postupně a opakovaně během vývoje a následně po dokončení, abych zajistil, že aplikace bude co nejvíce bezchybná.

Důraz jsem kladl na ověření správné funkčnosti všech interaktivních prvků aplikace, jako jsou tlačítka, filtrování dat dle data nebo podle identifikátoru relace, pole pro vkládání textu a zobrazených grafů. Konkrétně jsem ověřoval, zda tlačítka reagují a spouští požadovanou reakci, zda aplikace zobrazuje pouze data dle uživatelem zadaných filtrů, zda aplikace správně reaguje na vstupy uživatele a zda jsou všechny interaktivní prvky dostupné a funkční. Dále jsem počítal zobrazené hodnoty v tabulkách, abych ověřil správnost zobrazených grafů. V rámci testování interaktivních prvků jsem se také zaměřil na ověření rychlosti a odezvy aplikace.

Celou aplikaci jsem manuálně otestoval a zkontoval, že implementovaná funkctionalita funguje. Testování je možné automatizovat automatizovanými testy, které lze vytvářet pomocí nástrojů, jako je například Selenium, Katalon Studio, Cypress. Automatizované testy, lze opakovatelně spouštět a testovat tak aplikaci v pravidelných intervalech.

Během testování jsem identifikoval několik problémů, obzvlášť při zobrazování grafů. Při jejich implementaci se mi stalo, že jsem některá data špatně počítal a zobrazoval tak nepravidlivé údaje. Tento problém jsem řešil například při zobrazení všech zachycených zpráv v čase a zobrazoval tak jiný počet zachycených zpráv v daném časovém intervalu než ve skutečnosti byl. Tuto chybu, jsem úspěšně vyřešil stejně tak jako ostatní.

4.2 Testování použitelnosti

Internet se stal součástí každodenního života a lidé používají různé webové aplikace, ať už k práci či zábavě několikrát denně. Z tohoto důvodu je důležité zajistit, aby webové stránky byly uživatelsky přívětivé. Testování použitelnosti je proces při kterém se měří a identifikují slabiny rozhraní webové stránky za účelem zlepšení uživatelské přívětivosti.

Pokud jsou stránky uživatelsky nepřívětivé a uživatel se na stránce neorientuje, může to znamenat nedůvěru v produkt či službu, kterou stránky reprezentují. To může mít za následek, že uživatel opustí stránku a hledaný produkt či službu se pokusí najít jinde. Z tohoto hlediska je uživatelská přívětivost klíčovým faktorem, obzvlášť pro internetové obchody.

Existují různé techniky jak testovat použitelnost webových stránek. Mezi ty nejpoužívanější patří:

- Analýza úkolů — proces, při kterém se vytvoří různé úkoly či případy užití, které by uživatel typicky prováděl. Poté se tyto úkoly předají subjektu a analyzují se uživatelem prováděné kroky k jejich dokončení. Tato technika byla použita při testování použitelnosti mnou vytvořené Blazor aplikace.
- Sledování chování uživatele — sledování chování uživatele ze zaměřuje na chování uživatele na stránce, např. průměrná doba stráveného času na stránce, geografické poloze uživatelů, bounce rate¹, atd. Existují nástroje, jenž tato data zaznamenávají automatizovaně. Mezi tyto nástroje patří například Google Analytics nebo Hotjar.
- A/B testování — A/B testování je založeno na kvantitativním testování, vyžaduje dvě či více verzí webových stránek a určitému množství testovacích uživatelů. Testovací uživatelé se rozdělí na skupiny dle počtu verzí webových stránek a následně se zaznamenává uživatelská reakce a použitelnost na každou z verzí.

Každá z těchto technik má silné a slabé stránky a výběr správné závisí na okolnostech a zdrojích. Kombinací uvedených či jiných metod lze dosáhnout lepších výsledků než použitím pouze jedné techniky.

4.3 Testování aplikace

V této části popíšu, jak jsem testoval aplikaci se soutěžním příkladem, který byl také použit k otestování nástroje Xplico a NetworkMiner v předešlé kapitole 2.4. Dále také popíšu jak jsem aplikaci otestoval technikou analýzy úloh, kdy jsem vzal nezávislého uživatele, který neměl s aplikací předešlou zkušenosť a analyzoval, jak s aplikací pracuje. Na závěr této sekce uvedu grafický případ užití, který slouží k demonstraci funkcionality aplikace.

¹Bounce rate je ukazatel, který udává procento návštěvníků, kteří opustili webovou stránku po prohlédnutí pouze jedné stránky. Vyšší bounce rate obvykle naznačuje, že uživatelé nenachází na stránce to, co hledají, nebo že je design stránky nedostatečně přehledný.

4.3.1 Specifický případ užití

Pro tento případ užití jsem využil stejný specifický případ jako při testování nástroje Xplico a NetworkMiner viz kapitola 2.4. V tabulce 4.1 je zaznamenáno jak lze jednotlivé úlohy vyřešit pomocí mnou vytvořené Blazor aplikace.

1	Jaká emailová adresa náleží Anně?
	Zobrazení kolekce zpráv (L7 -> SMTP -> Client messages). U jména Anny lze vidět emailovou adresu a tedy odpověď: „ <i>sneakyg33k@aol.com</i> “
2	Jaká je emailová adresa milence Anny?
	Zobrazení kolekce zpráv (L7 -> SMTP -> Client messages). Analyzováním jednotlivých zpráv zaslaných Annou lze určit odpověď: „ <i>mistersecretx@aol.com</i> “
3	Jaké dva předměty řekla Anna svému milenci aby přinesl?
	Zobrazení kolekce zpráv (L7 -> SMTP -> Client messages). Vyhledání zprávy pro adresáta „ <i>mistersecretx@aol.com</i> “. V detailu zprávy v sekci Email -> Payload lze v obsahu zprávy vyčíst odpověď: <i>fake passport and a bathing</i>
4	Kolik příloh odeslala Anna celkem?
	Zobrazení kolekce zpráv (L7 -> SMTP -> Client messages). Otevřením otevíracího panelu s názvem „ <i>Aggregation</i> “ a napsáním jména Anny do vstupního pole se zobrazí agregovaná data ze všech emailových zpráv, zaslaných Annou včetně počtu příloh.
5	Název přílohy, kterou Anna poslala svému milenci?
	Zobrazení kolekce zpráv (L7 -> SMTP -> Client messages). Vyhledání zprávy pro adresáta „ <i>mistersecretx@aol.com</i> “. V detailu zprávy v sekci Email -> Attachments lze v kolekci příloh zprávy vyčíst odpověď: „ <i>secretrendezvous.docx</i> “

Tabulka 4.1: Specifický případ užití mnou vytvořené Blazor aplikace.

4.3.2 Analýza úkolů

Při testování technikou analýzy úkolů jsem použil dva případy užití. Případy užití vykonával můj spolužák (dále jen uživatel). Pro uživatele jsem připravil sérii úkolů, které poté vypracoval. Práci dílčích úkolů jsem měřil stopkami a následně jsem zaznamenal, jak se uživateli s aplikací pracuje. Uživatel neměl žádnou předešlou zkušenosť s mojí aplikací a při konání jednotlivých úkolů aplikaci viděl poprvé. Při analyzování uživatele vykonávajícího jednotlivé úkony jsem sledoval následující faktory:

- Rychlosť, s jakou uživatel plnil úkoly.
- Schopnost uživatele najít požadované informace.
- Způsob, jakým uživatel interagoval s webovým rozhraním.

Do následující tabulky 4.2 jsem zaznamenal výsledky z testování na uživateli. Jednotlivé vlastnosti jsem ohodnotil číslicí 0 až 10 s tím, že číslo 0 znamená zápornou hodnotu a 10 kladnou hodnotu. Jednotlivé úkoly, které uživatel vypracovával:

1. Kolik zachycených komunikací má případ s názvem „*Network puzzle*“

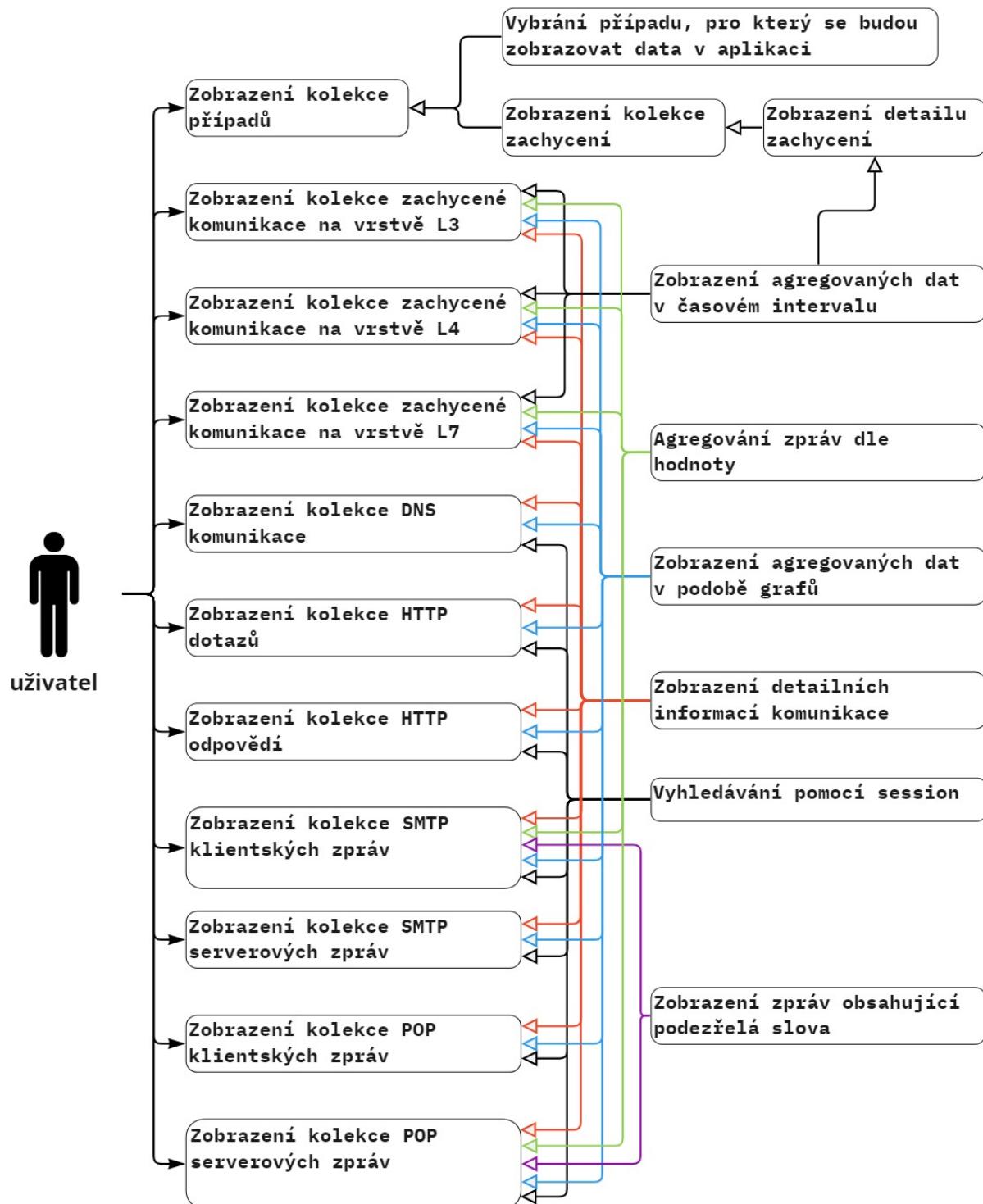
2. Jaký je nepoužívanější protokol na vrstvě L3?
3. Z jaké adresy se odeslalo nejvíce zpráv v zachycené komunikaci na vrstvě L4. Kolik zpráv z této adresy bylo odesláno a na jaké porty?
4. Kolik je dotazů a kolik odpovědí v DNS komunikaci?
5. Jaký je nejčastější reply code?
6. Kolik zpráv je od 6. 2. 1995 do 2. 4. 1999?
7. kolik dotazu je na url adresu wikipedia?
8. jaký je nejčastější typ dotazu?
9. Kolik příloh typu .mp4 bylo odesláno při http dotazu?
10. Kolik zpráv mělo odpověď 4xx mezi datem 20. 5. 1987 a 28. 4. 1991?
11. Kolik zpráv má uživatel „Ann Dercover“?
12. Co je obsahem zprávy uživatele „Ann Dercover“ příjemci „mistersecretx@aol.com“ a kolik zpráva obsahuje přílohy
13. Kdo poslal nejvíce zpráv?
14. Kolik bylo celkem odesláno příloh a jakého typu bylo nejvíce odesláno?
15. Obsahuje některá ze zpráv v předmětu nebo obsahu zprávy slovo kokain?
16. Kolik zpráv odeslal uživatel se jménem „Zola97“ od roku 2002?

Úkol	Čas na dokončení	Chyby	Snadnost použití (1-10)
1.	28s	0	8
2.	13s	0	10
3.	1min 14s	0	8
4.	22s	0	10
5.	44s	0	9
6.	1min 41s	0	10
7.	1min 53s	0	9
8.	22s	0	10
9.	33s	0	9
10.	1min 5s	0	7
11.	3min 32s	0	6
12.	1min 3s	0	7
13.	14s	0	10
14.	7s	0	10
15.	2min 16s	0	6
16.	59s	0	9

Tabulka 4.2: Výsledky analýzy použitelnosti.

4.3.3 Případ užití aplikace

Následující diagram užití slouží k vizuální reprezentaci různých scénářů, které lze provádět s tímto nástrojem. Tento diagram zahrnuje všechny funkcionality mnou vytvořeného nástroje k vizualizaci forenzně sítových dat.



Obrázek 4.1: Případ užití, demonstrující funkcionality mnou vytvořené Blazor aplikace.

Kapitola 5

Závěr

Cílem práce bylo vytvořit uživatelsky přívětivou webovou aplikaci, která bude sloužit vyšetřovatelům jako nástroj k vizualizaci zachycených forenzně sítových dat. Pomocí tohoto nástroje tak budou moci vyšetřovatelé rychleji a efektivněji prošetřovat činy v rozporu se zákonem a identifikovat tak pachatele.

Nejprve bylo nutné si nastudovat forenzní sítovou analýzu jako takovou k porozumění problematiky. Po nastudování potřebných informací jsem analyzoval již existující nástroje. Analyzování těchto nástrojů jsem prováděl přímo v aplikaci, případně ze snímků či videí. Při analýze jsem se zaměřil na UX a UI nástrojů tak abych poté navrhl aplikaci, která by byla uživatelsky přívětivá a prospěšná v oblasti vizualizace forenzně sítových dat. Důkladnou analýzu jsem provedl u nástroje Xplico a NetworkMiner. U těchto nástrojů jsem vytvořil případ užití demonstруjící jejich funkcionality a také jsem tyto nástroje podrobil testovacímu případu.

Poté jsem začal s návrhem mé aplikace. Nejprve jsem navrhl definici aplikačního rozhraní, abych definoval jak bude komunikovat webová aplikace se serverem společně s objekty pro datový přenos. Dalším krokem bylo navrhnout uživatelské prostředí. Nejdříve jsem začal vytvářet drátové modely, abych navrhl funkcionality a optimální rozvržení komponent na jednotlivých stránkách. Následně jsem začal vytvářet grafické modely, které už měli reprezentovat výsledný vzhled aplikace.

Nedílnou součástí práce byla implementace. Po dokončení všech návrhů jsem začal vytvářet serverovou část, která komunikuje s webovou aplikací a slouží k přenesu dat. Hlavním důvodem tvorby serverové části je otestování správné komunikace ze strany webové aplikace. Jelikož hlavním tématem této práce je především webová aplikace vizualizující forenzně sítová data, nevěnoval jsem kvalitě kódu na serverové části tolik pozornosti. Dále bylo potřeba vytvořit data, která by zaplnila aplikaci a bylo možné demonstrovat implementovanou funkcionality. Některá data jsem vytvořil ručně abych demonstroval testovací případ použitý při analýze nástroje Xplico a NetworkMiner. Zbylá data jsem vygeneroval pomocí knihovny Bogus. Jakmile byla serverová část funkční a server vracel skrze aplikační rozhraní data, začal jsem vytvářet webovou aplikaci. K tomu jsem použil framework od společnosti Microsoft Blazor, který překládá C# kód do WebAssembly, které jede ve webovém prohlížeči. Výsledná webová aplikace byla navržena a modularizována tak, aby případná rozšířitelnost byla jednoduchou záležitostí. Aplikace také byla rádně otestována mnou samotným a také proběhlo testování s uživatelem, který s aplikací neměl předchozí zkušenost.

Seznam pojmu

CAM Content Addressable Memory. 9

DDoS Distributed Denial of Service. 7

DHCP Dynamic Host Configuration Protocol. 3, 4, 7, 10

DNS Domain Name System. 3, 4, 7, 10, 26

HTML Hypertext Markup Language. 22

HTTP Hypertext Transfer Protocol. 12, 26, 32

IDS Intrusion Detection System. 12

IMAP Internet Message Access Protocol. 12

IoT Internet of Things. 5

IP Internet Protocol. 6, 10, 12, 13, 18, 26

IPS Intrusion Prevention System. 12

IRC Internet Relay Chat. 12

LAN Local Area Network. 9

MAC Media Access Control. 9, 10, 13

MSN Microsoft Notification Protocol. 12

NFAT Network Forensic Analysis Tools. 11

NIDS Network Intrusion Detection Systems. 7

OSPF Open Shortest Path First. 10

OWASP Open Web Application Security Project. 14

POP Internet Message Access Protocol. 12, 26, 32

QoS Quality of Service. 13

RIP Routing Information Protocol. 10

RTP Real-time Transport Protocol. 12

SDN Software-defined networking. 5

SIP Session Initiation Protocol. 12

SMTP Simple Mail Transfer Protocol. 12, 26, 30, 31

SPA Single Page Application. 22, 23

SQL Structured query language. 7

TCP Transmission Control Protocol. 12

UDP User Datagram Protocol. 12

UI User Interface. 22, 25, 30, 31, 34, 42

UX User Experience. 22, 27, 30, 32, 36, 37, 42

XML Extensible Markup Language. 12

Literatura

- [1] *Libpcap Library* [online]. 2007 [cit. 2022-12-18]. Dostupné z: <https://www.sciencedirect.com/topics/computer-science/libpcap-library>.
- [2] *Network forensics tools survey and taxonomy* [online]. 2021 [cit. 2022-12-20]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=231003.
- [3] *IPScan* [online]. 2022 [cit. 2022-12-18]. Dostupné z: <https://nmap.org/https://angryip.org/about/>.
- [4] *Metasploit Framework* [online]. 2022 [cit. 2022-12-20]. Dostupné z: <https://docs.rapid7.com/metasploit/msf-overview>.
- [5] *Metasploit Pro* [online]. 2022 [cit. 2022-12-20]. Dostupné z: <https://www.rapid7.com/products/metasploit/download/editions/>.
- [6] *Nessus* [online]. 2022 [cit. 2022-12-20]. Dostupné z: <https://www.tenable.com/products/nessus>.
- [7] *NetworkMiner* [online]. 2022 [cit. 2022-12-18]. Dostupné z: <https://www.netresec.com/?page=NetworkMiner>.
- [8] *Xplico* [online]. 2022 [cit. 2022-12-02]. Dostupné z: <https://www.xplico.org/>.
- [9] *MDN Web Docs* [online]. 2023 [cit. 2023-3-3]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>.
- [10] *Microsoft Blazor* [online]. 2023 [cit. 2023-3-6]. Dostupné z: <https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-7.0>.
- [11] *Microsoft SignalR* [online]. 2023 [cit. 2023-3-6]. Dostupné z: <https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-7.0>.
- [12] *WebAssembly* [online]. 2023 [cit. 2023-3-6]. Dostupné z: <https://webassembly.org/>.
- [13] DAVIDOFF, S. a HAM, J. *Network Forensics Tracking Hackers through Cyberspace*. Prentice hall, 2012. ISBN 978-0132564717.
- [14] GEBHARDT, T. a REISER, H. P. *Network Forensics for Cloud Computing*. Springer, Berlin, Heidelberg, 2013. ISBN 978-3-642-38540-7.
- [15] IBM. *IBMNetDetector* [online]. 2022 [cit. 2022-12-02]. Dostupné z: <https://www-50.ibm.com/partnerworld/gsd/solutiondetails.do?solution=14447&lc=en&stateCd=P&tab=2>.
- [16] JASWAL, N. *Hands-On Network Forensics*. Packt, 2019. ISBN 978-1789344523.
- [17] JOSHI, R. a PILLI, E. S. *Fundamentals of Network Forensics*. Springer, 2016. ISBN 978-1-4471-7297-0.

- [18] KUROSE, J. F. a ROSS, K. W. *Computer Networking: A Top-Down Approach eighth edition*. Pearson, 2022. ISBN 9356061319.
- [19] MATOUŠEK, P. *Síťové aplikace a jejich architektura*. Nakladatelství VUTIUM, 2014. ISBN 978-80-214-3766-1.
- [20] MIKOWSKI, M. S. a POWELL, J. C. *Single Page Web Applications: JavaScript end-to-end*. Manning, 2013. ISBN 978-1617290756.
- [21] NIKSUN. *Niksun* [online]. 2022 [cit. 2022-12-02]. Dostupné z:
<https://www.niksun.com/netdetector.php>.