FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Automatické rozpoznávání síťových zařízení a jejich závislostí
Student:	Josef Koumar
Vedoucí:	Ing. Tomáš Čejka, Ph.D.
Studijní program:	Informatika
Studijní obor:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Seznamte se s problematikou monitorování síťového provozu pomocí tzv. síťových toků (IP Flows).
Seznamte se s open source systémem NEMEA [1,2] pro automatickou analýzu provozu a detekci bezpečnostních událostí.
Navrhněte NEMEA modul pro analýzu rozšířených obousměrných síťových toků (biflow) pro rozpoznání serverů a jejich služeb, případně i dalších informací o zařízeních.
Implementujte navržený NEMEA modul, který bude detekovat servery a udržovat seznam jejich klientů. Výstup modulu, tzn. graf závislostí klientů na službách serverů, zkuste vizualizovat pomocí existujících nástrojů.
Vyvinutý modul otestujte pomocí reálného síťového provozu (dodá vedoucí práce), zaměřte se na vyhodnocení propustnosti modulu a jeho náročnosti na výpočetní a paměťové zdroje.

Seznam odborné literatury

[1] T. Cejka, et al.: "NEMEA: A Framework for Network Traffic Analysis," in *12th International Conference on Network and Service Management (CNSM 2016)*, Montreal, Canada, 2016.
[2] https://nemea.liberouter.org

prof. Ing. Pavel Tvrdík, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 11. prosince 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Automatické rozpoznávání síťových zařízení a jejich závislostí

Josef Koumar

Katedra počítačových systémů
Vedoucí práce: Ing. Tomáš Čejka, Ph.D.

19. května 2020

Poděkování

Děkuji svému vedoucímu Ing. Tomáši Čejkovi, Ph.D. za rady a vstřícnost. Dále děkuji své rodině a přátelům za podporu a pomoc při psaní této bakalářské práce. Zejména přátelům Marcelu Poláčkovi za odborné rady a Tereze Ausficírové za rady ohledně stylizace českého jazyka.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 19. května 2020 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2020 Josef Koumar. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Koumar, Josef. *Automatické rozpoznávání síťových zařízení a jejich závislostí.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato práce se zabývá návrhem a implementací PassiveAutodiscovery modulu, který získá informace o zařízeních a určí jejich role v síti. Je vytvořen pro existující síťový modulární systém NEMEA. Teoretická část práce obsahuje popis, jakým způsobem modul získává informace o zařízeních ze sítě a praktická část obsahuje návrh a implementaci vzniklého modulu. Výsledky testování potvrzují funkčnost a ukazují časovou a paměťovou náročnost celého modulu.

Uživatel díky PassiveAutodiscovery modulu získá základní informace o všech zařízeních komunikující na měřené síti, role zařízeních v této síti, závislosti mezi zařízeními a statistiky používání sítě.

Klíčová slova NEMEA, autodiscovery, detekce zařízeních, role zařízení v síti, servery a klienti, analýza síťového provozu, pasivní analýza, CESNET, python

Abstract

This thesis is about design and implementation of the PassiveAutodiscovery module, which receives information about devices and determines their role in network. Module is designed for an existing network modular system NEMEA. The theoretical part contains a description of how the module obtains information about devices from the network and the practical part contains the design and implementation of the resulting module. The test results confirm functionality and show the time and memory demands of the whole module.

Thanks to PassiveAutodiscovery, the user receives basic information about all devices communicating on the measured network, the role of the user in this network, interconnection between devices and network statistics.

Keywords NEMEA, autodiscovery, device detection, role of the device in the network, servers and clients, network traffic analyze, passive analyze, CESNET, python

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Analýza komunikace na síti je v dnešní době jedna z podstatných úloh každého monitorovacího systému a lze ji provádět různými způsoby podmíněnými konkrétním úkolem analýzy. V konkrétním případě bezpečnostní analýzy je cíl odhalit potencionálně škodlivou komunikaci, jako jsou neoprávněné přístupy k zařízením v síti. Dále může analýza podávat statistické informace o síti a provozu na ní nebo monitorování funkčnosti a vyhodnocování chyb způsobených konfigurací.

Monitorování sítě a následná analýza dat, stejně jako bezpečnost obecně, jsou prostředky pro minimalizaci rizik a hrozeb. Ve většině případů firmu motivuje k implementování monitorování sítě či bezpečnosti případná ztráta při bezpečnostních událostech nebo ztráty způsobené výpadky sítě. Dále díky monitorování sítě může firma sledovat přístupy k jednotlivým částem v síti a tím kontrolovat zda fungují nasazené bezpečnostní politiky. To vše lze z monitorování sítě vyčíst. Rychle a efektivně lze vyčíst i další informace užitečné k dalšímu použití.

Z komunikace na lokální síti lze vyčíst informace o zařízení, které danou komunikaci vysílá či přijímá. Analýza těchto informací nejen přiřadí komunikaci v síti danému zařízení, ale také může rozhodnout o roli zařízení v síti a vytvoření mapy závislostí mezi zařízeními. Takové informace jsou pro správce sítě či zaměstnavatele hodně důležité. Dozví se, jaká zařízení se v síti nacházejí a jak se různá zařízení chovají na síti, a na základě těchto informací můžou být zavedeny změny v síti pro zlepšení komunikačních možností nebo pro zvýšení bezpečnosti. V případě znalosti, že bezpečnostní kamera je jedno ze zařízení připojené k síti může pomoci správci sítě ke specifikování pravidel filtrování v síti, což zablokuje možnost, že by bezpečnostní kamera chovala nestandardně. Rozpoznávání zařízení může být také použito pro blokování přístupu specifickým zařízením. Dále může správce sítě výstup monitorování porovnat s dokumentací sítě a nalézt tak nesrovnalosti s předpokladem.

Ještě podstatnější jsou tyto informace například pro nového správce sítě,

který převzal síť po bývalém správci bez předání potřebných informací. Z analýzy pak dostane představu o tom, jaká zařízení na síti jsou, jak mezi sebou komunikují, jaké služby vyžadují a jaké služby sami poskytují. Zařízení, která služby poskytují, se nazývají servery a jsou v síti ze všech nejdůležitější, jelikož poskytují služby, které ostatní zařízení používají. Analýza pak rozhodne, jaká zařízení jsou servery, a vypíše o nich důležité informace. Mezi nejdůležitější informace patří zejména IP adresa, poskytované služby a seznam klientů.

V práci se budeme zabývat návrhem a implementací modulu pro open-source monitorovací systém NEMEA [2], který disponuje těmito vlastnosti a tento modul následně otestujeme na reálně funkční síti.

Kapitola 1

Cíl práce

Cílem této práce je vytvořit analyzér rozšířených síťových toků zvaných IP flowsIP flows (popsaných např. v [1]), jehož výstupem bude seznam serverů poskytující služby, seznam zařízení vyžadující tyto služby a mapa závislostí mezi těmito dvěma seznamy. Pro tento účel vytvoříme modul pro již existující open-source monitorovací systém NEMEA [2]. Tento modul nazveme PassiveAutodiscovery modul a bude následně volně dostupný ve veřejně dostupném repozitáři na github.com.

PassiveAutodiscovery modul bude schopen z vstupních IP flows získat potřebné informace pro rozhodování o roli zařízení v síti a následnému oštítkování jednoduchými štítky vystihujícími jeho role v síti. Mezi štítky patří třeba DHCP Server, DNS Server a File Server.

Dále modul získá základní informace o jednotlivých zařízeních a z komunikace na síti zjistí závislosti mezi nimi na lokální síti. Tyto závislosti přehledně vypíše včetně služeb, které byly v jednotlivých závislostech požadovány respektive poskytovány, a počtu přenesených paketů. Tuto funkcionalitu provede i pro přístupy zařízení mimo lokální síť a taktéž získané informace přehledně vypíše. Následně informace o závislostech modul vykreslí v mapách závislostí. Nakonec je statisticky vyhodnotí a přehledně vypíše pomocí grafů. Tyto záznamy bude modul vhodně dlouhodobě uchovávat.

Zařízení je v lokálním segmentu sítě jednoznačně rozpoznatelné podle MAC adresy a v celé lokální síti rozpoznatelné podle IP adresy. V této práci rozebereme problém jednoznačného rozpoznání zařízení v síti a pokusíme se implementovat algoritmus, který by spolehlivě rozpoznal zda se jedná o stejné zařízení i v případě změny IP adresy.

Kapitola 2

Existující relevantní programy a práce

Tato kapitola se zabývá již existujícími řešeními problému vyhledávání zařízení v síti, výpisu informací o nich a hledání závislostí mezi nimi. Ke každému programu je popsán jeho způsob řešení tohoto problému a následně je srovnán s vyvíjeným modulem pro systém NEMEA.

Základní kategorizací programů je rozdělení na proprietární a open-source programy. Tedy zda za jejich používání jejich autor či společnost zodpovědná za jejich vytvoření požaduje peněžní odměnu či nikoliv. Vybral jsem z každé této kategorie nejvýznamnější monitorovací programy.

2.1 Proprietární programy

2.1.1 Cisco ISE

Cisco ISE (Cisco Identity Services Engine) [3] je kritická komponenta zero-trust architektury, což je typ zabezpečení firemní sítě, ve které se po každém zařízení vyžaduje ověření „totožnosti“. Cisco ISE umožňuje zautomatizování procesu ověřování „totožnosti“ zařízeních dle předepsaných politik dané firmy.

Aby tato komponenta mohla ověřit „totožnost“ zařízení musí jej nejdříve nalézt a zjistit o nich co nejvíce informací. Proto obsahuje funkcionalitu nazvanou Asset Visibility, která umožňuje profilování zařízení. Cisco ISE [4] nejdříve použije k profilování aktivní sondy a senzory zařízeních, kterými nasbírá data o zařízeních. U aktivních sond v konkrétním případě používá utilitu NMAP (Network Mapper). Dále k profilování používá i pasivní způsob monitorování, který zpracovává data generovaná zařízeními. Jakmile Cisco ISE získá informace o zařízeních porovná je s databází a dle politik v ní nastaví zařízením oprávnění v síti.

V porovnání s naším vyvíjeným PassiveAutodiscovery modulem Cisco ISE

vyhledává zařízení a informace o nich pouze pro účely zero-trust architektury. K profilování zařízení Cisco ISE používá zvolený typ analýzy. Lze nastavit komponentu tak, aby používala pouze pasivní způsob monitorování, při které spoléhá na použité protokoly stejně jako náš modul. Také v pasivní analýze nahlíží Cisco ISE do aplikační vrstvy TCP/IP modelu u specifických protokolů, jako jsou CDP (Cisco Discovery Protocol), SMB (Server Message Block) a další, k zpřesnění pasivní analýzy, kvůli tomu je přesnější než náš modul. PassiveAutodiscovery modul má navíc, ale vyhledávání závislostí a vykreslování grafů závislostí. Dále se liší konkrétním použitím získaných informací. Náš modul informace získává pro zjištění rolí zařízeních a závislostí mezi nimi. Cisco ISE informace využívá jako základní data pro zero-trust architekturu.

2.1.2 User Device Tracker

User Device Tracker je součástí Network Performance Monitoru, což je programová sada vytvořená firmou SolarWinds, která se zabývá vývojem programových řešení managementu sítí, monitorování sítí, managementu databází a IT bezpečností.

User Device Tracker [5] je implementován pro automatické sledování zařízení a správu portů na síťových přepínačích (angl. switch). Jeho hlavní úlohou je nacházení zařízení či uživatelů na síti a spojovat je s konkrétními porty na konkrétním síťovém přepínači. Hledání provádí dle uživatelského jména, IP adresy nebo MAC adresy a s cílem zvýšit bezpečnost v síti. V programu je možné určitá zařízení či uživatele zařadit na whitelist a neupozorňovat na jejich aktivitu explicitně, pouze logovat z jakého portu jakého switche se naposledy připojili nebo zda jsou na něm zrovna aktivní a ostatní zařízení zařadit na takzvaný watch list a program pak na ně automaticky upozorní pokud se do sítě přihlásí.

Program dále umožní kooperaci se síťovými zařízeními, díky čemuž může administrátor vypínat porty přímo z programu a tím zamezit případným bezpečnostním útokům ze stran neznámých či podezřelých zařízení.

Tento proprietární program je spíše určen pro zabezpečení a management switchů a zařízení vyhledává s cílem informovat administrátora o připojení zařízení ke switchi na určitém jeho portu. I tak vypisuje o zařízení základní informace, jako například IP adresu, MAC adresu, výrobce síťová karty či čas poslední aktivity v síti, čímž se mírně přibližuje k našemu PassiveAutodiscovery modulu, který taktéž tyto informace o zařízení vypisuje, ale s úplně odlišným cílem rozpoznávat role zařízení v síti a hledání jeho závislostí. Celková sada Network Performance Monitor dokáže mapovat závislosti mezi zařízeními a ty potom vykreslovat do map, stejná funkcionalita je zařazena i do našeho modulu.

2.1.3 PRTG Network Monitor

PRTG Network Monitor [6] je monitorovací systém vytvořený firmou Peassler, která se zabývá vývojem monitorovacího software s cílem usnadnit administrátorům práci a zvýšit bezpečnost v jejich firmách.

Umožňuje zachytávání paketů, diagnostiku sítě, vyhledání síťových zařízení v síti, zmapování sítě, optimalizaci sítě a monitorování firewallů, IP adres, VoIP služeb, LAN sítě, Wi-Fi připojení, aktivit a bezpečnosti. Z těchto funkcionalit nás nejvíce zajímá monitorování LAN sítě.

Monitorování LAN sítě umožní administrátorovi sledoval aktivitu na síti, včetně pracovních stanic, serverů, routerů či tiskáren. PRTG monitoruje LAN síť a odesílá upozornění na problémy s připojením, omezenou dostupnost nebo přetížený server.

V článku [7] se dozvíme způsob hledání zařízení v LAN síti PRTG Network Monitoru, které probíhá následujícím způsobem:

1. Po zadání IP rozsahu PRTG použije utilitu ping pro najití všech zařízení v zadaném rozsahu.

2. Následně zjistí roli jednotlivých zařízení v síti za použití SNMP (Simple Network Management Protocol), WMI (Windows Management Instrumentation Remote Protocol) a dalších protokolů.

3. Všechna nalezená zařízení jsou nakonec umístěna do takzvaného device tree (někdy devicetree), což je datová struktura popisující hardwarové komponenty počítače.

PRTG hledá zařízení v síti a rozhoduje o jejich roli v síti aktivním přístupem, tedy interaguje se zařízením s cílem zjištěni o jaké zařízení se jedná, na rozdíl od našeho modulu, který všechny informace o zařízení získává pasivním poslouchánim komunikace na síti. Dále je náš modul open-source, což je výhoda oproti proprietárnímu PRTG.

2.1.4 OpManager

OpManager je monitorovací systém od firmy ManageEngine, která je IT management divize mezinárodní firmy Zoho corporation. ManageEngine se soustředí na vývoj monitorovacích softwarů v oblasti IT managementu a má na svědomí přes 90 programových řešení v této problematice.

Monitorovací systém nabízí velice mnoho funkcionalit, které se v převážné většině věnují managementu síťových zařízení a serverů. V konkrétních příkladech to jsou monitorovaní síťových rozhraní, dostupnosti služeb, hardware serverů a síťových zařízení či zátěže na síťové linky. Nejpodstatnější funkcionality OpManageru jsou pro naší práci Ethernet Monitor, LAN Monitor a Network Device Discovery.

Ethernet Monitor [8] provádí v OpManageru monitorování lokální sítě se soustředěním na Ethernetové porty (dále jen porty). Ke každému portu vypisuje zda je na něm připojeno aktivní zařízení, základní informace o připojeném zařízení, zejména IP adresu, a přenosovou statistiku odeslaných a přijatých síťových rámců. Tyto informace pro každý port se v OpManageru nazývají Snapshot page.

LAN Monitor [9] v OpManageru má za úkol monitorovat LAN síť za účelem zvýšení dostupnosti, ale také provádí in-depth analýzu výkonu sítě. Za těmito účely monitoruje dostupnost jednotlivých zařízení, míru jejich komunikace na síti a vykresluje je do přehledové mapy zařízení. Dále umožňuje sledovat zvolený subnet LAN sítě pro evidenci jednou kategorizovaných zařízeních v daném subnetu a přehledně informace o nich vypisovat.

Vypisuje o zařízeních například:

- *Status* uvádí zda je v dané chvíli zařízení aktivní,
- *IP adresu* zařízení,
- *Typ zařízení* určuje operační systém zařízení a
- *Kategorii* upřesnující roli zařízení v síti.

OpManager vypíše všechny tyto informace s jistotou jen pokud byl na daný subnet předtím spuštěn Network Device Discovery popsaný dále.

V článku [10] se dočteme, že *Network Device Discovery* pomáhá nalézat a sbírat informace o zařízeních jako jsou routery, switche, koncová zařízení neboli hosti a firewally. Dále také poskytuje mapování závislostí mezi těmito zařízeními. Nacházení zařízení probíhá stejným aktivním způsobem jako u PRTG Network Monitoru.

Tento program se v mnohých ohledech podobá PRTG Network Monitoru. Na rozdíl od našeho modulu provádí OpManager jednorázově nacházení zařízení aktivním způsobem při spuštění Network Device Discovery funkcionality, ale při běhu LAN Monitoru se provádí, jako u našeho modulu, také pasivní analýza, která buď doplňuje předešlou jednorázovou aktivní analýzu nebo je jediným zdrojem u nově připojených zařízeních.

2.2 Open-source programy

2.2.1 Cacti

Cacti [11] je volně šiřitelný kompletně frontendový RRDTool, což je typ programu, který získává informace typu šířka pásma, teplota hardware a CPU load a ukládá je do databáze s časovou závislostí. Zároveň Cacti umožňuje další funkcionality jako dotazovací služby, pokročilé přizpůsobitelné grafy či management uživatelů.

Program lze nastavit tak, aby přímo sbíral data a základní konfiguraci RRDTool zakázat. Umožňuje navíc vkládat vlastní skripty na sbírání a práci s daty. Pro naše účely je nejzajímavější možnost hledání zařízení v síti, která v Cacti funguje následovně:

1. Použije SNMP nebo vlastní skript uživatele ke shromažďovaní dat o zařízení.

2. Z informací sestaví inventář síťových zařízení.

3. Inventář aktualizuje díky periodicky posílaným SNMP zprávám.

Cacti je open-source program, který je primárně zaměřen na funkcionality RRDTool a díky otevřenosti a vkládání vlastních skriptů pro něj komunita vytvořila množství různorodých skriptů analyzující síť. V porovnání s naším modulem se Cacti soustředí na jinou oblast monitorování a na naší oblast vyhledávání a přiřazování rolí zařízením se nespecializuje, ale vytvořili jednoduchý funkční podprogram, který aktivním způsobem získává data o zařízeních a periodicky tuto aktivitu opakuje pro aktuálnost dat.

2.2.2 Zenmap

Zenmap [12] je grafický scaner programu Nmap. Nmap je open-source program vyvíjený Gordonem Lyonem známým pod přezdívkou Fyodor.

Zenmap respektivě Nmap aktivně interaguje na síti s cílem najít zařízení a posbírat informace o nich, zejména otevřené porty na transportní vrstvě TCP/IP modelu, ze kterých usuzuje jaké služby dané zařízení v síti zprostředkovává. Tato činnost se provádí jednorázově na pokyn uživatele a výstup o zařízeních a jejich závislostech je přímo závislý na aktivních zařízeních v síti v daný moment.

O zařízeních dokáže vypsat:

- IP a MAC adresu,
- operační systém,
- otevřené porty a
- poslední boot.

Zenmap [12] je jednorázový aktivní analyzér sítě, která najde v zadaném subnetu všechny v daný moment aktivní zařízení a ty následně podrobí analýze otevřených portů. Oproti našemu modulu se liší v aktivním analyzování sítě, nenachází závislosti mezi zařízeními, nevypisuje statistické údaje o provozu na síti a nevykresluje grafy závislostí. Přesto je to výborný nástroj pro rychlou analýzu aktivních zařízení v síti.

2.2.3 Spiceworks

Spiceworks [13] je monitorovací nástroj, který není open-source, ale za jeho používaní se nemusí platit, protože jeho vývoj je sponzorovaný reklamou v monitorovacím nástroji. Jeden z jeho produktů podporuje operace zajímavé pro naši práci. Spiceworks Inventory podporuje Network Discovery and Mapping funkcionalitu.

Tato funkcionalita proskenuje síť za účelem získání informací o připojených zařízeních. Pro tento účel používá utility Ping a NMAP. Na základě vytěžených dat vytvoří inventář zařízení a mapu sítě, přičemž o každém zařízení eviduje základní informace jako jsou IP adresa, MAC adresa, Operační systém a další.

Tato utilita monitorovacího nástroje Spiceworks jako ostatní již zmíněné spoléhá na aktivní interakci se sítí přičemž vypisuje základní informace, vykresluje mapu sítě a vytváří statistiky o používání sítě. Náš modul taktéž vypisuje informace o zařízeních a statistiky o síti, ale primárně je určen pro klasifikování role zařízení v síti. Spiceworks se pokouší vykreslovat reálnou mapu sítě, zatímco náš modul se soustředí na vykreslování mapu závislostí mezi zařízeními.

2.3 Pozorování

Všechny zmíněné monitorovací nástroje či systémy spoléhají primárně na aktivní interakci v síti. Čímž získají rychleji přesnější údaje o zařízeních připojených a zapnutých na síti. Zatímco náš modul spoléhá na pasivní monitorování komunikace na síti, ze které zjišťuje všechny informace. Výhodou těchto programů je rychlost a přesnost získání dat o zařízeních, ale za cenu zatěžování linek dotazy na jednotlivá zařízení, čímž se do jisté míry snižuje propustnost v síti. Výhoda našeho vyvíjeného PassiveAutodiscovery modulu pro modulární systém NEMEA je, že svojí činností neomezuje propustnost v síti a nevýhodou je, že aby zjistil informace o konkrétním zařízení musí dané zařízení vygenerovat potřebnou komunikaci na síti.

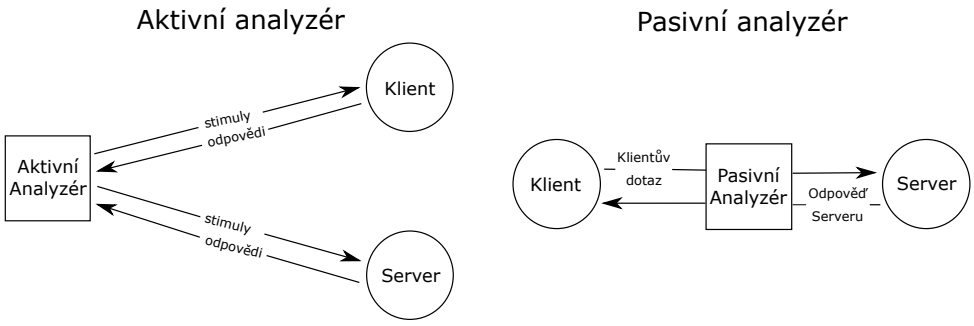
Kapitola 3

Vyhledávání zařízení a určení jejich rolí v síti

Tato kapitola se bude zabývat existujícími dvěma způsoby vyhledávání zařízeních na síti a získaní informací o nich. Jsou to aktivní a pasivní způsob.

V článku [14] se dočteme, že při aktivním způsobu analyzér – hardwarové zařízení či program nainstalovaný na počítači – zahájí komunikaci na síti vysláním síťových rámců směrem k jednotlivým zařízením na síti. Ve většině případů se takovému analyzéru zadá rozsah síťového subnetu, na kterém se nacházejí cílová zařízení pro analyzování. Tento subnet pak zařízení či program projde a na každou IP adresu z rozsahu pošle krátkou zprávu typicky protokolem ICMP, a pokud se na dané IP adrese nachází aktivní zařízení, odpoví na tuto zprávu, čímž se analyzér dozví o aktivním zařízení na dané IP adrese. Poté se na nalezená zařízení vysílají další rámce obsahující zprávy, z jejichž odpovědí se analyzér dozví potřebné informace jako IP adresu, porty, služby, časové informace a informace o operačním systému zařízení.

Pasivní způsob analýzy, jak jsme se dočetli v článku [14], většinou nijak nekomunikuje na síti. Analyzér naslouchá běžné komunikaci na síti a z ní získává informace. Pokud na síti proběhne daná komunikace, tak jsou informace z ní téměř shodné s aktivním způsobem. Pasivní způsob umožňuje oproti aktivnímu způsobu získání informací o závislostech mezi zařízeními a jejich chování v síti. Dále je výhodou pasivního způsobu, že nezatěžuje síťové linky komunikací směrem k zařízením. A díky závislostem a chování zařízení v síti dokáže pasivní způsob mnohdy identifikovat zabezpečovací brány (angl. firewall), síťové směrovače (angl. router) či síťové přepínače (angl. switch) a při provedení NAT (Network Address Translation) potencionálně charakterizovat zařízení za nimi. Limitací pasivního způsobu je umístění analyzéru na síti. Musíme vždy rozhodovat kam umístit analyzér na základě informací, které chceme získat ze sítě. Této problematice se budeme v práci hlouběji zabývat v samostatné kapitole. Pasivní analýza může být použita například na zjištění



Obrázek 3.1: Aktivní vs. pasivní analýza

běžného chování zařízení v síti, prosazování politik na síti, detekování hrozeb a monitorování událostí.

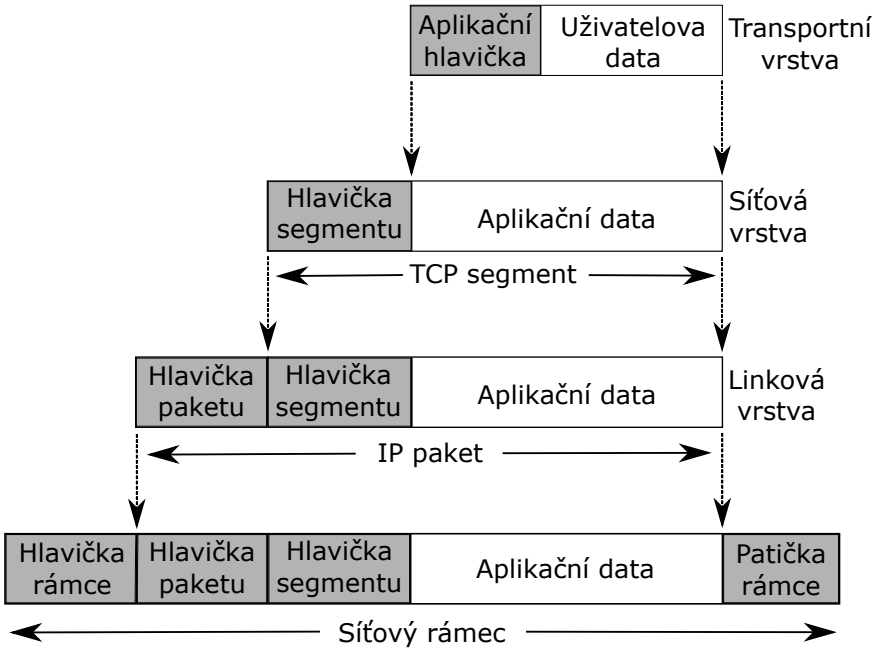
V našem případě použijeme pasivní způsob analýzy zejména na získání základních informací o zařízeních, zjištění jejich rolí v (lokální) síti a závislostí mezi nimi. Nyní si popíšeme, jak toho docílíme.

3.1 Základní informace o zařízení

Mnohdy je spekulativní, co je to základní informace o zařízení. Pro tuto práci to však bude IP adresa zařízení, kterou uživatel může nastavit staticky nebo získat ze sítě dynamicky, čas poslední komunikace na síti, MAC adresa jeho síťové karty, kterou má většinou přidělenou od výrobce (v dnešní době lze MAC adresu změnit), a vendor, tedy výrobce síťové karty. Tyto informace jsou v každé komunikaci, která na síti probíhá.

Síťová a koncová zařízení jsou spojena dohromady pomocí různých typů médií, jako je kroucená dvoulinka a optické kabely. Po těchto médiích se komunikace převádí ve formě signálů signalizujících vždy binární hodnotu nuly nebo jedničky. Dohromady tyto bity vyslané do sítě tvoří síťový rámec první linkové vrstvy TCP/IP modelu, který obsahuje hlavičku, tělo a patičku. V hlavičce je umístěna zejména zdrojová a cílová MAC adresa a v patičce je umístěn kontrolní součet pro tento rámec. Tělo rámce je tvořeno daty, kterými je paket síťové vrstvy TCP/IP modelu. Paket je tvořen hlavičkou a tělem. Hlavička obsahuje zejména zdrojovou a cílovou IP adresu a tělo obsahuje segment třetí transportní vrstvy TCP/IP modelu. V segmentu se taktéž nachází hlavička a tělo. Zde je v hlavičce zejména zdrojový a cílový port použitého protokolu. V těle jsou data čtvrté aplikační vrstvy TCP/IP modelu, jejichž formát záleží na použitém protokolu. Na Obrázku 3.2 je vyobrazeno, jak zapouzdření (po-

psaný proces) probíhá při odeslání dat uživatele po síti. Data z komunikace tedy získáme nahlížením do hlaviček datových formátů jednotlivých vrstev.



Obrázek 3.2: Ukázka principu zapouzdření rámců v síťové komunikaci

V našem modulu se nepracuje přímo se síťovými rámci, nýbrž s flow záznamy, které se do našeho modulu dostanou z jiného již existujícího modulu. Ten vytvoří z rámců flow záznamy tak, že k rámci vyslanému ze zařízení přidá odpověď od druhého zařízení a takto vytvořený flow (biflow) odešle pomocí IFC rozhraní našemu modulu, který z něj pak získá snadněji informace ze všech hlaviček jednotlivých datových formátů popsaných vrstev. Výrobce síťového zařízení se pozná dle první poloviny MAC adresy, který je výrobcům přidělován.

Pro získání všech těchto informací musí být monitorovací systém správně umístěn v síti. Pokud bychom třeba umístili náš modul za router mimo měřený segment lokální sítě, tak pro všechna zařízení, která náleží do tohoto segmentu, získáme stejnou MAC adresu a samozřejmě i vendora. Konkrétně to bude MAC adresa routeru, za nímž se segment sítě nachází. Více si o této problematice a způsobu řešení v našem modulu řekneme v následující kapitole.

3.2 Přiřazování štítků

Štítkováním nazveme proces, při kterém rozhodneme o roli zařízení v síti. Každou roli budeme reprezentovat štítkem, který tuto roli bude dostatečně specifikovat, a tento štítek následně přiřadíme k zařízení. Přičemž jedno zařízení může v síti mít více rolí. Například pokud zařízení bude poskytovat přístup k souborům přes síť, tak tomuto zařízení přiřadíme štítek *File Server*.

O roli budeme rozhodovat primárně z použitých protokolů v síťové komunikaci, jelikož většina protokolů se používá pro jeden specifický účel. Například protokol DHCP se v sítích používá výhradně na přidělování IP adres zařízením. Z předchozí kapitoly víme, že protokoly jsou v síťové komunikaci jednoznačně identifikované pomocí čísel portů v hlavičce segmentu na transportní vrstvě TCP/IP modelu. Z flow záznamů tedy stačí přečíst použité porty a ty následně porovnat se seznamem protokolů, které se používají pro specifické úlohy v síti. Pro tento účel vytvoříme tabulku, která bude tyto informace obsahovat.

Tabulka 3.1: Rozdělení portů

Název	Číselný rozsah	Popis
Známé porty	0 – 1023	dobře známé služby
Registrované porty	1023 – 49151	registrované porty
Dynamické a soukromé porty	49152 – 65535	běžné použití

Porty na transportní vrstvě přidělovala organizace IANA (Internet Assigned Numbers Authority) a od roku 2001 je touto funkcí pověřena organizace ICANN (Internet Corporation for Assigned Names and Numbers). Porty se rozdělují do tří skupin, jak je znázorněno v Tabulce 3.1 [15].

Nyní si projdeme nejpodstatnější role a popíšeme si jejich specifické protokoly, podle kterých je budeme rozpoznávat.

3.2.1 Servery

Nejdůležitější zařízení v síti jsou bezpochyby servery, které poskytují ostatním zařízením služby, bez kterých by nemohli v síti fungovat. Existuje celá řada služeb, jenž můžou servery poskytovat. Tyto služby nyní identifikujeme a přiřadíme každé z nich štítek, jenž jí bude vystihovat. Obsah této sekce čerpá ze zdrojů [15], [16], [17], [18], [19], [20], [21], [22], a [23], které popisují jednotlivé použité protokoly.

DHCP Server přiřazuje zařízením dynamicky IP adresy na základě protokolu DHCP, konkrétněji jeho verzí pro IPv4 a IPv6.

File Server poskytuje přístup k souborům přes síť, přičemž pro tuto funkcionalitu je možné používat celou řadu protokolů. Mezi nejpoužívanější zástupce těchto protokolů patří SMB (Server Message Block), NFS (Network File System) a TFTP (Trivial File Transfer Protocol).

Mail Server umožňuje posílání elektronických zpráv (e-mailů) přes síť. Základními protokoly pro tuto službu jsou IMAP (Internet Message Access Protocol), SMTP (Simple Mail Transfer Protocol) a POP (Post Office Protocol). Tyto protokoly mají více verzí včetně těch šifrovaných.

Web Server poskytuje přístup k webovým stránkám přes protokoly HTTP (Hypertext Transfer Protocol) a jeho šifrované variantě HTTPS (HTTP over TLS/SSL). Tyto protokoly se taktéž používají pro přenos souborů v síti, proto do poznámky u tohoto štítku připíšeme možnost souborových služeb místo těch webových.

DNS Server překládá doménová jména na globální IP adresy, díky čemuž může uživatel z koncové stanice přistoupit na webovou stránku přes doménové jméno. Protokolem pro DNS službu je stejnojmenný protokol, tedy DNS protokol.

Authentication Server autentizuje přihlášení k určitě službě, aplikaci či operačnímu systému. Obvykle pomocí uživatelských jmen a hesel. Pro tuto službu se používají nejčastěji protokoly Kerberos, Radius a Tacacs.

Dictionary Server je sdílená informační struktura obsahující názvy prostředků na síti a IP adresy těchto prostředků v síti. Za prostředky se můžou považovat uživatelé, skupiny, různá zařízení, soubory či složky, tiskárny a telefonní čísla. Pro tento účel se používají protokoly LDAP (Lightweight Directory Access Protocol) a ACAP (Application Configuration Access Protocol).

Chat Server označuje servery v síti, které poskytují vzdálenou komunikaci, jako jsou Skype, Cisco Webex, TeamSpeak a další. Pro tuto oblast existuje řada různorodých protokolů, z nichž je nejpoužívanější IRC (Internet Relay Chat) protokol.

Streaming Server vysílá video přes síť ke klientovi s použitím například RSTP (Real Time Streaming Protocol) protokolu, který se taktéž často používá u IP kamer, proto IP kamery budou často ve výstupu našeho modulu označovány za *Streaming Server* či při použití HTTP protokolu *Web server*.

Log Server ukládá logy více různých zařízeních na síti. Na tento účel se nejčastěji používá Syslog protokol.

Database Server poskytuje databázové služby přes síť. Pro tento účel si takřka každý výrobce databázových systémů vytvořil vlastní protokol. Příklady databázových systémů, pro které tak bylo učiněno jsou MySQL, PostgreSQl či Microsoft SQL.

Proxy Server provádí prostředníka mezi klientem a serverem, přičemž sám vystupuje jako onen klient. Mezi protokoly pro to určené patří Socks a Tproxy protokoly.

Time Server je serverem, který synchronizuje čas zařízení v síti. Nejpoužívanější protokol pro tuto službu je NTP (Network Time Protocol).

3.2.2 Síťová zařízení

Mezi síťová zařízení budeme řadit zařízení, které mají v síti za úkol propojování ostatních zařízení nebo směrování komunikace mezi dalšími síťovými zařízeními. Nejdůležitějším takovým zařízením je router, který se běžně nachází v každé síti a pokud je tato síť připojená do internetu tak v ní musí router být. Pro tento typ zařízení budeme používat štítek *Router*.

Rozhodnout, zda je zařízení s danou IP adresou router, můžeme buď na základě protokolů [15], jako to děláme u serverů, nebo ze závislostí v síti. Protokoly, které routery využívají, se používají k výměně informací mezi routery. Pro výměnu informací se používají protokoly, které podporuje daný routovací protokol nastavený na routerech. Příkladem je RIP protokol (Routing Information Protocol), který používá porty s čísly 520 a 521. Dále při používaní takzvaného „virtuálního routeru“ mezi sebou routery nastavené v tomto uspořádání komunikují prostřednictvím portů 1985 pro protokol HSRP (Hot Standby Router Protocol) a 112 pro protokol VRRP (Virtual Router Redundancy Protocol).

Ze závislostí v síti lze také odhadnout, jaké zařízení je router. Stačí pro to sledovat závislosti mezi IP adresami a MAC adresami. Pokud pro jednu MAC adresu je v síti komunikace s více IP adresami (stejné verze), aniž by před tím byla provedena dynamická změna IP adresy, můžeme odhadnout, že se jedná o router, za nímž se nachází více zařízení komunikujících do segmentu sítě, ve kterém se nachází měřící systém NEMEA.

Stejně tak můžeme předpokládat, že v lokální síti jsou přiřazovány adresy z prefixů určených pro lokální síť, potom při komunikaci na globální IP adresu je pravděpodobně MAC adresa k ní závislá routerem.

3.2.3 Tiskárny

Dalším běžným zařízením v síti hlavně ve firemním prostředí jsou tiskárny. Bohužel často tiskárny používají pro přenos tiskových informací protokol HTTP, takže se může stát, že ve výstupu našeho modulu budou tiskárny označeny za *Web Server*, ale existují i protokoly, které se používají výhradně pro přenos tiskových informací k tiskárnám. V konkrétním případě to jsou protokoly LPD (Line Printer Daemon) [25], který používá Unixový operační systém pro komunikaci s tiskárnami, NPP (Network Printing Protocol) a IPP (Internet Printing Protocol) [24], který na rozdíl od ostatních protokolů podporuje šifrovaný přenos informací.

3.2.4 VoIP telefony

Telefony připojené do sítě se říká VoIP telefony (Voice over Internet Protocol). Tato zařízení se ve firemních sítích vyskytují poměrně často a nahrazují klasické telefony. Může se jednat o telefon, který je místo do telefonní sítě připojen do té počítačové nebo o softwarový program nainstalovaný v počítači.

V článku [26] se dočteme, že nejrozšířenějším protokolem pro tuto službu je protokol SIP (Session Initiation Protocol), který se používá pro navázání spojení a následná komunikace probíhá pod protokolem RTP (Real-time Transport Protocol), který ale nemá pevně určené číslo portu a může si ho zvolit, přičemž obvykle se jedná o čísla z rozsahu 16384–32767. Dále se často pro VoIP používá protokol H.323, který se také používá pro video-konference.

3.2.5 Koncová zařízení

Koncovými zařízeními myslíme všechna zařízení, která používají uživatelé na svoji každodenní činnost. Většinou se bude v našem případě jednat o stolní počítače či notebooky, ale může jít také o chytré mobilní telefony či tablety. Pro tento typ zařízení vytvoříme štítek *End Device*.

Tato zařízení většinou nemůžeme s jistotou rozlišit podle použitých protokolů, a tak můžeme odhadovat ze závislostí mezi zařízeními, ale odhady nemusí vždy být přesné. Ze zkušeností jak se běžní uživatelé, kteří většinou ovládají koncová zařízení, chovají si snadno, že většina uživatelů používá pro přístup na webové stránky zařízení v síti, proto pokud nalezneme závislost mezi dvěma zařízeními (většinou bude jedno v globální síti) a jedno z nich je *WEB Server*, tak druhému přiřadíme štítek *End Device*. Dále běžně uživatelé posílají e-maily, což vyžaduje komunikaci s nějakým *Mail Server*em.

Dále můžeme odhalit operační systém podle specifických protokolů pro něj používaných, ale nebudeme mít jistotu, že se daný operační systém používá na koncovém zařízení. Proto vytvoříme speciální štítky *Windows*, *UNIX* a *Mac OS*. Rozhodnutí, jestli je zařízení s daným operačním systémem koncová stanice nebo server, necháme na uživateli používajícím náš modul.

3.2.6 IoT

IoT (Internet of Things) zařízení jsou různorodá zařízení, většinou připojovaná do domácí sítě, s cílem vytvořit takzvanou „chytrou domácnost“, ale počítají se mezi ně i například IP kamery. V rámci našeho modulu pracujeme převážně s čísly portů a závislostmi, což nás ve světě rozpoznávání IoT dost limituje, jelikož používají protokoly primárně určené na jiné služby pro přenos informací. Příkladem je HTTP. Z práce o TCP skenování na IoT zařízení [27] s cílem rozpoznat je sice nemůžeme použít jejich logiku kvůli pasivnímu řešení našeho modulu, ale můžeme použít specifické porty, které v práci uvedli.

To nám umožňuje vytvořit dva štítky. *IoT* pro zařízení pro chytrou domácnost jako je třeba Amazon Echo a štítek *IP Camera* speciálně pro IP kamery.

U nich je dříve popsaný problém, že často používají pro přenos videa HTTP či RTSP protokoly, které mají jiný primární účel. Práce [27] nám dává specifické většinou sekundární protokoly pro Samsung, Dlink, Belkin a Netatmo IP kamery, ale i tak se bude stávat, že IP kamera bude ve výstupu našeho modulu označena za *WEB Server* nebo *Streaming Server*.

Kapitola 4

Umístění systému v síti

V této kapitole nastíníme problematiku umisťování zařízení v síti a pokusíme se doporučit nejlepší možnosti.

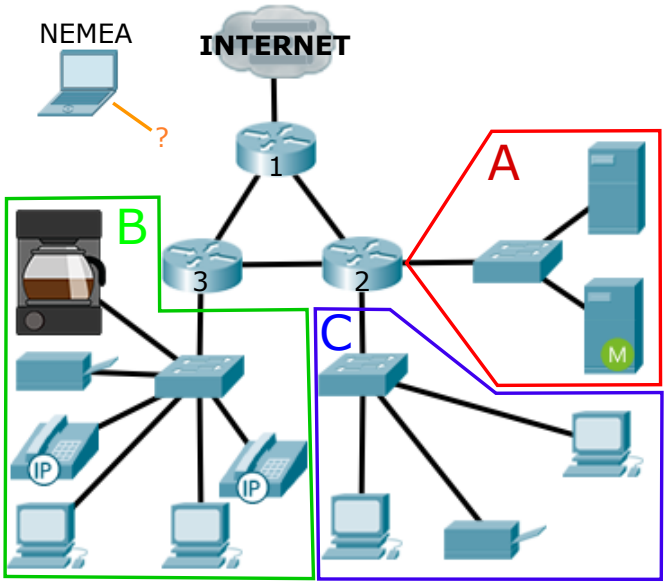
4.1 Lokální síť vs. lokální segment sítě

Pod pojmem lokální síť si všichni představíme síť s privátním rozsahem IP adres, která je za routerem či více routery, které jedním portem směřují do lokální sítě a jiným směřují do internetu, přičemž provozují pro lokální síť NAT (Network Address Translation). Úlohou překladu adres (proces, který provádí router při NAT) je umožnit všem zařízením lokální sítě, aby vystupovali v internetu pod jednou či více globálních adres, které jsou nastavené na portu síťového směrovače (angl. router) směřujícím do internetu.

Lokální síť může být pouhý síťový přepínač s pár zařízeními, ale také se může jednat o rozsáhlou síť s celou řadou síťových přepínačů, směrovačů, AP a dalších síťových zařízeních. Pokud je lokální síť členěná pomocí síťových směrovačů na více malých sítí s různými síťovými rozsahy IP adres, tak nazýváme každou takovou malou síť s vlastním síťovým rozsahem, která je propojena se zbylou lokální sítí síťovým směrovačem, lokální segment sítě. Tudíž každá lokální síť se skládá z jednoho či více segmentů. Příklad lokální sítě rozdělené na segmenty můžeme vidět v Obrázku 4.1, kde jsou nejdůležitější segmenty označeny velkými tiskacími písmeny.

Pokud náš modul zapojíme do jednoho segmentu lokální sítě, tak pro každé zařízení z tohoto segmentu dokáže nalézt jeho MAC adresu, ale už nezíská MAC adresy zařízeních z vedlejšího segmentu. Je to způsobené tím, že MAC adresy jsou v sítích používané pro síťování v jednom segmentu sítě a během přechodu paketů přes směrovač do jiného segmentu se při směřování paketu routerem směrem k cílovému zařízení vymění zdrojová MAC adresa odesílatele za MAC adresu směrovače, přičemž si směrovač původní adresu uschová k výměně u případné odpovědi.

V segmentu jsme tudíž schopni získávat více informací o zařízeních v daném segmentu. Konkrétně získáme MAC adresu, výrobce síťové karty zařízení a také můžeme odhalit změny IP adres pro zařízení a tím spojit provoz jednoho zařízení pod více než jednou IP adresou.



Obrázek 4.1: Příklad zapojení (vytvořeno v Cisco Packet Tracer a Inkscape)

4.2 Zapojení systému NEMEA do sítě

NEMEA systém musíme umístit do sítě tak, aby bylo možné sledovat provoz. Tudíž jej musíme připojit k nějakému síťovému zařízení, které umožňuje funkcionalitu známou jako port-mirroring. Port-mirroring je funkce switche či routeru, která veškerý provoz na daném zařízení kopíruje na port, na němž je port-mirroring nastaven. Po nastavení switche či routeru a zapojení počítače, na kterém je systém NEMEA nainstalován, vidí systém veškerý provoz, který prochází daným síťovým zařízením, ale zatěžuje to CPU síťového zařízení. Ke stejnému účelu lze použít speciální zařízení TAP (Terminal Access Point), které umí kopírovat provoz.

Rozhodnout, kam do sítě zapojit NEMEA systém s naším modulem, je složitější, než se zdá, jelikož každá volba ovlivní naměřený provoz. Na Obrázku 4.1 je vyobrazena síť, u které způsob zapojení NEMEA systému do sítě značně ovlivní získaná data. Pokud zapojíme modul do síťového přepínače (angl. switch) v segmentu A s nastavení port-mirroring, získáme provoz mezi zařízeními v segmentu A, provoz přístupů k zařízením v segmentu A ze segmentů B a C a provoz zařízení ze segmentu A do Internetu. Neodchytíme

přitom komunikaci mezi segmenty B a C a taktéž komunikaci v rámci jejich segmentů.

Abychom získali i tuto komunikaci, mohli bychom nastavit na síťových zařízeních v jejich segmentech port-mirroring a kopii provozu z port-mirroring posílat tunelem mezi síťovými zařízení na náš modul v segmentu A. To by ale zatížilo klíčové linky mezi segmenty a navíc bychom měli některé záznamy duplicitní. Například záznam o komunikaci ze segmentu B do segmentu C by port-mirroring zachytil dvakrát, jednou v segmentu B a podruhé v segmentu C.

Při zapojování našeho modulu do sítě musíme před volbou zvážit, jaký provoz nás zajímá nejvíce a co je cílem našeho měření. Již zmíněné zapojení do segmentu A nám v příkladu získá seznam zařízení, které přistupují ke klíčovým serverům sítě.

4.3 Jednoznačné určení zařízení

Na segmentu lokální sítě jsme schopni bezpečně rozhodovat jaké IP adresy a v jakém čase je zařízení mělo. Díky viditelnosti MAC adres a DHCP protokolu může náš modul sledovat změny IP adres pro danou MAC adresu a tím spojit komunikaci jednoho zařízení pod více IP adresami.

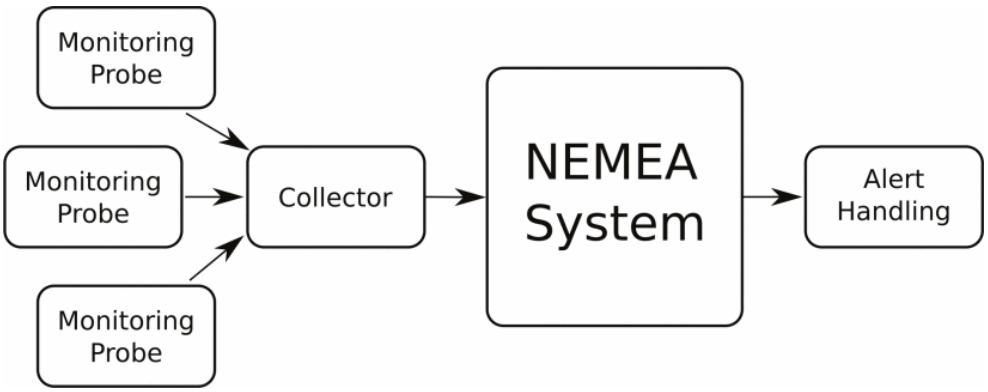
Na celé lokální síti to bohužel není možné, pokud neměříme ve všech segmentech, ale můžeme zachytávat DHCP komunikaci. Dané záznamy pak můžeme vypsat a tím pomoci správci sítě, aby měl přehled o komunikaci zařízeních s DHCP servery.

Kapitola 5

NEMEA systém

Network Measurements Analysis (NEMEA) [2] je navržen s ohledem na streamwise koncept, tzn. data jsou průběžně analyzována v paměti s minimálním ukládáním dat. Vyvíjen jako open-source projekt veřejně přístupný pro celosvětovou komunitu a navržen jak pro experimentální tak i provozní použití. Je schopen pracovat online, tzn. přímo připojen do sítě pracující s daty v jejím provozu, i offline, tzn. flow záznamy jsou uloženy a následně analyzovány nezávisle na provozu na síti.

Systém NEMEA [2] je navržen jako heterogenní modulární systém. Moduly jsou nezávislé procesy propojené jednosměrnými rozhraními pro komunikaci. Rozhraní přenášejí libovolná data ve formě toků zpráv — flow záznamy, výsledky analýzy a další. Obrázek 5.1 ukazuje jak probíhá typické monitorování provozu v systému NEMEA.

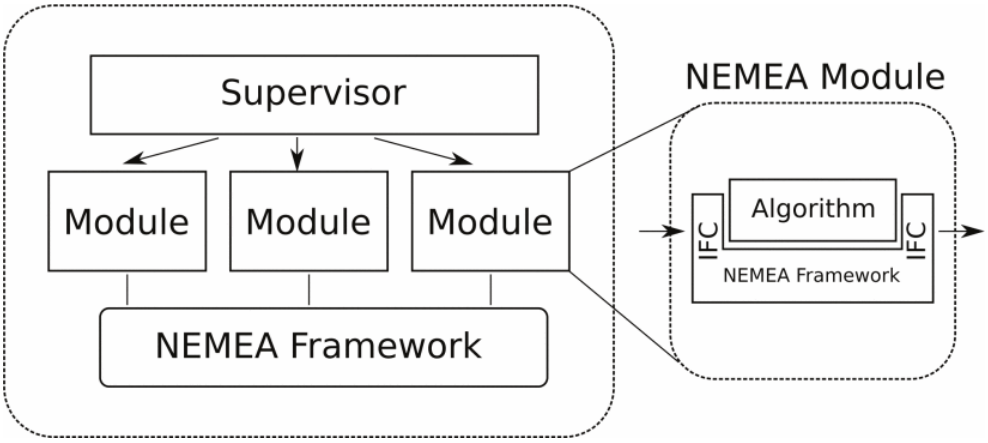


Obrázek 5.1: Monitorování s NEMEA systémem [2]

Infrastruktura monitorovacího systému je založena na exportu flow záznamů (pomocí monitorovacích sond), které jsou předávány k uložení (Collector) a analýze (NEMEA). Detekční moduly NEMEA vytvářejí výstrahy v jednotném formátu, který je vhodný pro následné zpracování a uložení.

5.1 Architektura

Jak již bylo řečeno, systém NEMEA [2] je modulární systém složený z nezávislých modulů. Tyto moduly jsou kontrolovány a monitorovány pomocí nástroje nazvaného supervizor, který se vyplatí používat při větším počtu modulů a může běžet jako systémový daemon nebo v interaktivním režimu. Supervizor periodicky dostává od modulů stavové informace. Dále sleduje hardwarové zdroje modulů, jako například využití procesoru a pamětí RAM. Všechny moduly jsou stavěny na NEMEA Framework, tzn. moduly používají funkcionality, které jsou implementovány ve veřejných knihovnách NEMEA Frameworku. Každý modul obsahuje algoritmus, který zajišťuje jeho funkcionality, pro které byl navržen. Na Obrázku 5.2 je architektura NEMEA systému vyobrazena.



Obrázek 5.2: Architektura NEMEA systému [2]

5.2 Komunikační rozhraní

Moduly NEMEA systému spolu mohou komunikovat skrze TRAP Communication Interfaces (IFC). Přičemž každý IFC je vstup nebo výstup modulu a každý modul může mít libovolný počet vstupních a výstupních IFC. Data, které se posílají na IFC, jsou formátovaná do zpráv maximální velikosti 64 kB a mohou obsahovat:

- flow záznam,
- výsledek algoritmu,
- statistické údaje nebo
- cokoliv jiného.

Existují různé druhy IFC a nejpodstatnější dva jsou založeny na UNIX soketech a TCP soketech. První se používá na komunikaci mezi jednotlivými moduly NEMEA systému a druhý se používá na komunikaci přes síť. Dále se může speciálními IFC zapisovat flow záznamy do souboru nebo při měření výkonu zapisovat data do blackhole.

Přes IFC se mohou posílat data ve třech formátech a to:

- nestrukturovaná data,
- JSON formát nebo
- speciální binární formát UniRec systému NEMEA, ve kterém se přenášejí nejčastěji flow záznamy mezi moduly.

5.3 Existující moduly důležité pro naši práci

5.3.1 Flow meter

Tento modul ze síťového rozhraní či souboru vytvoří IP flows, které propaguje dalším modulům pomocí výstupního IFC. Pro nás modul to bude zdroj IP flows, které vytvoří z rámců přicházejících na síťové rozhraní většinou přes port-mirroring.

5.3.2 Logger

Tento modul může z IFC rozhraní, který naplňuje flow meter, získávat IP flows a ukládat je do souboru. Díky tomu může uživatel IP flows uložit do souboru a do našeho modulu tyto data poslat kdykoliv z uloženého souboru.

Kapitola 6

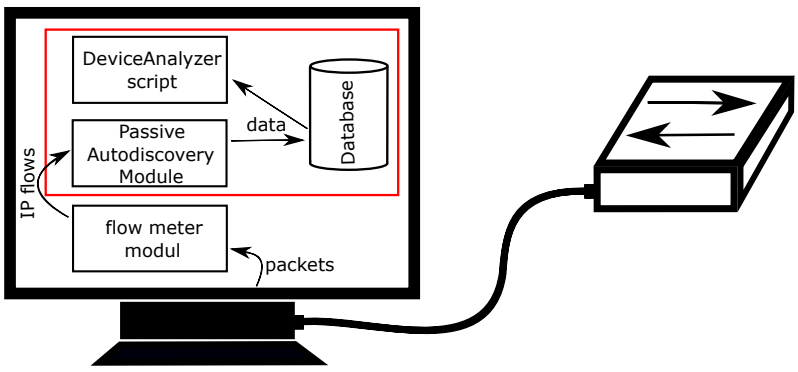
Návrh

6.1 PassiveAutodiscovery modul

Úkolem modulu je přiřadit zařízením štítky, zaznamenávat závislosti mezi zařízeními a následně je vykreslit do grafu závislostí.

Aby modul mohl zařízení přiřadit štítek, musí porovnávat čísla portů, které dané zařízení používalo, se seznamem portů typických pro určitá zařízení, který vytvoříme. Tedy proces štítkování převedeme na vyhledávání, zda se použitý port nachází v seznamu.

Co musíme řešit jako první je způsob uložení seznamu portů služeb, ukládání závislostí a dalších informací k zařízením. Ideálním řešením pro tento problém se nám nabízí databáze. Díky databázi můžeme nad daty provádět dotazy, kterými snadno přiřadíme zařízením štítky, rozhodneme o výrobci síťové karty či zmapování závislostí do grafu.

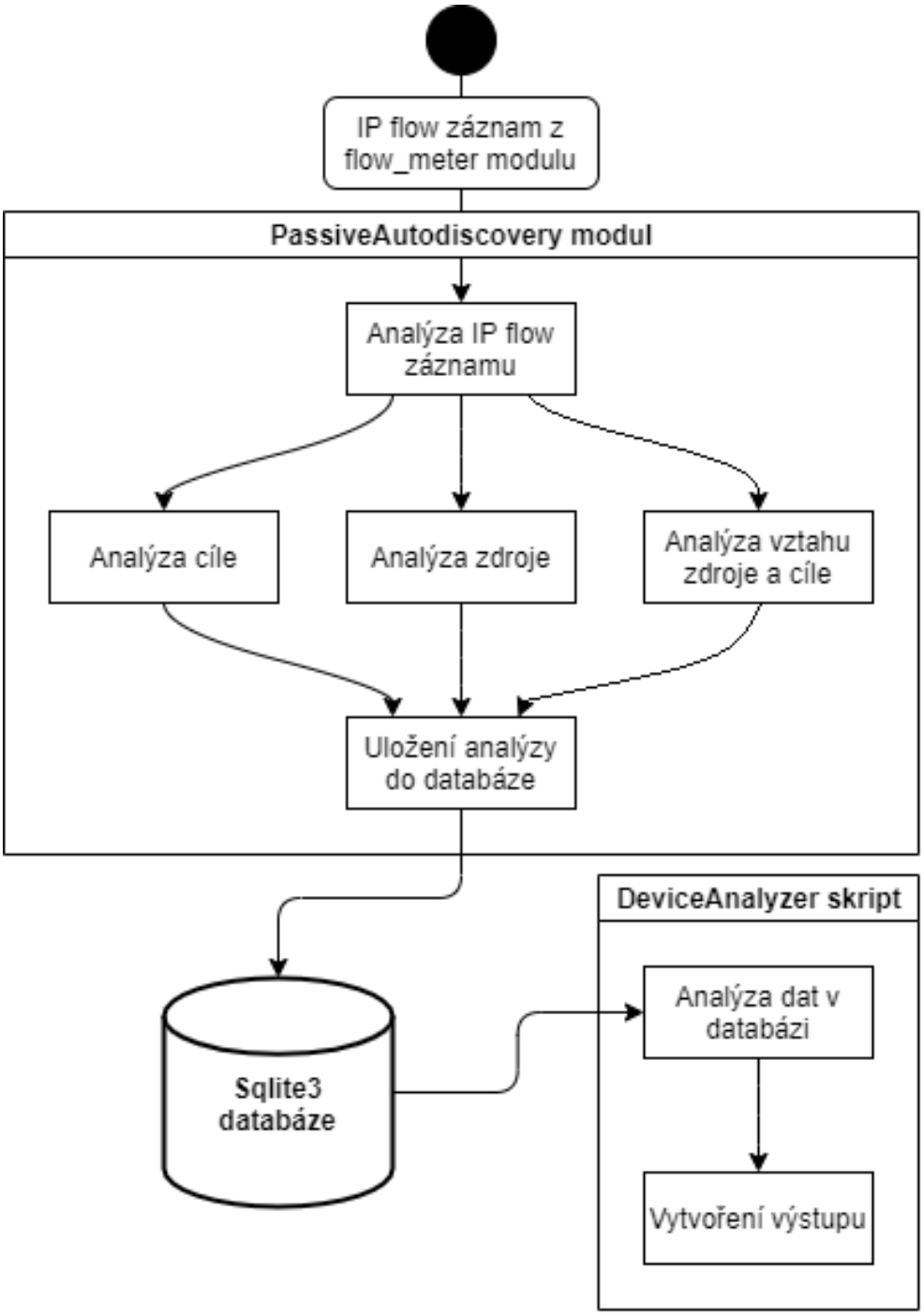


Obrázek 6.1: Návrh modulu (vytvořeno v Inkscape)

Jelikož můžeme předpokládat, že většina uživatelů našeho modulu bude chtít sledovat síť za účelem získání údajů o zařízeních v nějakém časovém okně, tak je pravděpodobné, že uživatel nejdříve spustí modul a na výsledky se bude chtít podívat až po skončení tohoto časového okna. Proto rozdělíme modul na dvě části.

První část je PassiveAutodiscovery modul samotný, který spustíme spolu s flow meterem z NEMEA systému. Uživatel nejdříve nastaví náš modul a poté modul bude dle nastavení pracovat s IP flows. Ty mu v předpřipraveném stavu předloží flow meter. Modul poté z IP flows vytěží veškerá užitečná data jako je zdrojová a cílová IP adresa, zdrojová a cílová MAC adresa či zdrojový a cílový port. Tato data modul vhodně uloží do databáze tak, aby je předpřipravil druhé části modulu. Zároveň bude modul moci vypisovat nově nalezené informace, se kterými během ukládání pracuje.

Druhá část programu bude skript, který bude pracovat s již uloženými daty v databázi a podá komplexní výstup o měření. To znamená, že projde v databázi všechna nalezená zařízení z měřených sítí a z databáze pomocí dotazů zjistí vše, co bylo z IP flows možné zjistit. Jeho primárním výstupem budou informace o zařízeních vypsané do příkazové řádky a obrázky grafů závislostí. Sekundárním bude JSON dokument obsahující všechny získané informace o zařízení. Tento skript nazveme DeviceAnalyzer a bude na našem modulu úplně nezávislý, takže bude moci být spuštěn kdykoliv po skončení běhu modulu a i během běhu modulu, ale pouze s voláním modulu.



Obrázek 6.2: Diagram spolupráce částí modulu (vytvořeno v app.diagrams.net)

6.2 Databáze

6.2.1 Volba databáze

Při volbě databáze musíme zvážit, za jakým účelem budeme databázi používat. Náš modul pouze potřebuje nasbíraná data vhodně ukládat a poté na nich provádět SQL dotazy. Nepotřebujeme tudíž databázový server se spoustou konfiguračních a přístupových funkcí. Zároveň stavba databázového serveru může být problematická a jeho běh náročný na zdroje měřícího zařízení. Proto hledáme databázi, která neběží na serveru, ale rychle a snadno se pomocí jednoduchých skriptů vytvoří, spustí a dokáže provádět dotazy.

Proto jsme zvolili SQLite3 databázi, která pracuje pouze se souborem, nad kterým provádí dotazy. Bohužel při každém vkládání dat do souboru SQLite3 čeká, až se do souboru, který představuje databázi, zapíší data, což je pomalejší než běžné databáze. Tuto nevýhodu se pokusíme co nejvíce minimalizovat možností používat RAM paměť pro průběžné ukládání databáze.

6.2.2 Databázový model

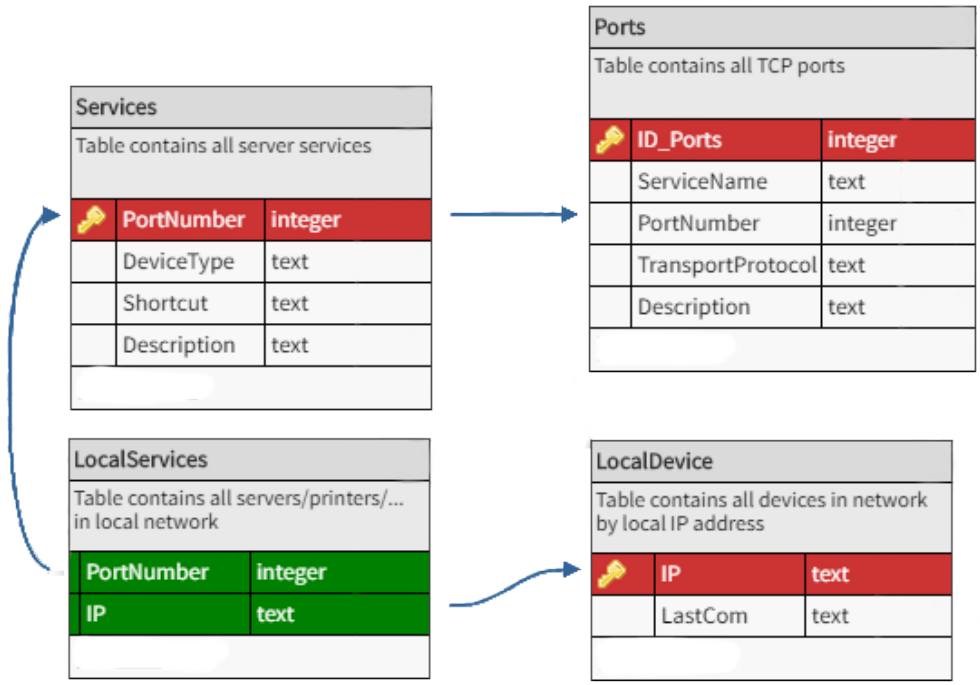
Nejdříve navrhneme podobu tabulek pro přiřazování štítků, což je primární funkcionalita našeho modulu. Jako základní kámen pro celou naší databázi musí být tabulka obsahující všechna nalezená lokální zařízení. Tuto tabulku pojmenujeme *LocalDevice*.

K přiřazování štítků potřebujeme tabulku obsahující základní informace o dobře známých a registrovaných portech, jejichž rozložení máme vyobrazeno v Tabulce 3.1. Tudíž vytvoříme tabulku *Ports*, do které tyto informace vložíme z oficiálního zdroje [15]. Dále vytvoříme tabulku pro porty používané specifickými zařízeními. Nazveme jí *Services* a vložíme do ní námi vytvořený seznam pro štítkování obsahující ke každému portu typ zařízení a popis použití portu u daného zařízení.

Tabulky *Services* a *LocalDevice* spojíme pomocí tabulky *LocalServices*, do které budeme přidávat k jednotlivým zařízením použité porty. Z této tabulky pak snadno přes dotazy získáme štítky pro jednotlivá zařízení. Tato část návrhu databáze je vykreslena na Obrázku 6.3.

Další důležitou funkcionalitou našeho modulu je vyhledávání závislostí mezi zařízeními.

K základní tabulce lokálních zařízení *LocalDevice* přidáme tabulku se závislostmi mezi lokálními zařízeními, kterou pojmenujeme *Dependencies*. Tabulka nám umožní vkládat jedinečné závislosti. Za jedinečnou závislost považujeme takovou závislost, u které je vazba mezi dvěma lokálními zařízeními a port použitý jedním ze zařízení se nachází v tabulce *Services*, respektive *Ports*. Pokud se port nalezne v tabulce *Services* je přidán štítek danému zařízení v tabulce *LocalServices*. Takže jedinečná závislost je vazba zařízení A na



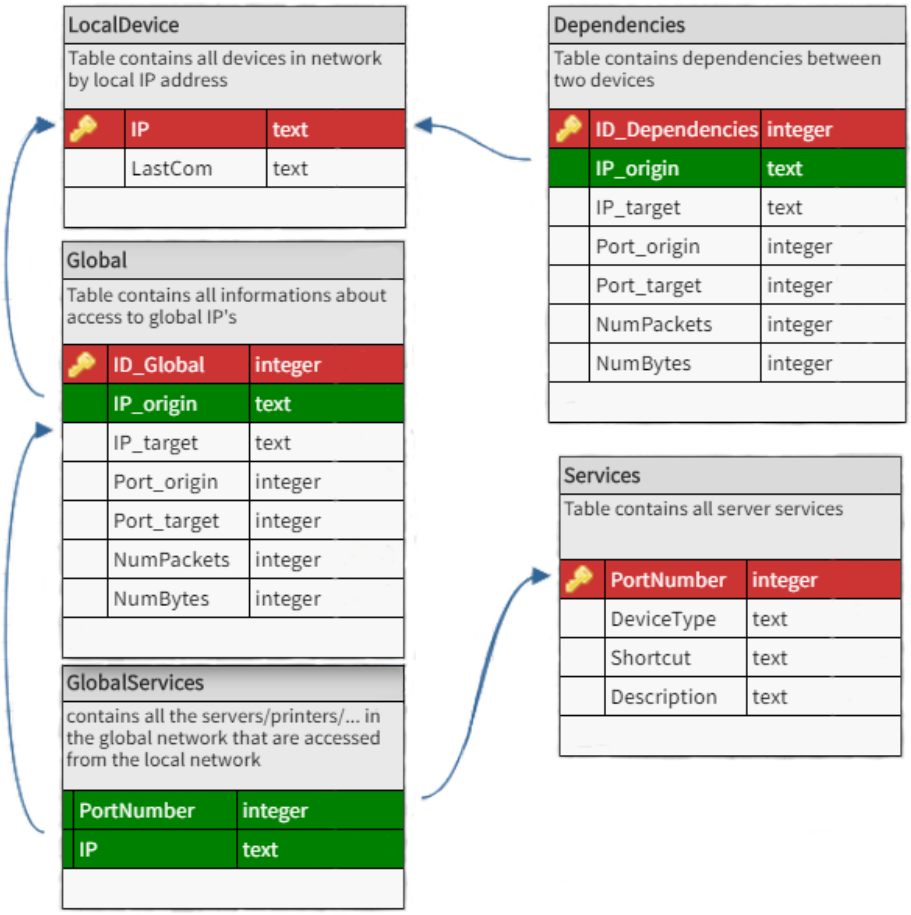
Obrázek 6.3: Přiřazování štítků (vytvořeno v `https://app.dbdesigner.net/`, upraveno v Adobe Photoshop)

zařízení B, přičemž port zařízení B, respektive port zařízení A, je známý či registrovaný port.

Dále vytvoříme tabulku pro přístupy lokálních zařízení do internetu (s IP adresou z globálního prefixu). Tabulka bude téměř shodná s tabulkou *Dependencies*, ale jedná se o vazbu lokálního zařízení na zařízení v internetu. Tuto tabulku nazveme *Global*. K zařízením z globální sítě taktéž vytvoříme tabulku typů zařízení *GlobalServices*. Nákres na Obrázku 6.4.

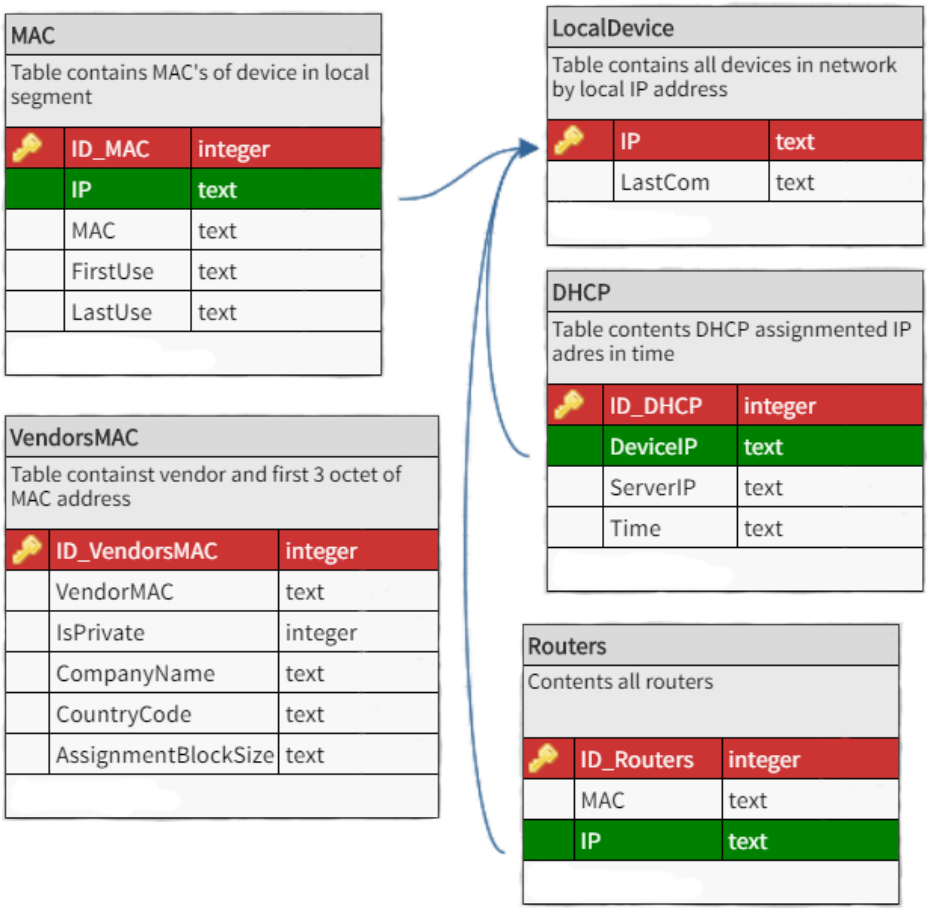
Další tabulky, které budeme potřebovat, jsou *MAC*, *VendorsMAC*, *DHCP* a *Routers* zobrazené na Obrázku 6.5. Tabulka *MAC* slouží k uložení MAC adresy k IP adresám na lokálním segmentu sítě. Úzce spolupracuje s tabulkou *Routers*, do které jsou v případě zjištění, že MAC adresa patří routeru, přelijí záznamy dané MAC adresy, které se nacházejí v tabulce *MAC*. V tabulce *Routers* se poté nachází MAC adresy routerů a všechny IP adresy, které jsou za těmito routery a komunikovali do segmentu sítě, ve kterém měříme.

Tabulka *DHCP* slouží k ukládání DHCP záznamů k jednotlivým zařízením, podle čehož poté rozhodujeme o tom, zda je MAC adresa routerem či nikoliv. Poslední přidaná tabulka obsahuje informace o výrobcích síťových karet, rozpoznatelných podle první poloviny MAC adresy.



Obrázek 6.4: Závislosti mezi zařízeními (vytvořeno v `https://app.dbdesigner.net/`, upraveno v Adobe Photoshop)

Na Obrázku 6.6 můžeme vidět návrh dvou tabulek *DependenciesTime* a *GlobalTime*, které slouží pro uložení časových záznamů závislostí. Pro každý IP flow záznam uchováváme o jakou závislost se jedná, čas a počet paketů v daném IP flow. Ze záznamů pak můžeme snadno zrekonstruovat jak probíhala komunikace pro každou závislost.

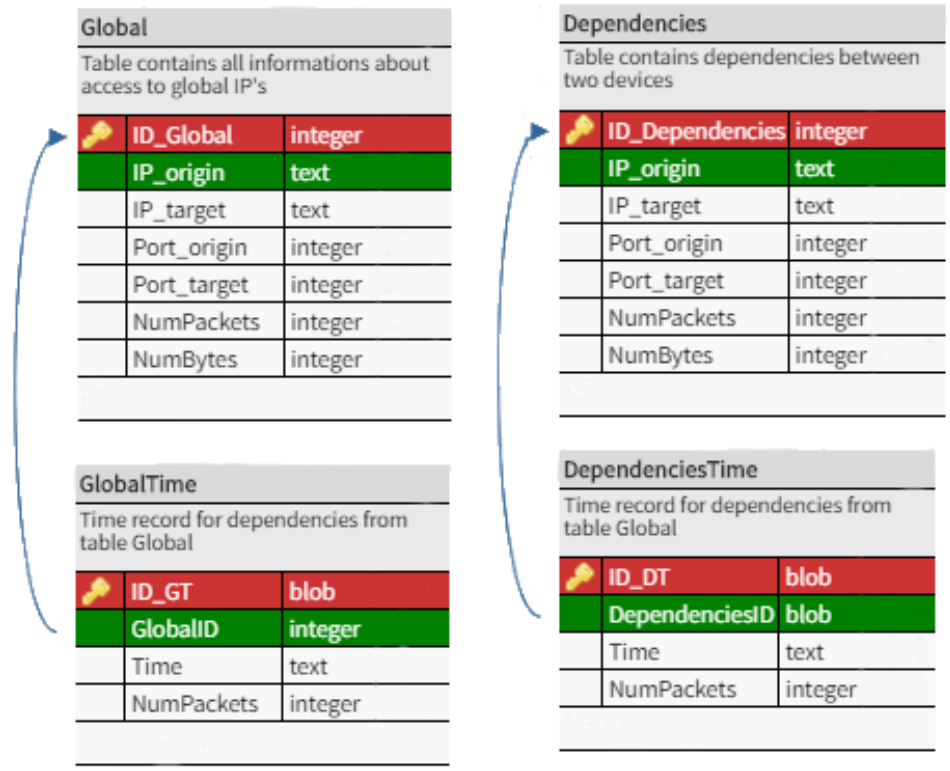


Obrázek 6.5: MAC a DHCP (vytvořeno v `https://app.dbdesigner.net/`, upraveno v Adobe Photoshop)

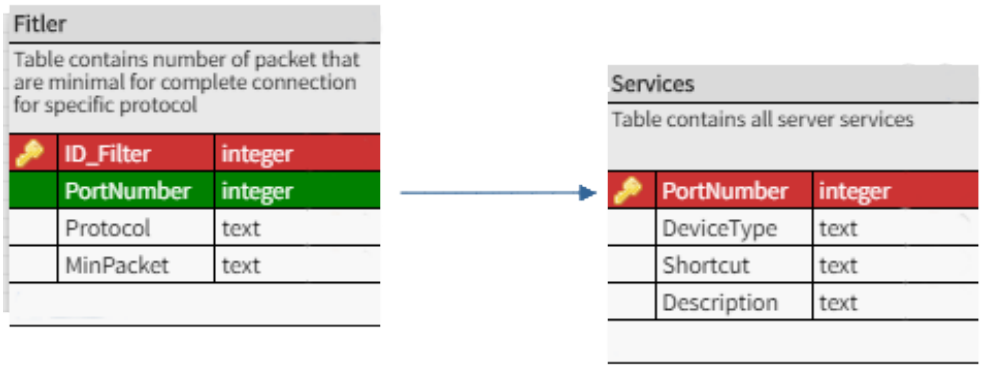
Poslední potřebnou tabulkou je *Filter*, kterou vytvoříme pro filtrování neúplného provozu. Tabulka obsahuje minimální počet paketů pro úplný provoz pro specifický protokol s daným číslem portu. Návrh můžeme vidět na Obrázku 6.7

6.2.3 Vytvoření databáze

Aby uživatel nemusel databázi vytvářet manuálně, tak je součástí výstupu této práce instalační skript, který automaticky databázi vytvoří z SQL souboru a vloží data do tabulek *Services*, *Filter*, *Ports* a *VendorsMAC*.



Obrázek 6.6: Časové záznamy (vytvořeno v `https://app.dbdesigner.net/`, upraveno v Adobe Photoshop)



Obrázek 6.7: Filtrování komunikace (vytvořeno v `https://app.dbdesigner.net/`, upraveno v Adobe Photoshop)

Kapitola 7

Implementace

V této kapitole vysvětlíme fungování naší implementace jednotlivých částí naší práce a ukážeme si nejdůležitější prvky implementace. Veškeré části práce jsou programované v jazyce python verze 3.6.

7.1 CreateScript

Python skript „CreateScript.py“ nejdříve zjistí, zda databáze se jménem zadaným v parametru -d již existuje, pokud ano, zeptá se nás, jestli jí chceme přemazat či nikoliv. To nám dává možnost zálohovat si databázi. Poté skript vytvoří ze souboru „Database_sqlite_create.sql“ databázi.

Jakmile je databáze vytvořená přistoupí skript pomocí knihovny *urllib* [28] na url odkazy, ze kterých stáhne data pro tabulky *Ports* a *VendorsMAC*. Pokud se mu nepodaří nějakou tabulku naplnit daty z internetu, použije naší zálohu, která je součástí vyvinutého programu. Nakonec skript použije námi vytvoření seznam zařízení a jejich často používaných portů k naplnění tabulky *Services*.

7.2 PassiveAutodiscovery modul

Náš výchozí modul primárně naslouchá na IFC rozhraní popsané v kapitole NEMEA systém a získané IP flows distribuuje do skriptu „Collector.py“, který IP flows zařazuje do databáze. Pro tuto funkcionalitu využívá především knihovny *pytrap* [2] z NEMEA systému.

Dále nám tento modul slouží jako komunikační rozhraní. Při spuštění pomocí parametrů určíme nastavení, kterým zpřesníme měření i následnou analýzu a při jeho běhu zde můžeme při vhodném nastavení nalézt základní informační výpisy o nově nalezených zařízeních či závislostí.

7.3 Collector

Skript Collector obsahuje obsahuje důležitou stejnojmennou funkci, kterou volá „PassiveAutodiscovery.py“ modul. Funkce dále dle nastavení zadané v modulu vyplňuje databázi „Database.db“ voláním dalších pomocných funkcí. Přičemž jejich volání záleží na IP adresách komunikujících zařízeních.

Pokud jsou obě zařízení lokální tak se pro ně volají funkce na přidání lokálního zařízení do tabulky *LocalDevice*, funkce pro kontrolování, zda zařízení nekomunikují pomocí specifického portu pro nějaké zařízení a nakonec funkce pro závislost dvou lokálních zařízení. Pokud je jedno z nich modulem považováno za lokální (zařízení z privátního rozsahu IP adres či z explicitně zadané sítě) a druhé nikoliv, tak se funkce volají pro jedno z nich a zároveň se volá funkce pro globální závislosti. V případě, že ani jedno zařízení není lokální, funkce skončí bez žádných akcí.

„Collector.py“ považuje při analýze za lokální zařízení zařazované do tabulky *LocalDevice* buď zařízení s IP adresou z privátních rozsahů, nebo z rozsahů námi zadaných při spuštění modulu. Každé nové zařízení je poté přiřazeno do tabulky, a pokud v ní již existuje, tak se aktualizuje čas poslední aktivity.

Ke každému lokálnímu zařízení se zaznamenávají jednotlivé štítky, které se snadno naleznou jednoduchým SQL dotazem:

```
SELECT * FROM Services WHERE PortNumber = {port};
```

kde port je použitý port na transportní vrstvě TCP/IP modelu zařízením v daném IP flow.

Pokud dotaz nalezne záznam, tak má zařízení roli zapsanou ve vybraném řádku z tabulky *Services*. Samozřejmě musíme před vložením záznamu do tabulky *LocalServices* zjistit, zda se totožný záznam již v tabulce nenachází, abychom nevytvářeli duplicitní záznamy.

Závislosti jak lokální, tak globální, se přidávají, pokud alespoň jeden z použitých portů v daném IP flow existuje v tabulce *Ports*, čili zda je port registrovaný či dobře známý. Pokud stejný záznam již existuje, tak skript navýší počet přenesených paketů v dané závislosti.

7.4 DeviceAnalyzer

Skript „DeviceAnalyzer.py“ vytváří hlavní výstup celého modulu a pracuje zcela nezávisle na „PassiveAutodiscovery.py“. To znamená, že jej uživatel může spustit kdykoliv, pokud existuje databáze „Database.db“ a jsou v ní nasbíraná nějaká data a zároveň žádný jiný skript do ní zrovna nezapisuje. Aby bylo možné pustit jej během měření sítě pomocí „Passiveautodiscovery.py“, musíme toto měření pozastavit.

Pomocí tohoto skriptu se nám zobrazí buď analýza pro jedno konkrétní zadané zařízení nebo kompletní analýza pro každé zařízení obsahující:

- *IP adresu*, ke které je přiřazená komunikace na lokální a globální sítí dále analyzovaná
- *další IP adresy* pokud se podařilo zjistit, že jedno zařízení komunikuje v síti pod více IP adresami (IPv4, IPv6 a změny na těchto adresách)
- *MAC adresu* pokud je zařízení ve stejném segmentu sítě jako měřící zařízení
- *Výrobce síťové karty* pokud máme k dispozici MAC adresu
- *Přiřazené štítky* spolu s popiskem obsahující jaký protokol byl použit
- *záznamy DHCP*
- *Lokální závislosti* mezi zařízeními, které „Collector“ považoval v době měření za lokální
- *Statistiku pro lokální závislosti* uvádějící v procentech, jaké protokoly byly nejpoužívanější
- *Globální závislosti* obsahujícím přístupy zařízení mimo síť
- *Statistiku pro globální závislosti* uvádějící v procentech, jaké protokoly byly nejpoužívanější

Tyto informace se ve většině případů získávají z databáze vhodnými SQL dotazy. V konkrétním případě si můžeme ukázat, jak vypadá vyhledání přiřazených štítků pro výpis:

```
SELECT S.PortNumber, S.DeviceType, S.Shortcut, S.Description
FROM LocalServices LS JOIN Services S
ON LS.PortNumber=S.PortNumber WHERE LS.IP='{ip}';
```

kde ip je IP adresa lokálního zařízení, pro které vypisujeme přiřazené štítky. Tento dotaz nám vrátí všechny štítky, které byly přiřazeny danému zařízení.

Nakonec analýzy zařízení se vypíše statistika celé sítě uvádějící v procentech, jaká zařízení z lokální sítě používala síť v době měření nejvíce. Pro přehledný výpis statistiky se používá knihovna *termgraph* [29]. Pokud uživatel zvolí, tak se vytvoří grafy závislostí, které jsou uloženy do souborů. Grafy se vytváří z tabulek *Dependencies* a *Global* pomocí knihoven *networkx* [30] a *matplotlib* [31].

Přičemž závislosti lokálních zařízení z tabulky *Dependencies* jsou rozděleny na dva grafy. Jeden pro IPv4 a druhý pro IPv6 verze IP adres. Tyto dvě verze spolu nikdy nekomunikují, proto graf závislostí s oběma verzemi IP adres byl

rozdělen na dva podgrafy, které se knihovna *matplotlib* [31] snažila oddělit, čímž způsobila zmenšení obou podgrafů a ty se staly nepřehledné. Kvůli tomu jsme se rozhodli rozdělit graf na verze. Grafy globálních závislostí jsou vytvářeny vždy pro každé zařízení zvlášť, čímž vzniknou grafy typu hvězda. Skript nastavíme pomocí přepínačů při spouštění. Také nám umožňuje vytvořit bipartitní graf, ve kterém se v jedná partitě nacházejí lokální zařízení a ve druhé partitě globální zařízení, přičemž globální zařízení se v grafu nacházejí pouze pokud k nim přistupovalo více lokálních zařízení.

Dalšími důležitými knihovnami jsou *ipaddress* [32] a *sqlite3* [33]. První používáme pro práci s IP adresami a druhou pro připojení a následnou komunikaci s databází.

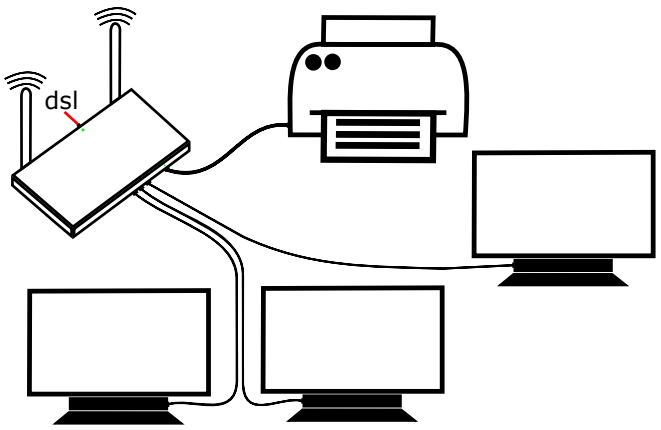
Kapitola 8

Testování

V této kapitole si ukážeme ukázky výstupu vyvinutých nástrojů, které byly nasazeny v sítích pro pilotní testování. Také si předvedeme časové a paměťové náročnosti vyvinutých nástrojů.

8.1 Domácí síť

Nejdříve provedeme testování na malé lokální síti, která se často nachází ve většině domácností. Provedeme měření způsobem přímého zapojení do sítě, jejíž zapojení jsme v upravené podobě vykreslili do Obrázku 8.1 (většina DSL modemů neumí port-mirroring, proto jsme pro měření umístili do sítě switch, který tuto funkcionalitu umožňuje).



Obrázek 8.1: Nákres domácí sítě (vytvořeno Inkscape)

Cílem měření na této konkrétní síti je odhalit z komunikace, zda je v síti tiskárna a počítače. Spustíme náš modul a počkáme, než se nám nasbírají potřebná data. Poté spustíme skript *DeviceAnalyzer* a výsledek vypíšeme do příkazové řádky.

Měření jsme prováděli jen po dobu, než jsme si byli jistí, že proběhla potřebná komunikace k rozpoznání důležitých zařízení. Na Obrázku 8.2 můžeme vidět výstup *DeviceAnalyzer* skriptu pro jeden z počítačů, přičemž globální závislosti byly nezajímavé, a proto jsme je z výstupu smazali. Všimněme si, že náš modul správně určil operační systém počítače a taktéž, dle přístupů na webové a e-mailové služby, rozhodl o tom, že se jedná o koncové zařízení.

```
Device ID:  1
  IP: 192.168.1.100
  Last communication:  Thu Apr  9 04:24:53 2020
  MAC: 00:0e:c6:e1:ce:b9,  Asix Electronics Corp ,  TW
  Labels:
    [ Windows ] -  SMB protocol (Server Message Block)  is a network communication protocol[3]
for providing shared access to files, printers, and serial ports between nodes on a network,
NetBIOS is input output system of MS Windows
    [ DHCP Client ] -  DHCP protocol (Dynamic Host Configuration Protocol-68) using DHCP client
(END DEVICE) for requset IP address from DHCP Server
    [ End Device ] -  PC, Mobile Phone,... (everything that can access web services)
    [ End Device ] -  PC, Mobile Phone,... (everything that can send emails)
  DHCP:
     192.168.1.1  in  Thu Apr  9 04:23:23 2020
     192.168.1.1  in  Thu Apr  9 04:23:14 2020
  Local Dependencies:
    ->  192.168.1.111   provides  [ WEB Server ] - number of packets:  840
    ->  192.168.1.255   requires  [ Windows ] - number of packets:  192
    ->  192.168.1.1   requires  [ ssdp ] - number of packets:  13
    ->  192.168.1.111   requires  [ 0 ] - number of packets:  8
    ->  192.168.1.1   provides  [ DHCP Server ] - number of packets:  3
    ->  192.168.1.255   requires  [ Windows ] - number of packets:  1

HTTP       : ████████████████████████████████████████████████████████████████
68.18
SMB-NetBIOS: ████████████████████████████ 31.33
DHCP       : | 0.49
  Global Dependencies:
    ->  90.182.119.78   provides  [ WEB Server ]  Domain:  78.gcache.iol.cz   - number of
packets:  177605
    ->  13.32.114.8   provides  [ WEB Server ]  Domain:  server-13-32-114-
8.prg50.r.cloudfront.net   - number of packets:  43387
    ->  95.216.163.111   provides  [ WEB Server ]  Domain:  dl4.cdn.filezilla-project.org
number of packets:  8898
    ->  173.194.150.233   provides  [ WEB Server ] - number of packets:  6480
    ->  104.17.167.73   provides  [ WEB Server ] - number of packets:  3463
    ->  172.217.23.246   provides  [ WEB Server ]  Domain:  prg03s06-in-f22.1e100.net   -
number of packets:  3337
    ->  104.27.143.70   provides  [ WEB Server ] - number of packets:  3177
    ->  13.32.105.58   provides  [ WEB Server ]  Domain:  server-13-32-105-
58.prg50.r.cloudfront.net   - number of packets:  1462
    ->  216.58.201.110   provides  [ WEB Server ]  Domain:  prg03s02-in-f110.1e100.net   -
number of packets:  1203
    ->  172.217.23.238   provides  [ WEB Server ]  Domain:  prg03s06-in-f14.1e100.net   -
number of packets:  1192

HTTPS: ████████████████████████████████████████████████████████████████ 99.68
IMAPS: | 0.12
DNS  : | 0.08
FTP  : | 0.06
HTTP : | 0.04
ESMTP: | 0.02
```

Obrázek 8.2: Počítač

Dále si povšimněme, že počítač získával IP adresu od serveru pomocí DHCP, proto má zařízení přiřazen štítek *DHCP Client*. V DHCP záznamech můžeme vidět kdy a od jakého serveru získával IP adresu. V lokálních zařízeních vidíme závislost k *192.168.1.111*, který podle modulu provozoval *WEB Server*. Víme, že přes protokol HTTP se nepřenáší jen webové služby, takže můžeme vytušit, že se nejedná o skutečný webový server. V našem případě víme, že zařízení je switch a pomocí protokolu HTTP se v době měření přistupovalo k nastavení switche přes webové rozhraní.

Na dalším Obrázku 8.3 je vyobrazen výstup pro další zařízení, tentokrát se jedná o router, který zároveň lokální síti poskytoval DHCP a DNS služby. Podstatnou informací jsou IP adresy, pod kterými zařízení v síti vystupovalo.

Posledním zajímavým zařízením v této síti je tiskárna. Výstup pro ni můžeme vidět na Obrázku 8.4. Můžeme si všimnout, že na tiskárně v době měření tisklo pouze jedno zařízení a použilo k domu dva protokoly a to PDP a IPP.

```
########################################################################
Device ID:  3
  IP: 192.168.1.1 < fe80::ce32:e5ff:fea0:4e0c >
  Last communication:  Thu Apr  9 04:24:02 2020
  MAC: cc:32:e5:a0:4e:0c,  TP-Link Tech Co, Ltd ,  CN
  Labels:
    [ DNS Server ] -  DNS protocol (Domain Name System) used to translate IP address to dom
function)
    [ DHCP Server ] -  DHCP protocol (Dynamic Host Configuration Protocol-67) using DHCP se
communication with client (END DEVICE)
    [ Router ] -  Routing network device
  DHCP:
    ---
  Local Dependencies:
    ->  192.168.1.105   requires  [ DNS Server ] - number of packets:  445
    ->  192.168.1.101   requires  [ DNS Server ] - number of packets:  153
    ->  192.168.1.105   requires  [ 50321 ] - number of packets:  120
    ->  192.168.1.100   provides  [ 62233 ] - number of packets:  13
    ->  192.168.1.106   requires  [ DNS Server ] - number of packets:  12
    ->  192.168.1.105   requires  [ 0 ] - number of packets:  3
    ->  192.168.1.100   provides  [ DHCP Client ] - number of packets:  3
    ->  255.255.255.255   requires  [ DHCP Server ] - number of packets:  1

DNS : ████████████████████████████████████████████████████████████████ 98.71
DHCP: | 1.29

  Global Dependencies:
    ---
```

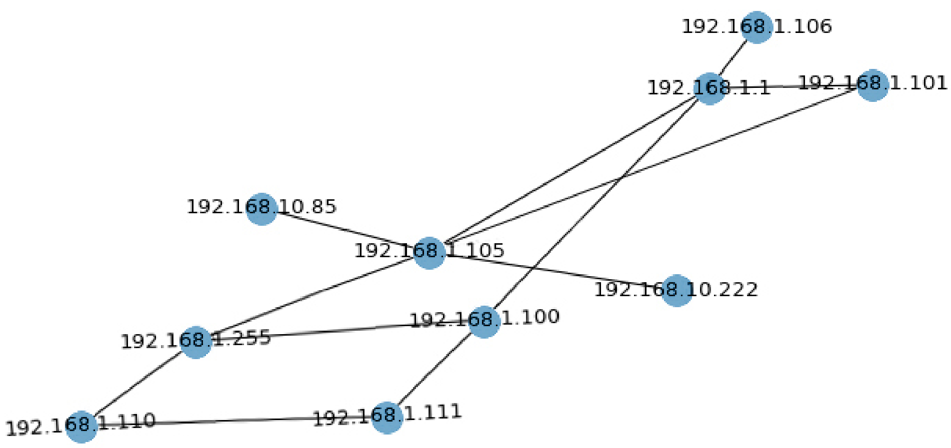
Obrázek 8.3: Router

Na Obrázku 8.5 můžeme vidět graf závislostí na lokální síti, který modul vytvořil na základě pozorované komunikace.

```
########################################################################
Device ID:  8
  IP: 192.168.1.101
  Last communication:  Thu Apr  9 04:24:20 2020
  MAC: 20:2b:c1:9a:d8:c4,  Huawei Tech Co, Ltd ,  CN
  Labels:
    [ Printer ] -  Printer PDP Data Stream, AppSocket use for communication with printer or printer servers,
Hewlett-Packard (HP) JetDirect network printing service
    [ Printer ] -  IPP protocol (Internet Printing Protocol) use for communication with printer or printer
servers
    [ End Device ] -  PC, Mobile Phone,... (everything that can access web services)
    [ End Device ] -  PC, Mobile Phone,... (everything that can send emails)
  DHCP:
    ---
  Local Dependencies:
    ->  192.168.1.105   requires  [ Printer ] - number of packets:  11870
    ->  192.168.1.105   requires  [ Printer ] - number of packets:  624
    ->  192.168.1.1   provides  [ DNS Server ] - number of packets:  153

IPP                      : ████████████████████████████████████████████████████████████████ 93.86
Printer_PDP_Data_Stream: ███ 4.93
DNS                      : | 1.21
```

Obrázek 8.4: Tiskárna



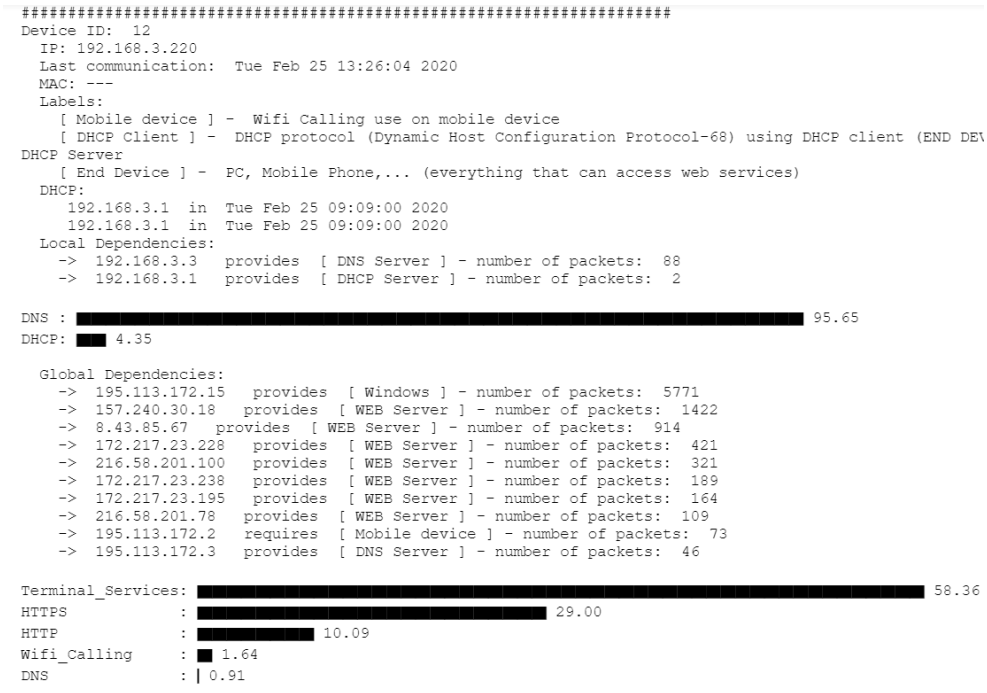
Obrázek 8.5: Graf lokálních závislostí pro IPv4 adresy

8.2 Síť v kanceláři

Sítě v kancelářích většinou obsahují pracovní počítače, tiskárny a zařízení pro sdílení souborů, jak souborové servery, tak sdílené disky. Ze serverů se objevují DHCP a DNS servery, které podporují činnosti počítačů. Ve větších firemních sítích se samozřejmě objevují nejrůznější autentizační, souborové, webové či e-mailové servery, ke kterým přistupují jak zařízení z lokální sítě, tak zařízení z internetu.

Pro toto testování jsme naměřili síť laboratoře monitorování síťového provozu na Fakultě Informačních Technologií ČVUT a vytvořili záznam v souboru *.trapcap*, ale v záznamu nemáme informace o MAC adresách. Nyní si ukážeme výstup o vybraných zajímavých zařízeních.

Na Obrázku 8.6 vidíme výstup o zařízení, které přistupuje na webové služby, přijímá IP adresu prostřednictvím DHCP protokolu a zároveň používá protokol typický pro volání přes internet pro mobilní zařízení. Z toho snadno odhadneme, že zařízení je pravděpodobně chytrý mobilní telefon připojený k Wi-Fi.



Obrázek 8.6: Mobilní zařízení

Mobilní telefon v lokální síti komunikuje se dvěma různými zařízeními, a to s DHCP Serverem s IP adresou končící na *.1*, který mu přidělil IP adresu a DNS Serverem s IP adresou končící na *.3*, který DNS dotazy dále distribuuje do internetu. Při pohledu na výpisy těchto dvou zařízení zjistíme, že zařízení

.1 zároveň poskytuje DNS službu a to právě pro zařízení *.3*, které je DNS Serverem pro většinu zařízení v síti, proto je pravděpodobné, že zařízení *.1* je routerem. Nicméně bez MAC adres náš modul nemohl rozhodnout, že je toto zařízení routerem i když tomu tak pravděpodobně je. Na Obrázku 8.7 můžeme vidět výstup zařízení *.3*, tedy DNS Serveru. Z něj si snadno všimneme, že hlavní globální závislost je s IP adresou *1.1.1.1*, která je známá poskytováním DNS služeb.

```
########################################################################
Device ID:  8
  IP: 192.168.3.3
  Last communication:  Tue Feb 25 13:28:26 2020
  MAC: ---
  Labels:
    [ DNS Server ] -  DNS protocol (Domain Name System) used to translate IP address to doma
    [ End Device ] -  PC, Mobile Phone,... (everything that can access web services)
  DHCP:
    ---
  Local Dependencies:
    ->  192.168.3.1   requires  [ DNS Server ] - number of packets:  2061
    ->  192.168.3.1   requires  [ 0 ] - number of packets:  480
    ->  192.168.3.150   requires  [ DNS Server ] - number of packets:  378
    ->  192.168.3.220   requires  [ DNS Server ] - number of packets:  88
    ->  192.168.3.248   requires  [ DNS Server ] - number of packets:  38144
    ->  192.168.3.204   requires  [ DNS Server ] - number of packets:  8
    ->  192.168.3.219   requires  [ DNS Server ] - number of packets:  12
    ->  192.168.3.151   requires  [ DNS Server ] - number of packets:  410
    ->  192.168.3.151   requires  [ 0 ] - number of packets:  31
    ->  192.168.3.248   requires  [ 0 ] - number of packets:  7

DNS: ████████████████████████████████████████████████████████████ 100.00

  Global Dependencies:
    ->  1.1.1.1   provides  [ WEB Server ] - number of packets:  44462
    ->  140.82.118.5   provides  [ WEB Server ] - number of packets:  103
    ->  109.248.43.36   provides  [ Time Server ] - number of packets:  28

HTTPS: ████████████████████████████████████████████████████████████ 99.94
NTP  : | 0.06
```

Obrázek 8.7: DNS Server

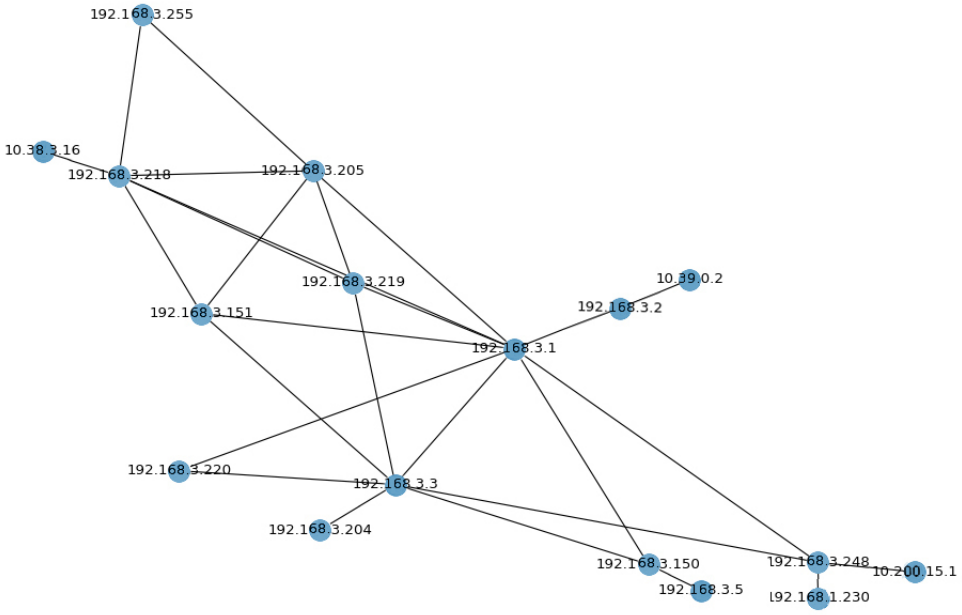
Na Obrázku 8.8 je vykresleno používání sítě zařízeními v procentech, ze kterého přehledně vidíme, jaká zařízení v síti přenesli nejvíce paketům. Síť nejvíce používaly osobní počítače.

```
########################################################################
  Print Statistic of using network by devices in %:

192.168.3.2                    : ████████████████████████████████████████ 36.96
192.168.3.205                  : ████████████████████████████████████████ 36.86
192.168.3.248                  : ███████████████████ 17.90
192.168.3.150                  : ██████ 4.32
192.168.3.5                    : ██ 1.61
192.168.3.3                    : | 0.99
192.168.3.218                  : | 0.39
192.168.1.230                  : | 0.27
192.168.3.219                  : | 0.25
10.200.15.1                    : | 0.12
192.168.3.220                  : | 0.11
192.168.3.1                    : | 0.11
```

Obrázek 8.8: Statistika používání sítě

Na Obrázku 8.9 je vykreslen graf závislostí pro zařízení s IP adresou verze 4. Z něj vidíme, že nejvytíženější zařízení v síti je zařízení s IP adresou končící na *.1*, jejž jsme ručně označili za router a zařízení s IP adresou končící na *.3*, které je DNS Serverem.

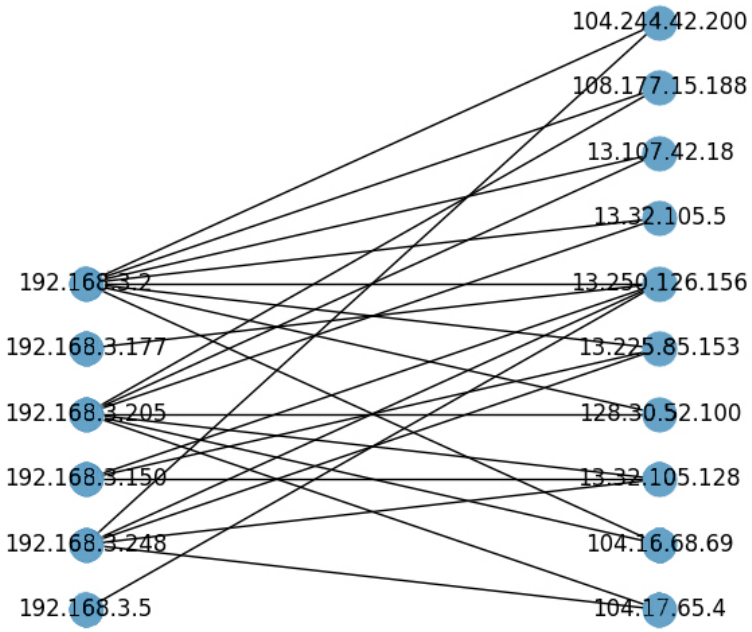


Obrázek 8.9: Graf lokálních závislostí IPv4

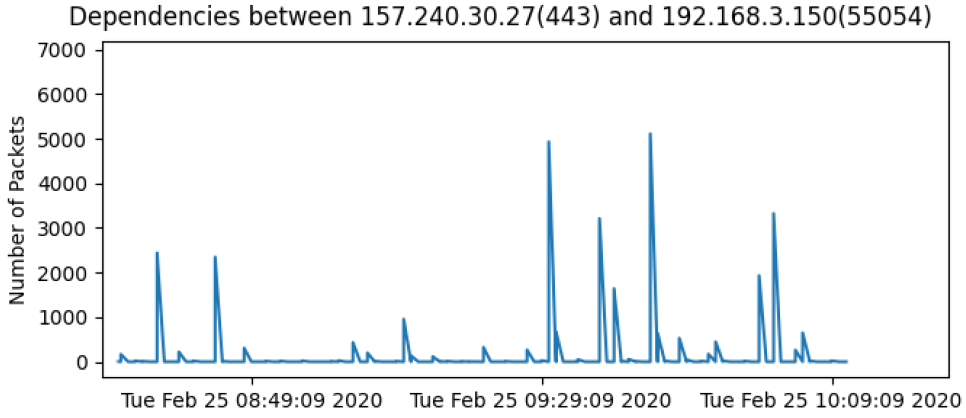
Na Obrázku 8.10 je vykreslen graf závislostí mezi lokálními (těmi zařízeními, které modul považuje za lokální) a globálními zařízeními. Přičemž globální zařízení se do grafu dostanou pouze tehdy pokud k nim přistoupí více lokálních zařízení. Tím vznikne bipartitní graf, kde na levé straně jsou lokální zařízení a na pravé straně jsou globální zařízení.

Dále můžeme při měření použít parametr *-T*, kterým zapneme práci s časem u závislostí, takže modul vyplní tabulky *DependenciesTime* a *GlobalTime* časovými údaji o závislostech z tabulek *Dependencies* a *Global*. Výstup z analýzy *DeviceAnalyzer* skriptu je poté vyobrazen na Obrázku 8.11, na kterém můžeme vidět přenesené pakety v závislosti na čase.

Stejný typ grafu, který ukazuje aktivitu zařízení v době měření, můžeme vytvořit pomocí parametru *-A*.



Obrázek 8.10: Bipartitní graf závislostí mezi lokálními a globálními zařízeními



Obrázek 8.11: Graf používání závislosti v čase

8.3 Globální síť

Nakonec pro testování zkusme náš modul vhodně nastavit tak, aby co nejlépe fungoval na globální síti. Pro tento účel jsme ve spolupráci se sdružením CESNET, operátorem národní akademické infrastruktury, získali anonymizované záznamy komunikace rozdělené po hodinách z globálních rozsahů ČVUT (*147.32.232.0/24*), VUT (*147.229.13.0/24*) a CESNET (*195.113.172.0/24*). Veřejné IP adresy byly ve výstupu zanonymizovány.

Jelikož je náš modul primárně určen na práci na lokálních sítích, tak musíme modulu nastavit IP adresy sítí, které chceme sledovat. Modul proto spustíme takto:

```
# ./PassiveAutodiscovery.py -i f:GlobalRecord.trapcap -d Database
-N 147.32.232.0/24 147.229.13.0/24 195.113.172.0/24 -! -G -U
```

kde v parametru *-N* nastavíme sítě pro monitorování (tyto bude modul považovat jako lokální), díky parametru *-!* vynutíme práci pouze s vypsanými sítěmi a parametr *-U* nám slouží k mapování pouze „běžně“ používaných protokolů.

Na Obrázku 8.12 je vyobrazen výstup *DeviceAnalyzer* skriptu pro jedno vybrané zařízení.

```
######################################################################
DeviceID:  9
  IP:  [redacted]
  Last communication:  Sun Mar 15 19:59:26 2020
  MAC: ---
  Labels:
   [ WEB Server ]  -  HTTP protocol (Hypertext Transfer Protocol) use for Web serve
HTTP for tranfer data
   [ File Server ]  -  FTP protocol (File Transfer Protocol) use for file transfer
   [ Streaming Server ]  -  RTMP protocol (Real Time Messaging Protocol) use by Fla
Server
   [ WEB Server ]  -  HTTPS protocol using for web servers (http over tls/ssl)
   [ File Server ]  -  Andrew File System (AFS) port for callbacks to cache manage
   [ Database Server ]  -  Microsoft SQL Server
   [ File Server ]  -  TFTP protocol (Trivial File Transfer Protocol) use for file
   [ File Server ]  -  NFS protocol (Network File System) use for file transfer
   [ Proxy Server ]  -  Transparent Proxy
   [ Mail Server ]  -  POP3S protocol (Post Office Protocol) use for mail services
   [ VoIP telephone ]  -  SIP protocol (Session Initiation Protocol) use for telep
   [ Windows ]  -  SMB protocol (Server Message Block)  is a network communication
printers, and serial ports between nodes on a network, NetBIOS is input output syst
   [ WEB Server ]  -  HTTP protocol (Hypertext Transfer Protocol) use for Web serve
HTTP for tranfer data (Alternate port)
   [ Dictionary Server ]  -  LDAP protocol (Lightweight Directory Access Protocol)
time safe data)
   [ File Server ]  -  NFS protocol (Network File System) use for file transfer
```

Obrázek 8.12: Zařízení v globální síti

Okamžitě si všimneme velkého počtu štítků, přičemž závislosti zodpovědné za tyto štítky mají většinou malý počet paketů. Z toho vyvodíme, že tyhle závislosti jsou vytvořené roboty, které prohledávají aktivním způsobem globální rozsahy IP adres za účelem zjišťování, jaké služby zařízení na daných IP adresách poskytují. Abychom tyto závislosti odstranili a tím zpřesnili měření,

musíme při spouštění našeho modulu přidat parametr *-D* spolu s počtem paketů, jenž slouží jako hranice pro vymazání závislosti.

Na Obrázku 8.13 je vyobrazen výstup analýzy při periodickém mazaní globálních závislostí.

```
######################################################################
Device ID:  9
  IP: [redacted]
  Last communication:  Sun Mar 15 19:59:26 2020
  MAC: ---
  Labels:
    [ WEB Server ] -  HTTP protocol (Hypertext Transfer Protocol) use for Web server (unen
HTTP for tranfer data
    [ File Server ] -  FTP protocol (File Transfer Protocol) use for file transfer (comman
    [ Streaming Server ] -  RTMP protocol (Real Time Messaging Protocol) use by Flash Play
Server
    [ WEB Server ] -  HTTPS protocol using for web servers (http over tls/ssl)
  DHCP:
    ---
  Local Dependencies:
    ---

  Global Dependencies:
    ->  [redacted] requires  [ Streaming Server ] - number of packets:  3232
    ->  [redacted]  provides  [ WEB Server ] - number of packets:  306
    ->  [redacted] requires  [ WEB Server ] - number of packets:  148
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  137
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  118
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  64
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  58
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  57
    ->  [redacted] requires  [ WEB Server ] - number of packets:  56
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  55
    ->  [redacted] requires  [ WEB Server ] - number of packets:  55
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  54
    ->  [redacted]requires  [ WEB Server ] - number of packets:  52
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  52
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  51
    ->  [redacted]  requires  [ WEB Server ] - number of packets:  51

RTMP : [redacted] 49.77
HTTP : [redacted] 42.90
FTP  : [redacted] 4.71
HTTPS: [redacted] 2.62
```

Obrázek 8.13: Zařízení v globální síti při mazání závislostí

Mazání závislostí s pěti a méně pakety výrazně snížilo počet štítků. Ale periodické mazání závislostí ne vždy bude fungovat. Záleží na zvoleném počtu paketů a také komunikaci, která probíhá během jedné periody (10 000 IP flows). Pokud v jedné periodě bude jeden robot vícekrát analyzovat dané zařízení, potom závislosti tím vzniklé mohou mít více paketů než je zvolené hranice. Tím pádem budou štítky nepřesné.

Proto zkusíme analyzovat každý IP flow s cílem pracovat pouze s úplnými závislostmi. To znamená, že pokud jedno zařízení komunikuje směrem na druhé zařízení pod nějakým specifickým protokolem z tabulky *Services* a druhé zařízení na jeho požadavky neodpovídá, vznikne neúplná závislost. Přidáním parametru *-F* spustíme filtrování IP flows. Výsledek filtrování TCP a UDP provozu můžeme vidět na Obrázku 8.14

```
  Labels:
    [ WEB Server ] -  avc.fit.cvut.cz
    [ Streaming Server ] -  RTMP protocol (Real Time Messaging Protocol) use by Flash Player
streaming server  Adobe Media Server
    [ File Server ] -  FTP protocol (File Transfer Protocol) use for file transfer (command)
    [ Database Server ] -  Microsoft SQL Server
    [ Windows ] -  SMB protocol (Server Message Block)  is a network communication protocol[
shared access to files, printers, and serial ports between nodes on a network, NetBIOS is in
MS Windows
    [ End Device ] -  PC, Mobile Phone,... (everything that can send emails)
  DHCP:
    ---
  Local Dependencies:
    ---

  Global Dependencies:
    ->  [redacted]  requires  [ 45688 ] - number of packets:  110285
    ->  [redacted]  requires  [ Streaming Server ] - number of packets:  3146
    ->  [redacted]  requires  [ 11919 ] - number of packets:  469
    ->  [redacted]  requires  [ 56161 ] - number of packets:  461
    ->  [redacted]  requires  [ 52054 ] - number of packets:  303
    ->  [redacted]  provides  [ WEB Server ] - number of packets:  301
```

Obrázek 8.14: Zařízení v globální síti při filtrování závislostí

Ze štítků z Obrázků 8.13 a 8.14 a ze závislostí můžeme odhadnout, že dané zařízení je server poskytující webový server, na kterém lze přehrávat videa za použití *RTMP* protokolu.

Ověříme si výsledek přeložením IP adresy na doménové jméno a získáme *avc.fit.cvut.cz*. Což je web zaměřený na nahrané přednášky Fakulty Informačních Technologií na ČVUT, které lze na tomto webu přehrát. Štítkování v tomto konkrétním příkladě bezpečně odhalilo úlohu zařízení na této IP adrese.

V dalších příkladech můžeme uvést zařízení, kterému náš modul přiřadil štítky *WEB Server* a *Mail Server*. Doménové jméno tohoto zařízení je *imap.fit.cvut.cz*, což je webový e-mailový klient FIT ČVUT. Dále jsme nalezli zařízení se štítky *Router* a *End Device*, jenž ukazuje na router, za kterým se nachází lokální síť.

8.4 Časová náročnost

8.4.1 PassiveAutodiscovery modul

V našem případě časová náročnost či složitost neurčuje, za jak dlouho program doběhne, ale spíše ukazuje jak dlouho a při jakém množství komunikace je modul schopen pracovat při běhu na síti.

Pro odhadnutí časové náročnosti nám velice dobře postačí hodinový záznam globální sítě popsané v předchozí podkapitole. V těchto záznamech se nachází 418 647 IP flows a v naší tabulce *Global*, kde se nachází pouze jedinečné závislosti, to vyplní přes 80 000 záznamů. Objem komunikace za tuto hodinu je tedy značný a bude postačovat pro odhadnutí časových náročností našeho modulu při různých nastaveních pomocí parametrů. Měření jsme prováděli na notebooku HP ProBook 4720s s procesorem Intel i3 M 350 2.2 GHz, RAM pamětí 4 GiB DDR3 a s SSD diskem s rychlostí čtení 500 MB/s a zápisu 320 MB/s.

Jak jsme již řešili v kapitole *Návrh* zvolená databáze *sqlite3* je oproti klasickým databázovým serverům pomalejší, jelikož po každém vložení či úpravě záznamu v databázi čeká, než se krok propíše do souboru *.db* představující databázi. Z toho důvodu řádně otestujeme, zda a jak to omezuje měření v síti, a výsledky měření porovnáme s variantou průběžného ukládání celé *sqlite3* databáze do RAM paměti. Tato varianta by měla urychlit vkládání a úpravy záznamů v databázi.

Zahájili jsme na tomto hodinovém záznamu měření s cílem zjistit časovou náročnost pro různé nastavení modulu pomocí parametrů. Výsledky všech měření jsou přehledně zobrazené v Tabulce 8.1. Všimněme si, že nejvíce časovou náročnost zlepšuje parametr *-RAM*, díky kterému se v průběhu měření celá *sqlite3* databáze ukládá do RAM paměti a do souboru se zapíše naráz na konci měření. Dalším parametrem, který měření urychlí je *-D*, který průběžně promazává tabulku *Global* od závislostí, které mají menší počet paketů než je zadaný počet. Tím může uživatel smazat „nepodstatné“ závislosti.

Když jsme testovali modul v reálných lokálních sítích, tak jsme nezaznamenali s měřením žádné časové problémy, ale objem komunikace na těchto sítích nebyl velký, a tak jsme si mohli dovolit informační výpisy z *PassiveAutodiscovery.py*. Avšak na tomto velkém hodinovém záznamu trvají pouhé výpisy nově nalezených zařízení, závislostí a dalších informací přes jednu hodinu. Při používání modulu na velkém provozu je proto nutné zvážit vypnutí výpisů pro zvýšení výkonu. Výkonnostní optimalizace modulu jsou nad rámec této bakalářské práce a jsou ponechány jako budoucí práce.

Tabulka 8.1: Tabulka měření času zpracování v hodinách a minutách v závislosti na zvolených parametrech.

Způsob měření	**Přepínače modulu**	**Čas měření**
S informačními výpisy	-G -l -s -L -m -S -g	3:30
Bez informačních výpisů	-G	2:37
Pouze „běžné“ porty	-G -U	1:20
S odstraňováním globálních závislostí	-G -U -D 5	0:47
Bez ukládání globálních závislostí	-U	0:37
Pouze „běžné“ porty s používáním RAM paměti	-G -U -RAM	0:24
Filtrování neúplných závislostí s používáním RAM paměti	-G -F -RAM	0:21
S odstraňováním globálních závislostí a s používáním RAM paměti	-G -U -D 5 -RAM	0:13
Bez ukládání globálních závislostí a s použitím RAM paměti	-U -RAM	0:05

8.4.2 DeviceAnalyzer

Čas běhu *DeviceAnalyzer* skriptu záleží na počtu záznamů v databázi a na použitých parametrech, protože některé výrazně zpomalí běh analýzy. Pro ukázku, jak je časová náročnost ovlivňována parametry, jsme použili záznam kancelářské sítě z druhého testování. Výsledky měření jsme zapsali do přehledné Tabulky 8.2.

Z tabulky vidíme, že nejvíce náročným parametrem je *-DNS*, který překládá každou IP adresu označenou štítkem *WEB Server* na doménové jméno, pokud to lze. Dalším výrazným parametrem je *-G číslo* (respektive *-L číslo*), který ve výstupu omezuje počet globálních (respektive lokální) závislostí. Čímž zlepší časovou náročnost.

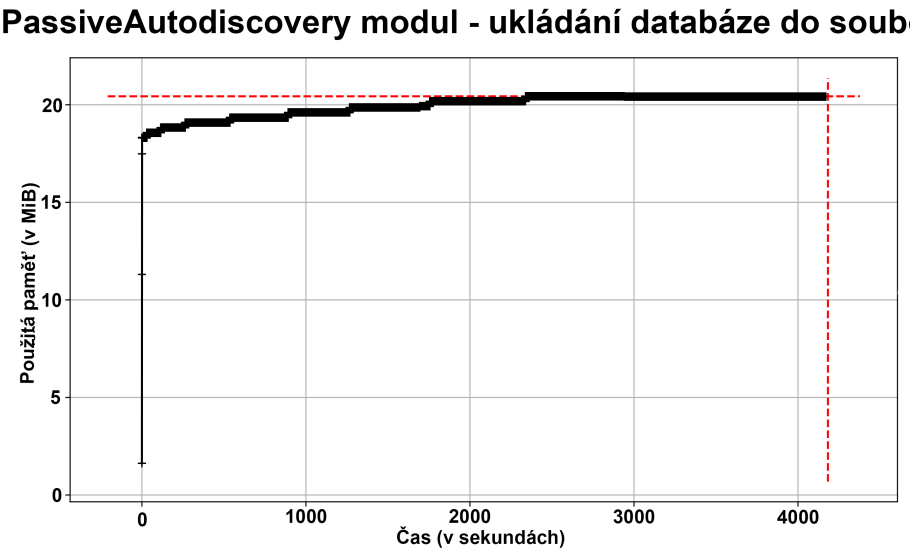
Tabulka 8.2: Tabulka měření času analyzování v hodinách a minutách v závislosti na zvolených parametrech.

Způsob měření	**Přepínače modulu**	**Čas měření**
Překlad DNS a neomezené globální závislosti s výstupem do příkazové řádky a tvorbou grafů	-p -DNS -l -g	5 m 31.12 s
Překlad DNS a neomezené globální závislosti s výstupem do souboru a tvorbou grafů	-f file -DNS -l -g	5 m 11.12 s
Překlad DNS a neomezené globální závislosti s výstupem do příkazové řádky	-p -DNS	4 m 52.34 s
Překlad DNS a neomezené globální závislosti s výstupem do souboru	-f file -DNS	4 m 48.46 s
Překlad DNS a omezené globální závislosti s výstupem do příkazové řádky a tvorbou grafů	-p -DNS -l -g -G 15	2 m 49.68 s
Překlad DNS a omezené globální závislosti s výstupem do souboru a tvorbou grafů	-f file -DNS -l -g -G 15	2 m 36.87 s
Omezené globálních závislostí s výstupem do příkazové řádky a tvorbou grafů	-p -l -g -G 15	2 m 43.16 s
Omezené globálních závislostí s výstupem do souboru a tvorbou grafů	-f file -l -g -G 15	2 m 42.30 s
Neomezené globálních závislostí s výstupem do příkazové řádky a tvorbou grafů	-p -l -g	2 m 17.93 s
Neomezené globálních závislostí s výstupem do souboru a tvorbou grafů	-f file -l -g	2 m 12.92 s
Pouze lokální závislosti s výstupem do příkazové řádky	-p -o	2.45 s
Pouze lokální závislosti s výstupem do souboru	-f file -o	2.28 s

8.5 Paměťová náročnost

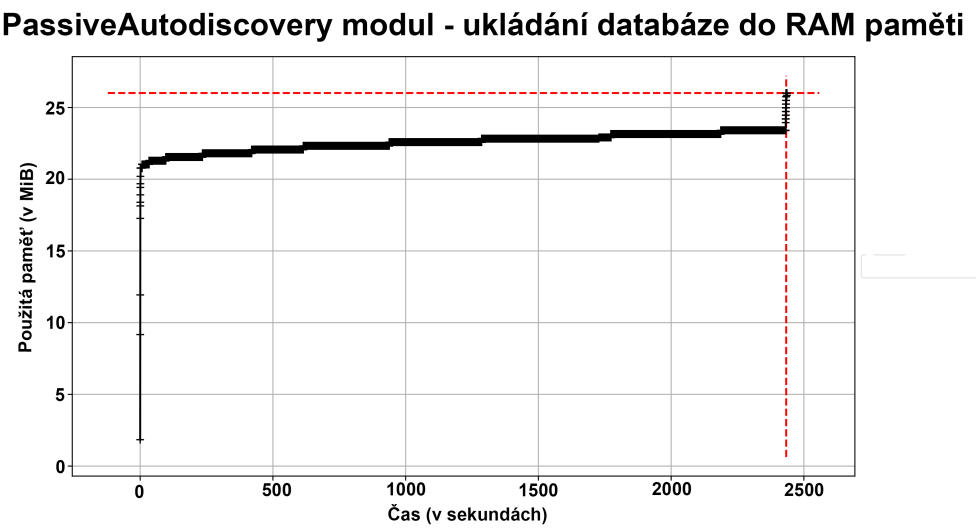
8.5.1 PassiveAutodiscovery modul

Paměťová náročnost *PassiveAutodiscovery* modulu je závislá na způsobu uložení *sqlite3* databáze a počtu záznamů do ní uložených. Pro odhadnutí a vykreslení této náročnosti jsme použili záznam globální sítě z předchozí ukázky. Na Obrázku 8.15 vidíme, že paměťová náročnost při uložení databáze do souboru je rostoucí v závislosti na počtu vložených dat do databáze. V druhé polovině měření již velká část zařízeních, služeb a závislostí v databázi již existovalo, proto množství alokované paměti stoupalo pomaleji.



Obrázek 8.15: PassiveAutodiscovery modulu s uložením databáze v souboru

Na druhém Obrázku 8.16 je naopak znázorněna paměťová náročnost při ukládání databáze do RAM paměti. Na první pohled si všimneme, že se využití paměti zvýšilo o velikost databáze a na konci běhu modulu využití vzrostlo. To je způsobené tím, že se na konci přenáší databáze z RAM paměti do souboru. Paměťová náročnost po většinu doby je stejně jako při ukládání databáze do souboru rostoucí v závislosti na počtu vložených dat do databáze.



Obrázek 8.16: Paměťová náročnost PassiveAutodiscovery modulu s uložením databáze v RAM paměti

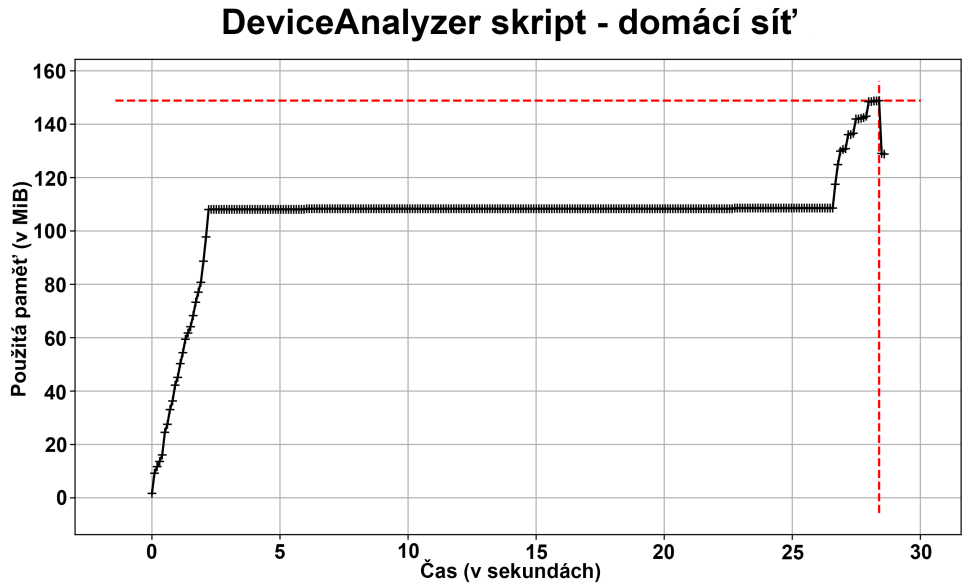
8.5.2 DeviceAnalyzer

U *DeviceAnalyzer* skriptu ukážeme vývoj paměťové náročnosti na dvou grafech, ve kterých porovnáme využití paměti v závislosti na velikosti analyzované databáze.

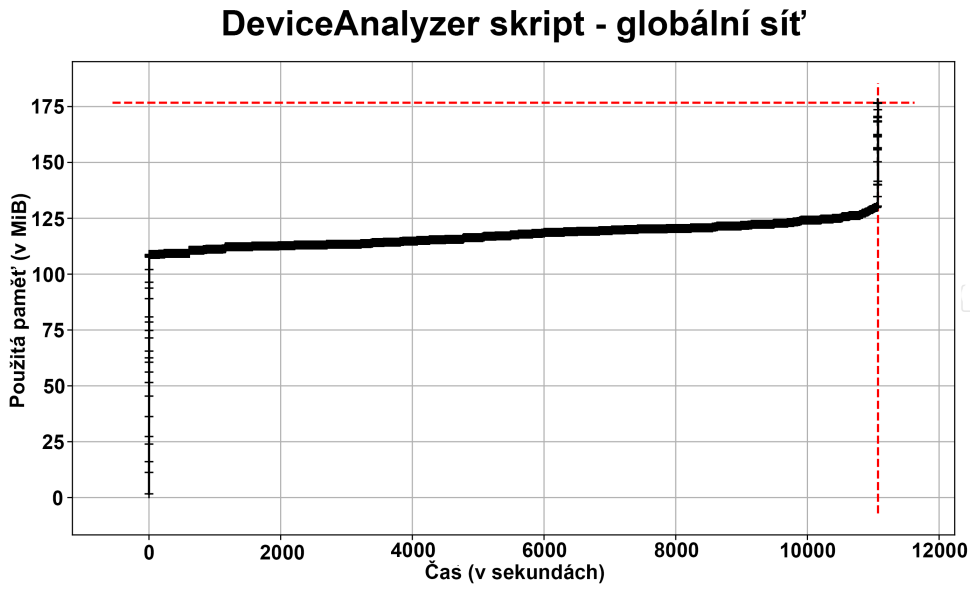
Na Obrázku 8.17 je vykresleno využití paměti při analýze domácí sítě z první ukázky testování. Všimněme si, že většinu běhu skriptu byla náročnost konstantní a na konci běhu se zvýšila při vykreslování grafů závislostí a statistiky využití sítě zařízeními.

Na druhém Obrázku 8.18 je k porovnání zobrazen vývoj paměťového využití skriptem při analýze globální sítě z druhé ukázky, která v porovnání s druhým modulem má mnohonásobně větší vstupní databázi k analýze. Zajímavostí je vývoj, který je během běhu lehce rostoucí a na konci skriptu je taktéž zvýšení při vykreslování grafů a statistiky.

Přestože vstupní databáze byla v druhém příkladě mnohanásobně větší, tak se paměťové využití příliš nezvýšilo a ono zvýšení je způsobeno počtem zařízení a informací o nich, které se průběžně ukládají do výstupního JSON dokumentu. Z toho vyvodíme, že vstupní databáze příliš neovlivňuje paměťovou náročnost a paměťová náročnost je lehce rostoucí jelikož se výstup DeviceAnalyzer skriptu průběžně ukládá do JSON dokumentu, který je průběžně ukládán do RAM paměti.



Obrázek 8.17: Paměťová náročnost DeviceAnalyzer skriptu na domácí síti



Obrázek 8.18: Paměťová náročnost DeviceAnalyzer skriptu na globální síti

Závěr

Naším hlavním cílem bylo vytvořit seznam štítků, které určují roli zařízení v síti, a tyto štítky následně automaticky přiřazovat fyzickým zařízením na základě jejich provozu na síti. Tímto způsobem je možné zmapovat libovolnou síťovou infrastrukturu na základě pasivně pozorovaného síťového provozu. Dále jsme měli zajistit ukládání závislostí mezi zařízeními a ty následně vizualizovat pomocí grafů.

V rámci této bakalářské práce byl vytvořen prototyp softwarového modulu, který dokáže na základě síťových toků zmapovat aktivní zařízení, získat o nich detailnější informace a rozpoznávat závislostí mezi poskytovanými službami a klienty na síti. Prototyp byl implementován jako modul do modulárního systému NEMEA. Výsledná aplikace byla následně otestována na reálném síťovém provozu a proběhlo měření výkonu a potřebné alokované paměti.

Z dostupných zdrojů se nám podařilo vytvořit seznam štítků rolí s jejich specifickými či často používanými protokoly transportní vrstvy TCP/IP modelu. Ve vytvořeném seznamu je přes 40 různých štítků a přes 250 protokolů k nim přiřazeným.

Navrhli a implementovali jsme databázi, která obsahuje počáteční údaje pro náš modul včetně seznamu štítků. Do této databáze náš modul ukládá nasbíraná data, se kterými následně pracuje skript pro analýzu. Díky spolupráci s databází jsme dokázali v modulu implementovat přiřazování štítků jednotlivým zařízením pomocí jednoduchých dotazů, evidovat závislostí mezi zařízeními a jejich přístupy do internetu či záznamy o DHCP komunikaci. Vytvořili jsme skript, který naší databázi projde a vytvoří výstupní analýzu. Ta předně obsahuje pro každé fyzické zařízení přiřazené štítky a závislosti s ostatními zařízeními. Skript taktéž vytvoří graf závislostí mezi zařízeními, který uloží do souboru.

Modul je primárně určený pro lokální sítě, ale vhodně jsme ho rozšířili, aby byl schopen fungovat i na sítích globálních, přičemž výsledky testování na globálních sítích jsou dostačující pro přehled o rolích daných zařízení. Naše zvolená databáze má omezení, které způsobuje pomalejší vkládání a upravo-

vání dat v databázi, ale dokázali jsme implementovat ukládání do RAM paměti, které toto omezení zmenšuje, přičemž jsme při měření časové složitosti dokázali jeho funkčnost.

Možný budoucí vývoj

V rámci projektu ADiCT (Asset Discovery Classification Tagging) pod sdružením CESNET bude vytvořená centrální databáze, která bude shromažďovat informace o zařízeních (především serverech) v síti. Moduly vyvinuté v rámci této bakalářské práce se stanou jedním ze zdrojů primárních dat pro znalostní databázi. Proto proběhne v budoucnu předělání tohoto modulu pro práci s touto nově vzniklou jednotnou databází.

Ve skriptu pro analýzu jsme implementovali výstup ve formátu JSON dokumentu s cílem dlouhodobého uložení analýzy do souboru, se kterým může snadno pracovat jakýkoliv program či skript. Je předlohou pro možnou budoucí aplikaci, která by z tohoto výstupního JSON dokumentu mohla graficky prezentovat získané informace o zařízeních.

Literatura

[1] Hofstede R., Čeleda P., Trammell B., Drago I., Sandre R., Sperotto A. a Pras A, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX". [online] IEEE Communications Surveys Tutorials. Dostupné z: https://is.muni.cz/repo/1181098/flow-monitoring-explained-paper.pdf

[2] Cejka T., V. Bartos, M. Svepes, Z. Rosa a H. Kubatova, "NEMEA: A framework for network traffic analysis". [online]. 2016 12th International Conference on Network and Service Management (CNSM), Montreal, QC, 2016, pp. 195-201.

[3] Cisco Identity Service Engine [online]. Cisco Systems, Inc. [cit. 2020-04-20]. Dostupné z: https://www.cisco.com/c/en/us/products/security/identity-services-engine/index.html

[4] Cisco Systems, Inc. [online]. Cisco Identity Services Engine, Asset Visibility. 2019. [cit. 2020-04-20]. Dostupné z: https://www.cisco.com/c/dam/en/us/products/collateral/security/network-visibility-segmentation/ise-asset-visibility-aag.pdf

[5] Solarwinds.com [online]. SolarWinds Worldwide, LLC. [cit. 2020-03-21]. Dostupné z: https://www.solarwinds.com/user-device-tracker

[6] PRTG Network Monitor [online]. Paessler AG. [cit. 2020-03-21]. Dostupné z: https://www.paessler.com/monitoring

[7] PRTG Automatic discovery [online]. Paessler AG. [cit. 2020-03-21]. Dostupné z: https://www.paessler.com/network-discovery-tool

[8] ManageEngine OpManager Ethernet Monitor [online]. ManageEngine. [cit. 2020-03-22]. Dostupné z: https://www.manageengine.com/network-monitoring/ethernet-monitoring.html

[9] ManageEngine OpManager LAN Monitor [online]. ManageEngine. [cit. 2020-03-22]. Dostupné z: https://www.manageengine.com/network-monitoring/lan-monitoring.html

[10] ManageEngine OpManager LAN Monitor [online]. ManageEngine. [cit. 2020-03-22]. Dostupné z: https://www.manageengine.com/network-monitoring/network-device-discovery.html

[11] Cacti [online]. Cacti. [cit. 2020-03-22]. Dostupné z: https://www.cacti.net/

[12] Zenmap [online]. Nmap. [cit. 2020-03-22]. Dostupné z: https://nmap.org/zenmap/

[13] Spiceworks Inventory [online]. Spiceworks. [cit. 2020-03-24]. Dostupné z: https://community.spiceworks.com/support/inventory/

[14] Barish Stephen. Passive Network Analysis. Community Broadcom. [online]. Broadcom, 2007. [cit. 2020-03-26]. Dostupné z: https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=0541c345-7909-4682-8b31-6948f42571a9&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments

[15] IANA Assignments Ports. [online]. IANA. [cit. 2020-04-01]. Dostupné z: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[16] Villanueva John Carl. 12 File Transfer Protocols & The Businesses They're Best Suited For. [online]. jscape. [cit. 2020-04-16]. Dostupné z: https://www.jscape.com/blog/12-file-transfer-protocols-businesses

[17] RFC 1350: The TFTP Protocol (Revision 2). Dostupné z: https://www.rfc-editor.org/rfc/rfc1350.txt

[18] RFC 1813: NFS Version 3 Protocol Specification. Dostupné z: https://www.rfc-editor.org/rfc/rfc1813.txt

[19] RFC 3965: A Simple Mode of Facsimile Using Internet Mail. Dostupné z: https://www.rfc-editor.org/rfc/rfc3965.txt

[20] Mehl Bernhard. Authentication Protocols: LDAP vs Kerberos vs OAuth2 vs SAML vs RADIUS. [online]. kisiblog. [cit. 2020-04-16]. Dostupné z: https://www.getkisi.com/blog/authentication-protocols-overview

[21] RFC 4120: The Kerberos Network Authentication Service (V5). Dostupné z: https://www.rfc-editor.org/rfc/rfc4120.txt

[22] RFC 5997: Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol. Dostupné z: https://www.rfc-editor.org/rfc/rfc5997.txt

[23] RFC 2649: An LDAP Control and Schema for Holding Operation Signatures. Dostupné z: https://www.rfc-editor.org/rfc/rfc2649.txt

[24] RFC 2566: Internet Printing Protocol/1.0: Model and Semantics. Dostupné z: https://www.rfc-editor.org/rfc/rfc2566.txt

[25] RFC 1179: Line printer daemon protocol. Dostupné z: https://www.rfc-editor.org/rfc/rfc1179.txt

[26] Grech Matt. The Comprehensive Guide To Understanding VoIP Protocols and Standards. [online]. GETVOIP. [cit. 2020-04-16]. Dostupné z: https://getvoip.com/blog/2017/03/03/voip-protocols-and-standards/

[27] Sivanathan Arunan, Hassan Habibi Gharakheili, a Vijay Sivaraman. Can We Classify an IoT Device using TCP Port Scan?. [online]. School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia, 2018. [cit. 2020-03-28]. Dostupné z: http://www2.ee.unsw.edu.au/ hhabibi/pubs/conf/18iciafs-1.pdf

[28] Python Software Foundation. Urllib2. [software]. Dostupné z: https://docs.python.org/3/library/urllib.html

[29] Marcus Kazmierczak. Termgraph. [software]. 2019. [2019-11-19]. Dostupné z: https://github.com/mkaz/termgraph

[30] NetworkX developers. Networkx. [software]. Říjen 2019 [2019-10-01]. Dostupné z: https://github.com/networkx/networkx

[31] The Matplotlib development team. Matplotlip. [software]. 2020 [2020-04-21]. Dostupné z: https://github.com/matplotlib/matplotlib

[32] Philipp Hagemeister. Ipaddress. [software]. Říjen 2019 [2019-10-18]. Dostupné z: https://github.com/phihag/ipaddress

[33] Python Software Foundation. Sqlite3. [software]. Dostupné z: https://docs.python.org/2/library/sqlite3.html

Příloha A

Seznam použitých zkratek

ACAP Application ConfigurationAccess Protocol

ADiCT Asset Discovery Classification Tagging

AP Access Point

CDP Cisco Discovery Protocol

CESNET Czech Education and Scientific NETwork

Cisco ISE Cisco Identity Services Engine

CPU Central Processing Unit

ČVUT České Vysoké Učení Technické v Praze

DDR3 double-data-rate 3

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

DSL Digital Subscriber Line Transceiver

FTP File Transfer Protokol

GHz gigahertz

HP Hewlett-Packard

HSRP HotStandby Router Protocol

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol over TLS/SSL

ICMP Internet Control Message Protocol

IFC TRAP Communication Interface

IMAP Internet Message AccessProtocol

IoT Internet of Things

IP Internet Protocol

IPP Internet Printing Protocol

IPv4 Internet Protovol version 4

IPv6 Internet Protocol version 6

IRC Inter-net Relay Chat

JSON JavaScript Object Notation

LAN Local Area Network

LDAP Lightweight Directory Access Protocol

LPD Line Printer Deamon

MAC Media Access Control

NEMEA Network Measurement Analysis

NAT Network Address Translation

NFS Network File System

NMAP Network Mapper

NPP Network Printing Protocol

NTP Network Time Protocol

PDP Packet Data Protocol

POP Post OfficeProtocol

PRTG Paessler Router Traffic Grapher

RAM Random Access Memory

RIP RoutingInformation Protocol

RRDTool Round-Robin Database Tool

RSTP Real Time Streaming Protocol

RTMP Real-Time Messaging Protocol

RTP Real-time Transport Protocol

SIP Session Initi-ation Protocol)

SMB Server Message Block

SMTP Simple Mail Transfer Protocol

SNMP SimpleNetwork Management Protocol

SQL Structured Query Language

SSD Solid-state drive

TAP Terminal AccessPoint

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

TFTP Trivial File Transfer Protocol

TRAP Traffic Analysis Platform

VoIP Voice over Internet Protocol

VRRP Virtual Router Redundancy Protocol

VUT Vysoké Učení Technické v Brně

WMI Windows Management Instru-mentation Remote Protocol

Příloha B

Instalační a uživatelská příručka

B.1 Instalace modulárního systému NEMEA na operačním systému CentOS 8

- Instalace GIT:

```
yum install git
```

- Naklonování NEMEA adresáře z GitHub:

```
git clone --recursive https://github.com/CESNET/nemea
```

- Instalace závislostí pro systém NEMEA:

```
yum install -y bc autoconf automake gcc gcc-c++ libtool libxml2-
    devel make pkg-config libpcap-devel libidn-devel bison flex
```

Na CentOS 8 se přes yum nenainstaluje libpcap-devel a libidn-devel, proto musíme ručně stáhnout a nainstalovat z:

https://centos.pkgs.org/8/centos-powertools-x86_64/libpcap-devel-1.9.0-3.el8.x86_64.rpm.html

https://centos.pkgs.org/8/centos-powertools-x86_64/libidn-devel-1.34-5.el8.x86_64.rpm.html

- Instalace python3:

```
yum install python3
```

- Samotná instalace NEMEA systému:

```
cd nemea/
./bootstrap.sh
./configure --enable-repobuild --prefix=/usr --bindir=/usr/bin/
    nemea --sysconfdir=/etc/nemea --libdir=/usr/lib64
make
make install
```

- Instalace NEMEA-Framework

```
cd  nemea/nemea-framework
./bootstrap.sh
./configure
make
make install
yum install python3-devel
cd pytrap
mkdir -p /usr/local/lib64/python3.6/site-packages/
python3 setup.py install
cd pycommon
mkdir -p /usr/local/lib/python3.6/site-packages/
python3 setup.py install
```

B.2 Instalace PassiveAutodiscovery modulu

- Stažení souborů modulu:

```
git clone https://github.com/koumajos/PassiveAutodiscovery
```

- Doinstalování použitých python knihoven:

```
pip3 install -r requirements.txt
```

B.3 Použití

Pro každou část implementace modulu jsou možnosti modulu ve zkratce popsané v nápovědě:

```
./CreateScript.py -h

./PassiveAutodiscovery.py -h

./CreateScript.py -h
```

B.3.1 PassiveAutodiscovery

```
  -h, --help            show this help message and exit
  -i IFC_SPEC           specification of interface types and their
      parameters, see "-h trap" (mandatory parameter).
  -v                    be verbose.
  -vv                   be more verbose.
  -vvv                  be even more verbose.

  -d NAME, --database NAME
                        set name of the database without . part, default is
                            "Database"
  -N IPs [IPs ...], --networks IPs [IP/M, IP/M, ...] IP addresses and mask (
      IPaddress/MASK - 192.168.1.0/24) of networks to monitor
  -!, --OnlySetNetworks only monitor entered networks via parameter N (
      ussage: -N ... -! )
  -U, --UsualyPorts     map only "usualy" transport layer ports
  -F, --filterIPFlows   filter incompletely connection
  -G, --GlobalDependencies
                        mapping the dependencies to global subnets
  -D NUMBER, --DeleteGlobal NUMBER
                        delete periodicly dependencies that have setted
                            amount of packets from global dependencies
  -RAM, --RAM           safe database in RAM memory and safe to file after
      modul end
  -T, --time            safe to database also time records of local and
      global dependencies
  -l, --localdev        print if modul find new local device (print will
      slow program)
  -s, --localserv       print if modul find new local services (print will
      slow program)
  -L, --localdependencies
                        print if modul find new dependencies between two "
                            local" device(print will slow program)
  -m, --macdev          print if found MAC adress for "local" device (print
      will slow program)
  -S, --globalserv      print if modul find new global service (print will
      slow program)
  -g, --globaldependencies
                        print if modul find new dependency between "local"
                            device and global device (print will slow
                            program)
```

B.3.2 DeviceAnalyzer

```
-h, --help            show this help message and exit
-D DEVICE, --device DEVICE
                      analyze single device [DEVICE = IP address of device
                           to analyze]
-d NAME, --database NAME
                      set name of the database without . part, default is
                          Database
-G NUMBER, --GlobalNumber NUMBER
                      number of global dependencies to print, default: all
                           dependencies
-L NUMBER, --LocalNumber NUMBER
                      number of local dependencies to print, default: all
                          dependencies
-J NAME, --json NAME  print to JSON file [NAME = name of the file without
    . part (file will be automatic set to .json), default =
    PassiveAutodiscovery ]
-f NAME, --file NAME  print to file [NAME = name of the file without .
    part (file will be automatic set to .txt) ]
-p, --print           print to command line
-P NAME, --printJSON NAME
                      print from json file that was created by
                          DeviceAnalyzer script. Need define where print
                          output (command line/file)  with parameter [-p],
                           [-f].
-DNS, --DNS           transalte [WEB Servers] IP to domain name and show
    in output
-t NUMBER, --timeL NUMBER
                      generate graphs of using dependencies in time for
                          setted number of local dependencies from mostly
                          used. (for working must be run
                          PassiveAutodiscovery.py wiht parameter -T)
-T NUMBER, --timeG NUMBER
                      generate graphs of using dependencies in time for
                          setted number of dependencies of local device
                          with global devices from mostly used. (for
                          workign must be run PassiveAutodiscovery.py wiht
                           parameter -T)
-A, --activity        print graph of activity device in network over time.
-l, --localgraph      create graph of dependencies between local devices
    and safe it to file
-g, --globalgraph     create graph of dependencies between local device
    and all global devices which was visited by local device, then safe it
     to file
-b, --bipartite       create graph of dependencies between local devices
    and global devices that was visited by more local device, then safe it
     to file
-o, --onlylocal       Analyze only local dependencies
```

B.3.3 CreateScript

```
-h, --help                show this help message and exit
-d NAME, --database NAME  Set name of the database without . part,
    default is Database
```

Příloha C

Příklady použití modulu

C.1 Přímé zapojení modulu do sítě

Nejdříve musíme nastavit na síťovém zařízení port-mirroring nebo použít jinou metodu duplikování síťové komunikace. Jakmile k našemu zařízení směřuje duplikovaná komunikace, spustíme *flow meter* NEMEA modul, které nám z paketů přicházející na síťovou kartu vytvoří flow záznamy a dále je bude distribuovat našemu modulu pomocí IFC rozhraní.

```
# /usr/bin/nemea/flow_meter -I NetworkInterface -i u:flows
```

kde *NetworkInterface* je označení síťové karty a *u:flows* je výstupní IFC rozhraní.

Následně vytvoříme databázi, pomocí námi vytvořeného skriptu *CreateScript*, následovně:

```
# ./CreateScript.py -d Database
```

kde *Database* je název vytvořené databáze.

Nakonec spustíme modul, které IP flows použije k naplnění databáze, tímto příkazem:

```
# ./PassiveAutodiscovery.py -i u:flows -d Database -G
```

kde *Database* je název vytvořené databáze, *u:flows* je IFC rozhraní z *flow meter* a parametr *-G* povoluje ukládání závislostí lokálního zařízení se zařízením v globální síti. Můžeme měření zpřesnit či cílit například pomocí parametrů *-N* pro zadání IP adres sledovaných sítí, *-!* pro nastavení vyhledávání zařízení z pouze zadaných sítí a *-U* pro ukládání závislostí s pouze „běžnými“ porty.

Po skončení měření či jeho přerušení můžeme uložená data v databázi podrobit analýze. Pro analýzu jsme vytvořili skript *DeviceAnalyzer* a použijeme ho tímto způsobem:

```
# ./DeviceAnalyzer.py -d Database -p -f output.txt -j output.json -l -g
```

kde *Database* je databáze určená k analýze, přepínač *-p* je výstup do příkazové řádky, *output.txt* je stejný výstup do souboru, *output.json* je výstup ve tvaru JSON určený pro budoucí snadný přístup k datům, přepínač *-l* pro vy-

tvoření grafu lokálních zařízení a přepínač *-g* pro vytvoření grafu závislostí s globálními zařízeními pro každé lokální zařízení.

Lze nastavit skript *DeviceAnalyzer* tak, aby se soustředil na jedno konkrétní zařízení s cílem vypsat informace pouze o něm, pomocí:

```
# ./DeviceAnalyzer.py -d Database -p -D 192.168.1.1 -G 15 -DNS
```

kde parametr *-D* určuje IP adresu analyzovaného zařízení, parametr *-G* určuje počet globálních závislostí, která se vypíší (bez nastavení se vypíší všechny globální závislosti) a parametr *-DNS* pro přeložení všech IP adres zařízení označených štítkem *WEB Server* na doménová jména, která vypíše do výstupu.

C.2 Spuštění modulu ze záznamu sítě

Nejdříve si ukážeme, jak v NEMEA systému vytvoříme záznam sítě. Ten ve formátu *.csv* vytvoříme pomocí dvou NEMEA modulů přes příkazy:

```
# /usr/bin/nemea/flow_meter -I NetworkInterface -i u:flows
```

```
# /usr/bin/nemea/logger -i u:flows -w file.csv -t
```

kde *NetworkInterface* je označení síťové karty, *flow* výstupní IFC rozhraní, *file.csv* výstupní soubor a parametr *-t* pro přidání na první řádek *.csv* souboru hlavičku jednotlivých sloupců.

Ze získaného *.csv* souboru potřebujeme získat soubor s příponou *.trapcap*, který lze zadat našemu modulu jako vstupní soubor. To uděláme pomocí dalšího NEMEA modulu následovně:

```
# /usr/bin/nemea/logreplay -i f:output.trapcap -f file.csv
```

Jakmile máme soubor s příponou *.trapcap* můžeme kdykoliv spustit náš modul pomocí:

```
# ./PassiveAutodiscovery.py -i f:file.trapcap -d Database -G
```

a dále pokračovat jako v předchozí ukázce.

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `docs` ..... vygenerovaná dokumentace přiložených python skriptů pomocí sphnix
    - `Collector.py`
    - `CreateScript.py`
    - `DeviceAnalyzer.py`
    - `PassiveAutodiscovery.py`
  - `thesis`
    - `thesis.tex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `Ukázky testování` ..... ukázky testování
    - `README.txt` ..... soubor obsahující popis testování a výstupních souborů přiložených v této složce
  - `thesis.pdf` ..... text práce ve formátu PDF