VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Michal Procházka

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE NA PLATFORMĚ ANDROID URČENÁ K IDENTIFIKACI ZAŘÍZENÍ IOT
APPLICATION ON THE ANDROID PLATFORM IMPLEMENTED TO IDENTIFY IOT DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Michal Procházka**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Zdeněk Martinásek, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Michal Procházka ***ID:*** 203712
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Aplikace na platformě Android určená k identifikaci zařízení IoT

POKYNY PRO VYPRACOVÁNÍ:

V rámci teoretické části práce prostudujte způsoby detekce a identifikace zařízení IoT v počítačových sítích. Vypracujte přehled současného stavu problematiky a dosažené výsledky přehledně srovnejte. Na základě vypracované rešerše navrhněte vlastní aplikaci, která aktivně detekuje a identifikuje zařízení IoT v rámci lokální počítačové sítě na platformě Andriod. Navržené řešení implementujte pro platformu Android zařízení. Realizujte experimentální vývojové pracoviště, na kterém implementaci otestujte. Hlavním výsledkem práce je plně funkční implementace detekce a identifikaci IoT zařízení dostupných v rámci lokální počítačové sítě.

DOPORUČENÁ LITERATURA:

[1] GUO, Hang; HEIDEMANN, John. Detecting iot devices in the internet. IEEE/ACM Transactions on Networking, 2020, 28.5: 2323-2336.

[2] MEIDAN, Yair, et al. A novel approach for detecting vulnerable IoT devices connected behind a home NAT. Computers & Security, 2020, 97: 101968.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.
Konzultant: Dominik Malčík (REDAMP SECURITY s.r.o.)

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce se zabývá problémem detekování a identifikování zařízení na lokální síti. Práce se zabývá zpracováním přehledu dostupných nástrojů pro detekování. Nalezené nástroje jsou mezi sebou porovnány. Práce se věnuje implementaci a shrnutí několika přístupu jak detekovat zařízeni na síti. Závěr se věnuje porovnání vlastní aplikace s nejvhodnějším konkurentem pro komerční řešení.

KLÍČOVÁ SLOVA

Skenování, síť, zařízení, Android, aplikace

ABSTRACT

The bachelor thesis focuses on a problem of detecting and identifying devices on a local network. An overview of available tools for detecting devices on the local network is presented in this thesis. The tools that are found are compared with each other. The implementation part provides a summary of possible approaches to detection of devices on the local network. The conclusion is devoted to the comparison of own application and the most suitable candidate for commercial use.

KEYWORDS

Scanner, network, device, Android, application

PROCHÁZKA, Michal. *Aplikace na platformě Android určená k identifikaci zařízení IoT*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 71 s. Bakalářská práce. Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Michal Procházka
VUT ID autora:	203712
Typ práce:	Bakalářská práce
Akademický rok:	2020/21
Téma závěrečné práce:	Aplikace na platformě Android určená k identifikaci zařízení IoT

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Zdeňku Martináskovi, Ph.D. a odbornému konzultantovi Ing. Dominikovi Malčíkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Úvod **12**

1 Analýza dostupných nástrojů **13**
1.1 Analýza konzolových nástrojů 13
1.2 Analýza nástrojů s grafickým rozhraním 18
1.3 Analýza nástrojů dostupných na Google play pro Android 23
1.4 Analýza nástrojů s otevřeným zdrojovým kódem pro Android 28
1.5 Analýza licencí 30
1.6 Shrnutí 32

2 Praktická část **35**
2.1 Realizace experimentálního pracoviště 35
2.2 Ověření funkčnosti pracoviště 38
2.3 Vlastní návrh aplikace pro skenování 40
2.4 Vlastní implementace aplikace 41
2.5 Skenování zařízení na síti 43
2.5.1 Nalezení zařízení pomocí ping příkazu 43
2.5.2 Čtení ARP mezipaměti 46
2.5.3 Používání NMAP nebo ARP příkazu 47
2.5.4 Použití network service discovery 48
2.5.5 Použití ip příkazu 48
2.6 Vlastní implementace 49
2.6.1 Detekce IoT zařízení 53
2.6.2 Návrhy na rozšíření aplikace 54
2.7 Porovnání aplikace Ning s vlastní aplikací 54

Závěr **59**

Literatura **60**

Seznam symbolů a zkratek **64**

Seznam příloh **66**

A Přiložené soubory **67**

B Zdrojové kódy grafického rozhraní **68**

C Zprovoznění aplikace **71**
C.1 Instalace aplikace pomocí APK 71
C.2 Zprovoznění aplikace v android studiu 71
C.3 Ukázka aplikace bez instalace 71

Seznam obrázků

1.1 OpenVAS sken . . . 17
1.2 Angry IP scanner . . . 19
1.3 Dipiscan . . . 20
1.4 Nessus . . . 21
1.5 PortScanner . . . 22
1.6 Rumble . . . 23
1.7 Network scanner od Easy mobile . . . 24
1.8 Network scanner od First row . . . 24
1.9 Fing sken . . . 25
1.10 Fing podrobnosti . . . 25
1.11 Whos on my wifi . . . 26
1.12 Wifi monitor . . . 26
1.13 Wifiman . . . 27
1.14 Wifiman2 . . . 27
1.15 Ning . . . 29
1.16 Android network tools . . . 29
1.17 Graf licencí . . . 30
2.1 Android studio . . . 36
2.2 Schéma připojených zařízení . . . 36
2.3 Zapnutá testovací aplikace . . . 38
2.4 Zapnutá testovací aplikace po několika kliknutí . . . 38
2.5 Grafické rozhraní . . . 42
2.6 Ping Scan . . . 45
2.7 Hledání MAC adresy prvním nástrojem . . . 53
2.8 Hledání MAC adresy druhým nástrojem . . . 54
2.9 Ning při zapnuté funkci oddělení uživatelů . . . 55
2.10 Výsledek skenování vlastní aplikace . . . 58
2.11 Výsledek skenování pomocí Ning . . . 58

Seznam tabulek

1.1 Licence 32
1.2 Podpora operačních systémů konzolových skenerů 33
1.3 Podporované protokoly u konzolových skenerů 33
1.4 Podpora operačních systémů skenerů s grafickým rozhraním 34
1.5 Podporované protokoly u skenerů s grafickým rozhraním 34
2.1 Porovnání aplikací dostupných na Google play 40
2.2 Porovnání vlastní apliakce a Ning 57

Seznam výpisů

1.1 Výsledek skenování Netcatem. 15
1.2 Výsledek skenování sítě pomocí nmapu. 16
1.3 Výsledek skenování Sn0int skenerem. 17
2.1 Zdrojový kód testovací aplikace (MainActivity.kt). 37
2.2 Funkce pro zjištění IP zařízení. 43
2.3 Funkce pro zjištění dostupnosti zařízení pomocí ping příkazu. 44
2.4 Příkaz nmap pro skenování sítě. 47
2.5 Příkaz ARP pro skenování sítě. 47
2.6 Příkaz ip neighbour pro skenování sítě. 48
2.7 Spouštění aplikace. 49
2.8 Načtení komponentů do proměnných. 50
2.9 Zdrojový kód stisknutí tlačítka 50
2.10 Zdrojový kód ke spuštění skenování 51
2.11 Zdrojový kód získání času 51
2.12 Zdrojový kód souboru recycler data item 51
2.13 Zdrojový kód návratových proměnných. 51
2.14 Zdrojový kód zpracování výsledek skenu. 52
2.15 Zdrojový kód spuštění skenování. 52
B.1 Zdrojový kód grafického rozhraní testovací aplikace (activity_main.xml). 68
B.2 Zdrojový kód grafického rozhraní aplikace (activity_main.xml). . . . 68
B.3 Zdrojový kód grafického rozhraní aplikace pro recyclerView. 69

Úvod

Tato práce se věnuje oblasti sítí a to konkrétně možnostem detekování zařízení internetu věcí (IoT) na síti pomocí mobilního telefonu s operačním systémem android. Praktickým cílem práce je porovnání dostupných řešení pro různé operační systémy, konkrétně pro windows, linux a android, a nalezení volně dostupných aplikací nebo knihoven, které by mohly být použity jako modul komerčního řešení aplikace. V případě nenalezení vhodné dostupné knihovny je třeba vytvořit vlastní jednoduchou aplikaci, která dokáže zjistit zařízení připojená k lokální síti.

V dnešní době se k sítím připojuje stále více zařízení a vzniká tak potřeba sítě monitorovat. Zařízení internetu věcí umožňují ovládání spotřebičů na dálku, odečítání hodnot přístrojů, které mohou být použité např. v lékařství. Narušitel se může k síti připojit i s normálním zařízením jako je chytrý telefon. Tato zařízení je také potřeba detekovat. Přesné číslo počtu zapojených zařízení je těžko odhadnutelné. Průzkumy odhadují mezi 7 až 22 miliardami připojených zařízení v roce 2016. Každý výzkum trochu jinak definoval zařízení, která patří do IoT. Nicméně, počet instalovaných zařízení do sítě se každoročně zvyšuje. Odhad pro rok 2021 hovoří o instalaci 31 miliard nových zařízení. Počet instalovaných IoT zařízení se meziročně zvýšil o 200 procent od roku 2016 do 2021 [1] [2]. Vzhledem k tomuto trendu vyvstala potřeba vypracovat přehled nástrojů umožňujících zmapování sítě a identifikaci nalezených zařízení. Prvním cílem této práce má tedy být zmapování dostupných nástrojů na různých operačních systémech a jejich porovnání. Dále je dán za cíl vytvoření aplikace pro nalezení zařízení, která se nacházejí na lokální síti.

Část vytvořené aplikace bude sloužit jako modul pro aplikaci externí firmy. Firma a externí konzultant mi dali za cíl zmapovat dostupný trh nástrojů pro skenování zařízení na síti pro všechny operační systémy a následně vyhodnotit, který z nalezených nástrojů by mohl splňovat podmínky použití jako modul komerční aplikace. V případě nenalezení vhodné aplikace nebo knihovny bude výsledkem práce vytvoření vlastní aplikace, která zařízení na síti dokáže detekovat.

Teoretická část se věnuje zmapování trhu dostupných nástrojů pro různé operační systémy. Nejprve se věnuji nástrojům, které mohou být ovládány pomocí konzole. Dále následuje přehled nástrojů s grafickým uživatelským rozhraním. Nakonec byl zpracován i přehled aplikací pro zařízení s operačním systémem android. Po přehledu zhodnocených nástrojů následuje popis a vyhodnocení použitých licencí, pod kterými jsou dané nástroje dostupné. Praktická část je rozdělena na tři části. První část porovnává aplikace dostupné přes obchod Google play. Druhá část se zaměřuje na návrh a implementaci vlastního řešení. V implementaci řešení je zahrnuto několik možných přístupů k detekci zařízení a jejich výhody, nevýhody nebo komplikace. Poslední část se soustřeďuje na porovnání aplikace Ning s vlastní aplikací.

1 Analýza dostupných nástrojů

S rozvojem sítí vyvstává potřeba monitorovat síť a zjišťovat zařízení k ní připojená. Síťové skenery slouží k detekování zařízení zejména v lokální síti. Základní využití skenerů je zjištění aktivních zařízení v daném rozsahu internet protokolu (IP) adres a navrácení seznamu zařízení. Pokročilejší skenery dokáží navrátit více hodnot než jen IP adresy, např. Media Access Control (MAC) adresy, seznam otevřených portů na zařízení atp. Díky zjištěným informacím mohou hackeři, penetrační testeři nebo systémoví administrátoři odhalovat známé zranitelnosti systému a následně je zneužívat, resp. opravovat. Síťový skener lze použít i v domácnosti ke zjišťování připojených zařízení a případně odpojení neznámých [3].

Nástroje lze rozdělit na dvě kategorie u operačních systémů Windows, Linux a Mac, a to konkrétně na konzolové nástroje a nástroje s grafickým uživatelským rozhraním. U operačního systému Android se vyskytují pouze nástroje s grafickým uživatelským rozhraním, ale uplatním zde rozdělení nástrojů dostupných na goole play, které v tomto případě neobsahovaly dostupné zdrojové kódy, a na nástroje s otevřeným zdrojovým kódem.

1.1 Analýza konzolových nástrojů

Tato kapitola se zabývá analýzou nástrojů, které jsou dostupné pouze pomocí konzole. Tyto nástroje jsou dostupné hlavně pro operační systémy určené pro stolní počítače. U nástrojů byla prozkoumána jejich historie, licence a zda jsem je byl schopen zprovoznit na zařízení s operačním systémem Linux s distribucí Fedora 33.

Dmitry

Nástroj vyvinul James Greig v roce 2016. Celý nástroj je volně dostupný pod licencí GNU General Public License v2.0. Slouží k získání informací o hostiteli pomocí nástrojů *whois* hledání a následně využívá skenování portů pomocí Transmission control protocol (TCP). U nástroje nedochází k dalšímu vývoji a dochází zde jen k nejnutnějším úpravám. Sám autor napsal, že pokud by se měl nástroj použít v dnešní době, tak by musel být přepsán do funkční podoby [4]. Nástroj se objevuje v součásti distribuce Kali Linux, které se používá k penetračnímu testování nebo k digitální forenzní analýze. Nástroj Dmitry jsem nebyl schopen zprovoznit na svém zařízení kvůli nekompatibilitě novějších verzí knihoven.

MASSCAN

MASSCAN začal vyvíjet Robert Graham v roce 2013. Nástroj je volně dostupný pod licencí GNU Affero General Public License version 3. Nástroj je aktivně vyvíjen na platformě github, kam mohou uživatelé volně přispívat připomínkami a návrhy. Oproti později zmíněnému Nmap skeneru se aplikace zaměřuje na skenování větších sítí. Nástroj je zaměřený především na velké sítě za účelem zjištění základní topologie sítě, ale díky množství dostupných přepínačů jej lze efektivně využít i v rámci menších sítí. Původní myšlenka autora byla použít skener na celý internet, kdy autor uvedl, že MASSCAN dokáže zmapovat celý internet do 5 minut, kdy vysílá 10 miliónů paketů za sekundu z jednoho zařízení.

Během detekování zařízení a otevřených portů se nástroj může pokusit o navázání spojení se zařízeními prostřednictvím následujících protokolů:

- file transfer protocol (FTP),
- hypertext transfer protocol (HTTP),
- Internet message access protocol (IMAP4),
- memcached,
- post office protocol (POP3),
- simple mail transfer protocol (SMTP),
- secure shell (SSH),
- secure sockets layer (SSL),
- server message block (SMBv1),
- server message block (SMBv2),
- teletype network (Telnet),
- remote desktop protocol (RDP),
- virtual network computing (VNC).

Nástroj využívá vlastní konfiguraci TCP/IP, díky které lze upravovat interní firewall, který aplikace využívá, aniž by bylo pozměněno nastavení sítě na daném zařízení [5]. Nástroj jsem nebyl schopen zprovoznit kvůli novějším verzím používaných knihoven.

Nikto2

První verze Nikto vyšla na konci roku 2001, kterou vydal Chris Sullo. Nástroj je volně dostupný pod licencí GNU General Public License v2.0. Nikto2 je dostupný na githubu a postupně dochází k přepsání celého skriptu na modernější kód. Používaný protokol ke zjišťování informací o zařízení je TCP. Skener se zaměřuje na odhalování slabin u webových serverů včetně potenciálně nebezpečných souborů nebo špatně nastavených konfigurací serveru. Nástroj se může použít, pokud některé ze zařízení

nabízí přístup do nastavování pomocí webu [6]. Nevýhoda nástroje je nutnost zjištění informace, že dané zařízení je připojené na síť. Nástroj jsem nebyl schopen zprovoznit na zařízení.

Netcat

První verze Netcatu vyšla 28. října 1995 a byla vydána autorem s přezdívkou Hobbit. Nástroj je dostupný pod licencí European Union Public Licence. Samotný Netcat už není vyvíjen. Poslední vydaná verze je 1.10 z roku 2007. Z projektu vychází několik následovatelů. Největší využití našel jako rozšiřující balíček aktualizovaného Nmapu. Netcat využívá k navázání a zjišťování informací o zařízení TCP a UDP pakety [7]. Na výpise z konzole 1.1 lze vidět jeden z možných výsledků skenování na síti. Uvedené zjištěné informace se týkají otevřených portů na lokálním zařízení. Bylo zjištěné, že je otevřený pouze port 80, který slouží k HTTP komunikaci. Nástroj v dnešní době slouží jako doplněk pro netcat a je podporován. Nástroj jsem byl schopen zprovoznit a udělat tím sken sítě.

```
[mprochazka@forbiddenlaptop ~]$ nc -zv 192.168.68.107 80
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.68.107:80.
Ncat: 0 bytes sent, 0 bytes received in 0.02 seconds.
[mprochazka@forbiddenlaptop ~]$ nc -zv 192.168.68.107 443
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Connection refused.
[mprochazka@forbiddenlaptop ~]$ nc -zv 192.168.68.107 22
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Connection refused.
```

Výpis 1.1: Výsledek skenování Netcatem.

Nmap

Nmap je nástroj, který byl poprvé zveřejněný v časopise Phrack Magazine v roce 1997. Vývojářem je Gordon Lyon, který na něm neustále pracuje a poslední vydanou verzí je 7.91 v roce 2020. Nástroj je volně dostupný skener pod licencí GNU General Public License v2.0. Slouží k hledání zařízení v síti, zjišťování otevřených portů, operačních systémů nebo aplikací spuštěných na zařízení, které komunikují s internetem. Nmap podporuje skriptování díky modulu NSE (Nmap Scripting engine) pro lepší testování sítí. Díky skriptům můžeme automatizovat operace a napojovat je na další aplikace, které můžou prozrazovat více o slabinách jednotlivých zařízení [8]. Ve výpisu 1.2, lze vidět výsledek skenování sítě. Bylo nalezeno 8 zařízení za necelé tři sekundy skenování. Přepínače *-sP*, slouží k zjištění zařízení na síti pomocí příkazu *ping*, ale

nespustí se zjišťování otevřených portů u nalezených zařízení. Nástroj patří k základním příkazům Linuxu a bylo možné jednoduše s pomocí návodů napsat příkaz pro skenování sítě.

```
[mprochazka@forbiddenlaptop ~]$ nmap -sP 192.168.68.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-24 10:34 CEST
Nmap scan report for _gateway (192.168.68.1)
Host is up (0.0032s latency).
Nmap scan report for 192.168.68.104
Host is up (0.018s latency).
Nmap scan report for 192.168.68.107
Host is up (0.010s latency).
Nmap scan report for 192.168.68.108
Host is up (0.012s latency).
Nmap scan report for 192.168.68.112
Host is up (0.0078s latency).
Nmap scan report for 192.168.68.114
Host is up (0.016s latency).
Nmap scan report for 192.168.68.118
Host is up (0.020s latency).
Nmap scan report for forbiddenlaptop (192.168.68.127)
Host is up (0.0016s latency).
Nmap done: 256 IP addresses (8 hosts up) scanned in 2.86 seconds
```

Výpis 1.2: Výsledek skenování sítě pomocí nmapu.

OpenVAS

První verze OpenVAS byla vyvinuta vývojáři projektu Nessus v roce 2005. Původní vývojáři se na projektu OpenVAS přestali podílet ve stejném roce a začali vyvíjet komerční verzi nazvanou Nessus, která neměla otevřený zdrojový kód. Následně mezi roky 2006-2008 se na vývoji vystřídalo několik vývojářů. V roce 2008 se do vývoje zapojil momentálně hlavní vývojář nástroje a to společnost Greenbone Networks. Firma produkt neustále vyvíjí a vytváří nové testy zranitelností. Skener je volně dostupný na githubu pod licencí GNU General Public License v2.0. OpenVAS je aktivně vyvíjen [9]. Tento nástroj jsem byl schopen zprovoznit na zařízení s operačním systémem Ubuntu 20.04. Nástroj může být ovládán pomocí konzole nebo grafického rozhraní. Skenování proběhlo snadno a výsledek byl přehledně zapsán do portable document format (PDF) souboru. Skenování proběhlo na virtuálním stroji, který není připojen k ostatním zařízením a na výsledku skenu lze vidět pouze vlastní zařízení, viz obrázek 1.1.

1 Result Overview

Host	High	Medium	Low	Log	False Positive
10.34.6.134	0	0	0	6	0
Total: 1	0	0	0	6	0

Vendor security updates are not trusted.
Overrides are off. Even when a result has an override, this report uses the actual threat of the result.
Information on overrides is included in the report.
Notes are included in the report.

Obr. 1.1: Výsledek skenu pomocí openVAS.

Sn0int

První verze vyšla 10. srpna 2018. Samotný autor není uveden, pravděpodobně se na vývoji podílí více autorů. Nástroj je volně dostupný pod licencí GNU General Public License v3.0. Sn0int je dostupný a aktivně vyvíjený na githubu. Nástroj je určený pro profesionály z oblasti kybernetické bezpečnosti, kteří hledají informace o daném cíli. Nástroj dokáže zjistit veřejné informace o webových stránkách, zmapovat lokální síť, získat informace o subdoménách z certifikátů nebo také najít nahotu v obrázcích [10]. Ve výpisu na obrázku 1.3 lze vidět nalezené subdomény hlavní domény *seznam.cz*. Nástroj disponuje značným množstvím dalších možností a dokáže instalovat různé moduly pro rozšíření jeho funkcionality. Nástroj jsem byl schopen nainstalovat, ale nenašel jsem konkrétní knihovnu pro skenování sítě. Výsledek skenování je získání informace o subdoménách domény *seznam.cz*.

```
[mprochazka@forbiddenlaptop sn0int]$ sn0int
[+] Connecting to database
[+] Loaded 60 modules
[sn0int][default] > workspace bpp
[+] Connecting to database
[sn0int][bpp] > add domain
[?] Domain: seznam.cz
[sn0int][bpp] > use ctlogs
[sn0int][bpp][kpcyrd/ctlogs] > run
[*] "seznam.cz"     : Adding subdomain "diskuze.seznam.cz"
[*] "seznam.cz"     : Adding subdomain "blog.seznam.cz"
[*] "seznam.cz"     : Adding subdomain "firma.seznam.cz"
[*] "seznam.cz"     : Adding subdomain "info.seznam.cz"
```

```
[*] "seznam.cz"      : Adding subdomain "internet.seznam.cz"
[*] "seznam.cz"      : Adding subdomain "kariera.seznam.cz"
[*] "seznam.cz"      : Adding subdomain "nenechamevasvtom.seznam.cz"
[*] "seznam.cz"      : Adding subdomain "onas.seznam.cz"
```

Výpis 1.3: Výsledek skenování Sn0int skenerem.

Scapy

Jednu z prvních verzí vydal Philippe Biondi 22. června 2007. Nástroj je volně dostupný pod licencí GNU General Public License v2.0. Scapy dokáže skenovat zařízení, vypsat cestu k cílovému zařízení, zjistit slabiny a vytvářet útoky na nalezené slabiny. Nástroj byl napsán v programovacím jazyce python. Hlavním cílem projektu bylo vytvořit nástroj pro zachytávání a upravování paketů [11]. Nástroj jsem byl schopen zprovoznit, ale nedokázal jsem vytvořit funkční skenování sítě.

ZMap

První verze vyšla 16. srpna 2013, byla vydaná asistentem profesora ze Stanfordské univerzity Zakirem Durumericem. Nástroj je dostupný pod licencí Apache License, Version 2.0. Ve vývoji pokračují zakladatel projektu a kolegové zakladatele na Standfordské univerzitě. ZMap dokáže při nasazení na jediném zařízení s rychlostí připojení 1 Gb/s zmapovat celý veřejný internet za cca 45 minut [12]. Nástroj jsem byl schopen zprovoznit pouze částečně. Při pokusu o skenování sítě ZMap zobrazoval chybu týkající se špatně nastavené výchozí brány. Tuto chybu se mi ani při opakovaných pokusech nepodařilo odstranit a nenašel jsem žádné návody, jak tuto chybu odstranit.

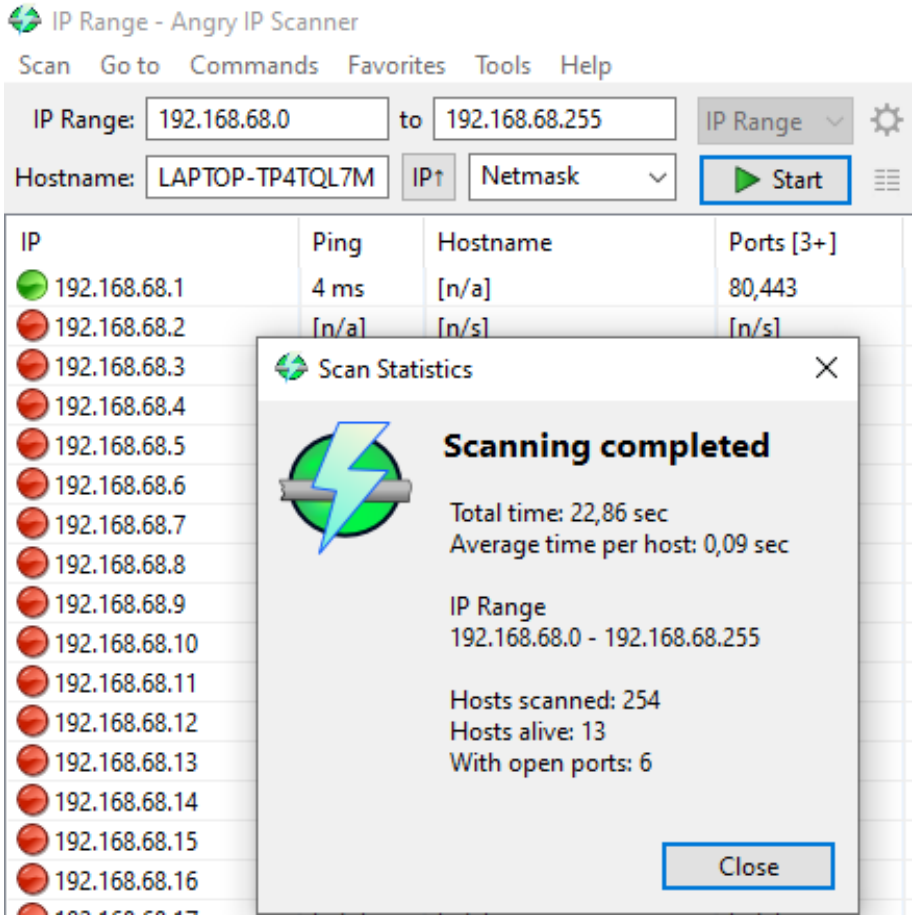
1.2 Analýza nástrojů s grafickým rozhraním

V této kapitole jsou zmíněny nástroje, které mají možnost ovládání pomocí grafického uživatelského rozhraní. U nástrojů jsou popsány podporované operační systémy, protokoly a jejich historie. Nástroje jsem se pokoušel zprovoznit na operačním systému Windows 10 nebo na operačním systému Linux (distribuce Fedora 33).

Angry IP scanner

První verze, kterou vydal Anton Keks, se objevila na githubu 20. dubna 2001. Nástroj je dostupný pod licencí GNU General Public License v2.0. Zdrojový kód je dostupný na githubu a je možné podávat návrhy nebo připomínky na jeho zlepšení. Na projektu se aktivně pracuje. Nové verze vychází každý měsíc. Program používá

k analýze komunikaci TCP, UDP a ICMP. Podporuje IPv4 i IPv6 adresy a multivláknové operace pro skenování sítě [13]. Na obrázku 1.1 lze vidět výsledek skenování sítě, kdy vidíme, že bylo nalezeno 13 zařízení za 23 sekund. Nástroj jsem byl schopen zprovoznit velice snadno na systému Windows.



Obr. 1.2: Výsledek skenování Angry IP skenerem.

Archery

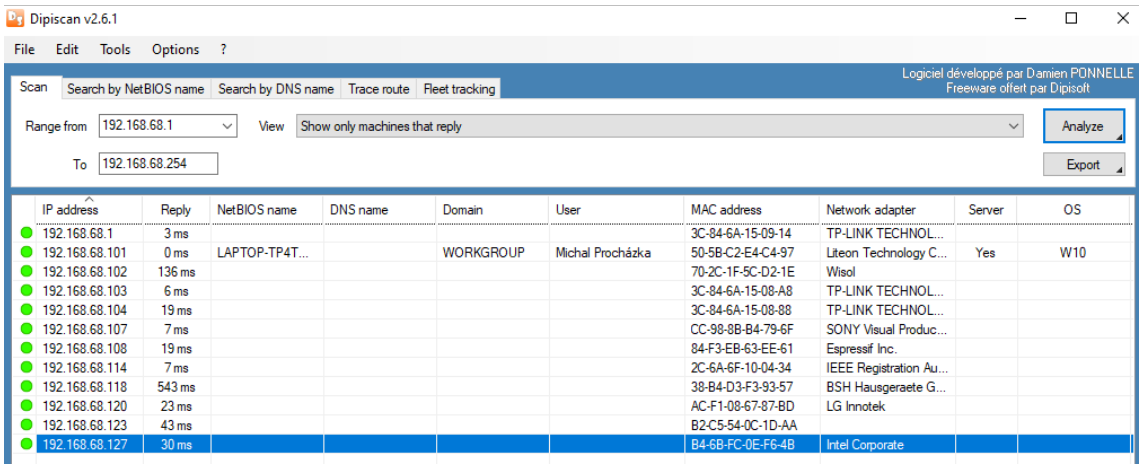
Archery je projekt vytvořený Anandem Tiwari v roce 2017. Nástroj je dostupný pod licencí GNU General Public License GPL v3.0. Zdrojový kód je dostupný na githubu a prostřednictvím diskusních příspěvků je možné podávat návrhy na nové funkce nebo na jejich opravy. Ke skenování sítě aktuálně podporuje použití tří nástrojů:

- Nmap,
- Nessus scanner,
- OpenVAS.

Podporuje nejen použití samotných nástrojů, ale i výstupů k analýze. Zde uvedené nástroje jsou popsány v této práci. K aktualizacím dochází poměrně často, standardně několikrát za čtvrtletí [14]. Nástroj jsem nebyl schopen zprovoznit kvůli nezprovoznění ostatních potřebných nástrojů.

Dipiscan

Vývoj nástroje Dipiscan začal v roce 2014 Damienem Ponnellem, který na něm i nadále pracuje sám. Damien Ponnelle původně vyvíjel IPScan32, ale vzhledem k nutnosti modernizace a upravení kódu byl vyvinut právě Dipiscan. Nástroj je dostupný pod licencí Freeware. Kód není dostupný, ale uživatel může nahlašovat chyby nebo návrhy přes webovou stránku nebo přes email. Vychází pravidelné aktualizace s opravami a vylepšeními [15]. Nástroj byl velice snadno instalovaný na systému Windows a jeho ovládání bylo intuitivní. Výsledek skenování lze vidět na obrázku 1.3.

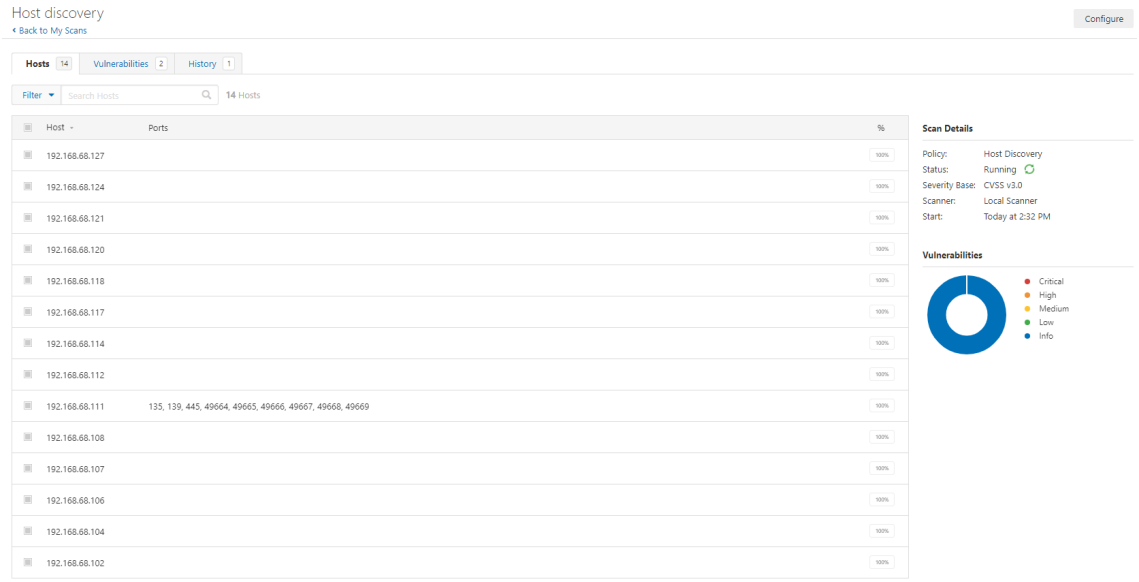


Obr. 1.3: Výsledek skenování Dipiscanem.

Nessus

Projekt vývoje Nesuss začal Renaud Deraison v roce 1998 společně s internetovou komunitou. V roce 2005 Renaud Deraison založil firmu Tenable, která začala Nessus nabízet jako placený produkt pod proprietární licencí. V dnešní době má Nessus dvě varianty, velmi omezenou verzi zdarma pro vyzkoušení a placenou verzi s plnou

podporou. Produkt má velkou komunitu uživatelů a aktivní podporu vývojářů, aktualizace vychází téměř každý den [16]. Nástroj bylo možné nainstalovat v testovací podobě a spustit test na platformě Windows na WiFi. Nástroj nabízí několik základních možností skenování sítě. U každého zařízení nachází otevřené porty. Výsledný sken nalezených zařízení lze vidět na obrázku 1.4.



Obr. 1.4: Výsledek skenování Nessus skenerem.

Netcrunch

Netcrunch začala vyvíjet společnost AdRem Software v roce 1998. Firma nabízí placený produkt pod licencí proprietární software s vyzkoušením na 30 dní zdarma. Nástroj využívá sadu protokolů SNMP k detekci zařízení v síti. Program je často aktualizován a nové verze vychází po několika měsících. Po zakoupení produktu firma nabízí podporu pro řešení problémů [17]. Nástroj jsem nebyl schopný nainstalovat, protože nepodporuje novější knihovny.

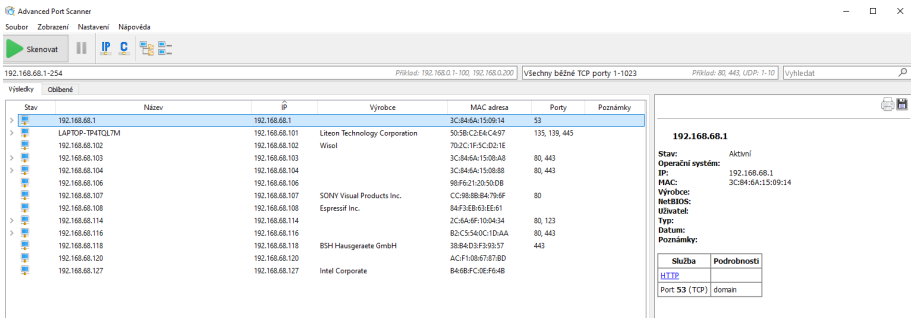
NmapSi4

NmapSi4 vyvinul Francesco Cecconi v roce 2009 a jedná se o grafickou nadstavbu pro klasický nmap. Nástroj je dostupný pod licencí GNU General Public License v2.0. Program umožňuje jednoduché ovládání nmap skeneru pomocí grafického rozhraní, ukládání a načítání výsledků ze skenů sítě. Dokáže si zapamatovat filtry pro hledání zařízení v síti. Program nemá velkou základnu uživatelů, kteří jej využívají.

Aktualizace nevychází moc často, poslední dvě velké aktualizace vyšly v roce 2015 a následně v roce 2020 [18]. Nástroj nebylo možné nainstalovat, jelikož není aktivně vyvíjen a nepodporuje distribuci novější distribuce Fedory.

PortScanner

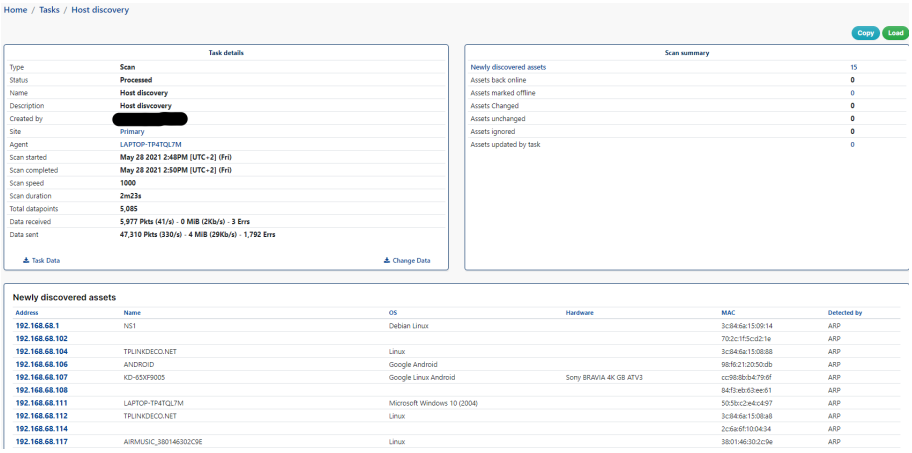
PortScanner vyvíjí firma SolarWinds a verze 1.0 vyšla v roce 2017. Firma nabízí placený produkt pod proprietární licencí s možností vyzkoušení softwaru na 30 dní zdarma nebo produkt s méně funkcemi zcela zdarma. Nástroj dokáže skenovat rozsah IP adres nebo samotné domény, dokáže odhalit otevřené porty pro TCP a UDP komunikaci. Pomocí příkazu ping může zjišťovat MAC adresu zařízení nebo operační systém zařízení. Program dokáže naplánovat skenování sítí a následně je ukládat do souborů pro následující porovnání [19]. Nástroj bylo velice snadné instalovat na Windows zařízení a vytvořit rychlý a jednoduchý skenování sítě. Výsledný sken sítě lze vidět na obrázku 1.5.



Obr. 1.5: Výsledek skenování PortScannerem.

Rumble network discovery

Rumble network discovery vyvíjí společnost Rumble, která začala s programováním skeneru v roce 2018. Nástroj je rozdělen na více částí a každá část má jinou licenci. Můžeme zde najít licence: Apache License v2.0, BSD 3-Clause "New"or "Revised"License, BSD-2-clause nebo MIT License. Zdrojový kód je dostupný na githubu a je možné nahlašovat chyby nebo podávat návrhy na zlepšení. Nástroj skenuje rozsah IP adres, u kterých zjišťuje informace a ty následně hledá v cloudové databázi. Z databáze zjišťuje informace o zařízeních, které vypisuje do konzole nebo grafického rozhraní. K aktualizacím dochází velmi často, v podstatě na týdenní bázi, kdy jsou opravovány chyby nebo vydána nová funkcionalita [20]. Nástroj nebylo možné vyzkoušet ve verzi pro osobní využití. Nástroj se obsluhuje pomocí webové stránky. Výsledný sken aplikace lze vidět na obrázku 1.6.



Obr. 1.6: Výsledek skenování Rumble.

Umit Network scanner

Umit network scanner je jedna z dalších grafických nadstaveb konzolového Nmapu. Vývoj začal Adriano Monteiro Marques během Google Summer of Code v roce 2005. Nástroj je dostupný pod licencí GNU General Public License. Cíl vývoje bylo poskytnout nástroj dostatečně flexibilní a zároveň přehledný pro začínající i zkušené uživatele Nmapu. Program si sice našel své uživatele, ale již nedochází k vývoji nových verzí. Poslední verze vyšla v roce 2013 [21]. Nástroj nebylo možné nainstalovat, protože poslední verze vyšla před osmi lety a nepodporuje novější knihovny.

1.3 Analýza nástrojů dostupných na Google play pro Android

Tato kapitola se bude zabývat nástroji, které jsou dostupné v obchodě Google play, který umožňuje distribuci aplikací pro operační systém Android. Nástroje byly testovány na telefonu Redmi Note 8. Nástroje zde neobsahují otevřený zdrojový kód.

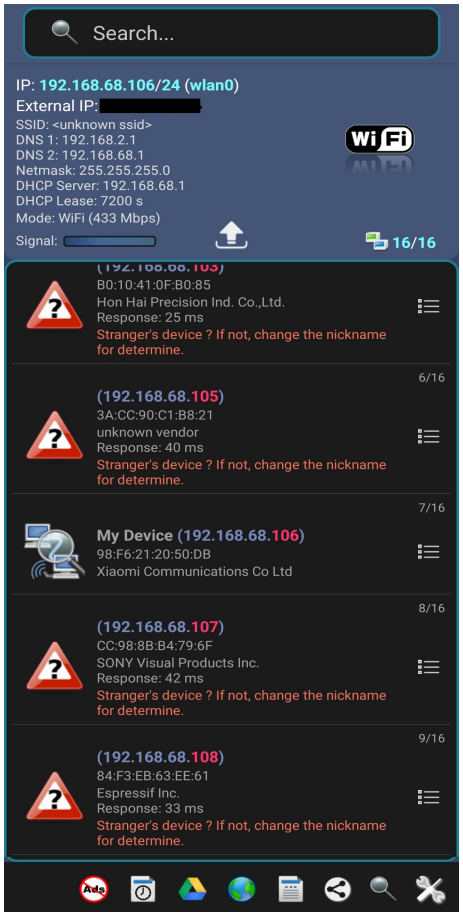
Network scanner od Easy Mobile

Network scanner od Easy mobile vznikl v roce 2014 a vyvíjí ho společnost Easy mobile. Zdrojový kód není dostupný a je vydán pod proprietární licencí. Nástroj dokáže zjistit zařízení připojené k síti, odezvu mezi zařízeními, otevřené porty, ukládání a načítání výsledků do a ze souboru. Aplikace je limitována pouze na skenování

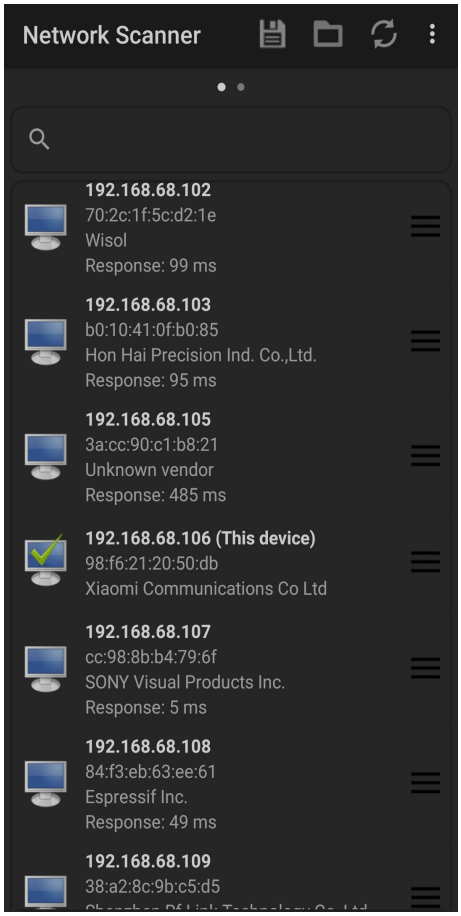
připojené WiFi sítě. V aplikaci nedochází k častým aktualizacím. Poslední aktualizace proběhla v srpnu 2020 [22]. Nástroj byl snadno ovladatelný a bylo možné zjistit základní informace o skenované sítí a nalezených zařízeních. Výsledný sken sítě lze vidět na obrázku 1.7.

Network scanner od First row

Network scanner od First Row vznikl v roce 2015 a vyvíjí ho společnost First row. Zdrojový kód není dostupný a je vydán pod proprietární licencí. Nástroj dokáže zjistit zařízení připojené k síti, odezvu mezi zařízeními, ukládání a načítání výsledků do a ze souboru. Aplikace dokáže skenovat připojenou WiFi síť i rozsah IP adres. K aktualizacím aplikace dochází pouze zřídka [23]. Aplikace nabízí snadné uživatelské rozhraní, které nabízí základní informace o skenovaných zařízení. Výsledný sken sítě lze vidět na obrázku 1.8.



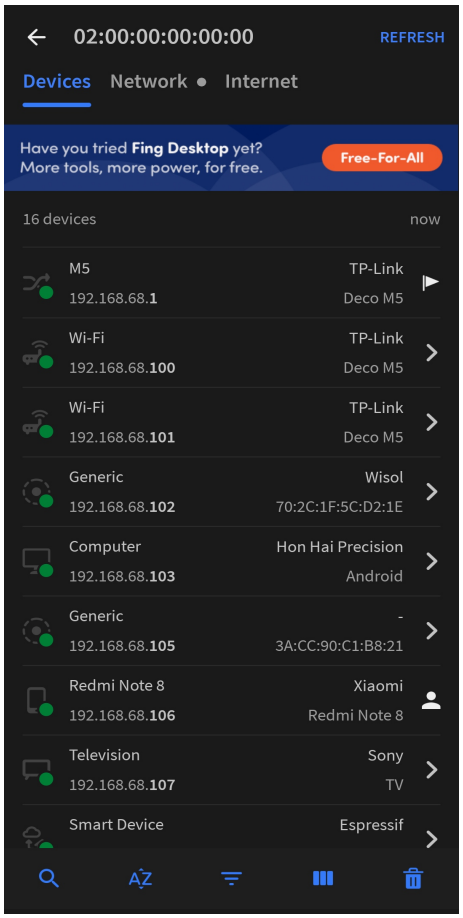
Obr. 1.7: Výsledek skenu pomocí network scanner od Easy mobile.



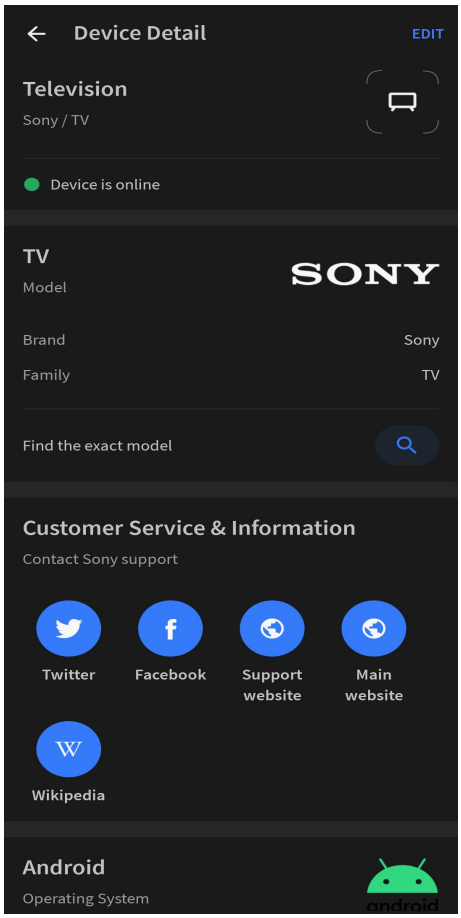
Obr. 1.8: Výsledek skenu pomocí network scanner od First row.

Fing

Fing pro mobilní zařízení vznikl v roce 2010 a vyvíjí ho společnost Fing Limited, který vydává verze pro počítače. Zdrojový kód celé aplikace není dostupný, ale některé části kódu lze najít k prozkoumání na githubu. Vydán je pod proprietární licencí. Dostupná rozšíření lze dokoupit v některém z nabízených balíčků. Nástroj dokáže zjistit zařízení připojené k síti, odezvu mezi zařízeními, otevřené porty, ukládání a načítání výsledků do a ze souboru a provádět testy rychlosti stahování a nahrávání dat. U nalezených zařízení zjistí výrobce, model zařízení a další dostupné informace ze své databáze. Aplikace dokáže skenovat pouze připojenou WiFi síť. V aplikaci dochází k velmi častým aktualizacím [24]. Aplikace nabízí více funkcí a bylo potřeba se zorientovat, jakou funkci aplikace chci využít. Oskenovaná síť je vyobrazena na obrázcích 1.9 a 1.10.



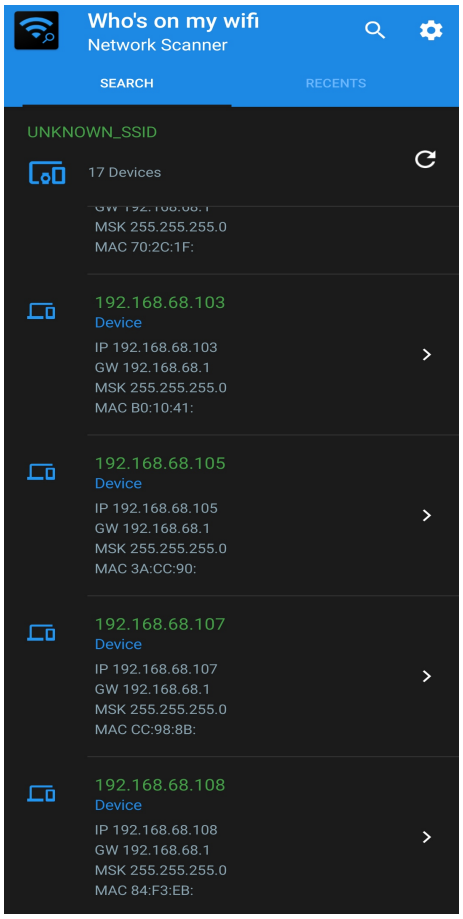
Obr. 1.9: Výsledek skenu pomocí Fing.



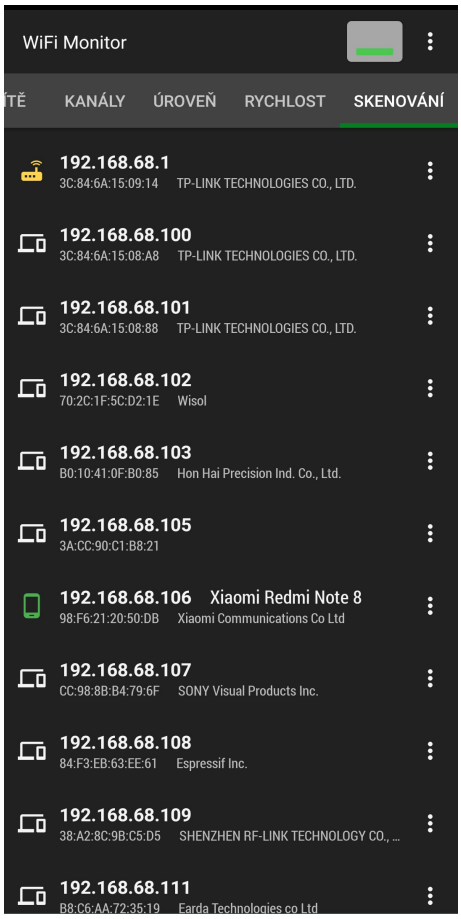
Obr. 1.10: Fing podrobnosti o nalezeném zařízení.

Who's on my wifi

Nástroj Who's on my wifi vznikl v roce 2017 a vyvíjí ho společnost Magdald. Zdrojový kód není dostupný a je vydán pod proprietární licencí. Je to jednoduchý nástroj, který slouží ke zjištění připojených zařízení k síti. Během skenování lze o dostupných zařízeních zjistit několik informací jako je MAC adresa a možnosti pojmenování zařízení. Aplikace dokáže skenovat pouze připojenou WiFi síť. V aplikaci dochází k občasným aktualizacím [25]. Aplikace má jednoduché grafické rozhraní, které umožní svým uživatelům snadné ovládání a zjišťování informací. Výsledný sken sítě je prezentován na obrázku 1.11.



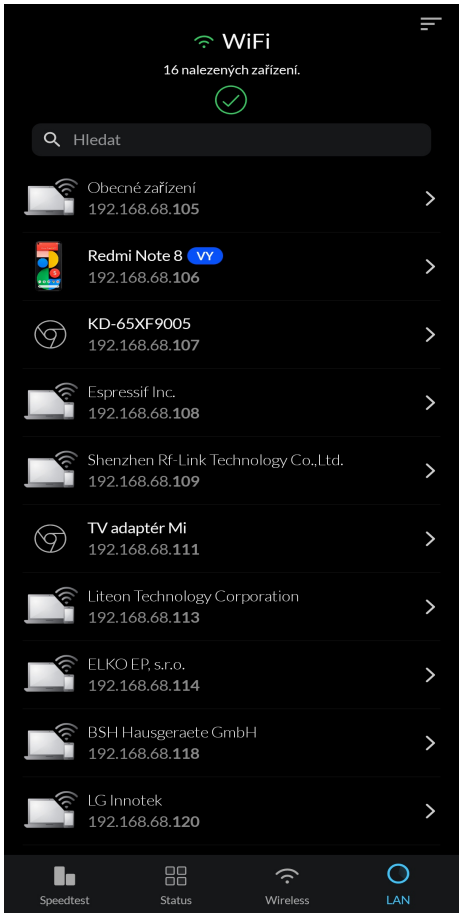
Obr. 1.11: Výsledek skenu pomocí Who's on my wifi.



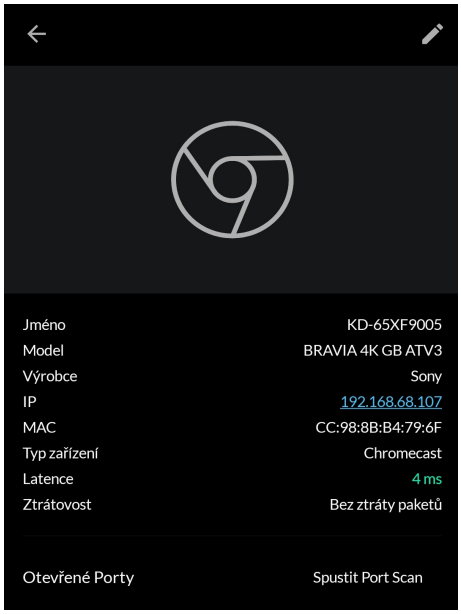
Obr. 1.12: Výsledek skenu pomocí Wifi monitor.

WiFiman

WiFiman vznikl v roce 2018 a vyvíjí ho společnost Ubiquiti Inc.. Zdrojový kód není dostupný a je vydán pod proprietární licencí. Nástroj slouží ke zjištění zařízení připojených k síti a několika informací o něm jako je MAC adresa, možnosti pojmenování zařízení, zjištění odezvy zařízení a otevřené porty. Následně dokáže vyhodnotit sílu okolních WiFi sítí. Aplikace dokáže skenovat pouze připojenou WiFi síť. V aplikaci dochází k častým aktualizacím [26]. Aplikace přehledně zpracovává nalezené zařízení a zobrazuje ikony zařízení nebo výrobce. Výsledný sken lze vidět na obrázku 1.13 a 1.14.



Obr. 1.13: Výsledek skenu pomocí WiFiman.



Obr. 1.14: WiFiman podrobnosti o nalezeném zařízení.

WiFi Monitor

WiFi monitor vznikl v roce 2015 a vyvíjí ho Alexander Kozyukov. Zdrojový kód není dostupný a je vydán pod proprietární licencí. Mezi zjišťované údaje patří MAC adresa, jméno výrobce zařízení a možnosti pojmenování zařízení. Následně je možné i vyhodnotit sílu okolních WiFi sítí. Aplikace dokáže skenovat pouze připojenou WiFi síť. V aplikaci dochází k častým aktualizacím [27]. Aplikace nabízí více funkcí a skenování lokální sítě se nachází na konci možných funkcí. Zobrazení nalezených zařízení je zobrazeno přehledně a lze to vidět na obrázku 1.12.

1.4 Analýza nástrojů s otevřeným zdrojovým kódem pro Android

V této kapitole jsou zmíněny nástroje, které jsou určené pro operační systém Android a mají otevřený zdrojový kód dostupný na githubu. Aplikace byly nainstalovány, pokud to bylo možné, a otestované.

Network Discovery

Aubort Jean-Baptista vyvinul aplikaci Network Discovery v roce 2009. Zdrojový kód je dostupný na githubu a je licencován pod GNU General Public License 2.0. Nástroj zjišťuje informace o zařízení připojených k lokální síti. Jsou nalezeny MAC adresy a jména výrobců zařízení. Aplikace dokáže skenovat pouze připojenou WiFi síť. Původní vývojář aplikaci přestal vyvíjet. Nástroj je napsaný v jazyce JAVA [30]. Nástroj nebylo možné zprovoznit, protože nepodporuje novější verze Androidu.

Network Tools Library

Projekt Network tools library vznikl v roce 2015 a vyvíjel ho Matthew Rollings. Zdrojový kód je dostupný na githubu a je licencován pod Apache License 2.0. Nástroj slouží ke zjištění zařízení připojených k lokální síti a několika informací o něm jako je MAC adresa. Aplikace dokáže skenovat pouze připojenou WiFi síť. Původní vývojář aplikaci přestal vyvíjet, pouze přijímá návrhy na opravení chyb nebo vylepšení. Aplikace je napsaná v jazyce JAVA [29]. Nástroj bylo možné nainstalovat a knihovna lze využít v rámci Android aplikací. Oskenovaná síť prostřednictvím této aplikace je zobrazena na obrázku 1.14.

Ning

Carsten Csiky začal vyvájet aplikaci Ning v roce 2019. Zdrojový kód je dostupný na githubu a je licencován pod GNU Lesser General Public License v3.0. Nástroj slouží, jako i ostatní nástroje, ke zjištění zařízení připojených k lokální síti a extrakci několika informací o zařízení. Mezi zjištěné údaje patří MAC adresa a jméno výrobce zařízení. Aplikace dokáže skenovat pouze připojenou WiFi síť. V aplikaci dochází k občasným aktualizacím a je napsaná v jazyce Kotlin [28]. Aplikaci bylo možné zprovoznit a otestovat. Výsledek skenu sítě můžete vidět na obrázku 1.13.

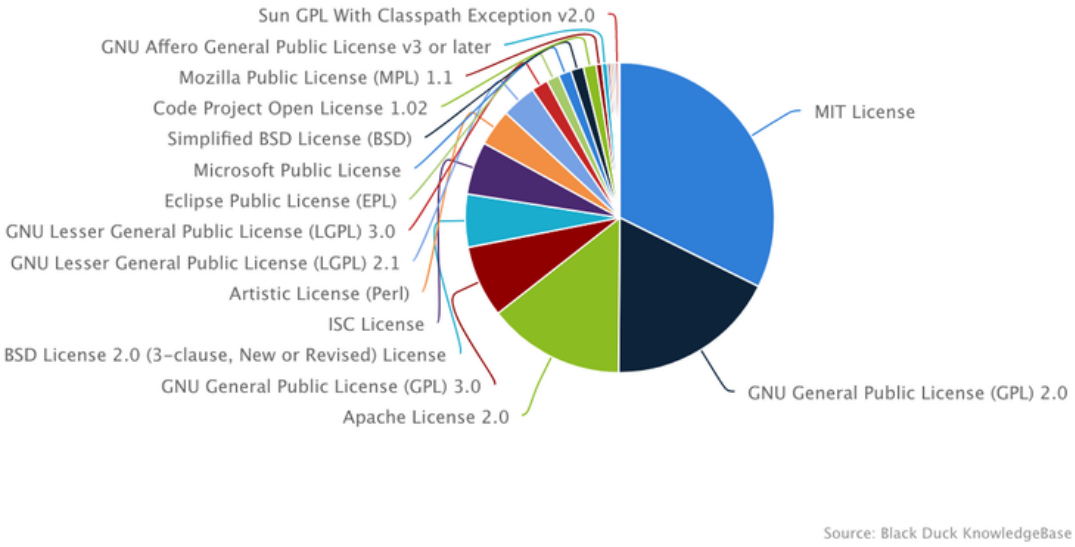
Obr. 1.15: Výsledek skenu pomocí Ning.

Obr. 1.16: Výsledek skenu pomocí Android network tools.

1.5 Analýza licencí

Licence slouží k možnostem užívání, upravování nebo redistribuování softwaru. Licence jsou součástí software, jelikož software je chráněn autorským zákonem. Autor si během psaní díla může vybrat pod jakou licencí bude distribuovat své dílo. Autor vybírá, zda bude vydán zdrojový kód jako otevřený software nebo proprietární (uzavřený) software. Otevřený software umožňuje ostatním nahlédnout do zdrojového kódu a podle licence s ním dále nakládat. Autor může určit zda se jeho dílo může používat v komerčních řešeních, pořizovat kopie nebo následně upravovat, zda vytvořená kopie musí mít stejnou licenci, uvedení původního autora a podobně.

Proprietární software zpravidla nemívá kód k nahlédnutí a často autor nabízí využít hotový program za poplatek. Licence u proprietární verze závisí na autorovi. S dílem nelze volně nakládat, nelze ho upravovat, distribuovat nebo používat v komerčních řešeních pokud nebylo sjednáno jinak. Možnosti, co je s dílem možné dělat, se upravují ve smlouvě nebo v end user license agreement (EULA) [31]. V roce 2017 firma Black Duck uskutečnila analýzu využití licencí v různých softwarových produktech a knihovnách, výsledky pro 70 miliónů projektů lze vidět na obrázku 1.17.



Obr. 1.17: Rozdělení licencí podle využití v softwaru [33].

GNU General Public Licence v2.0

GNU General Public typ licencí patří k nejpoužívanějším licencím, první verze vyšla v roce 1989. Hlavní cíl první verze bylo poskytnout lidem k binární verzi souboru také čitelný zdrojový kód. Druhá verze vyšla v roce 1991. Hlavní změnou je klauzule "svobodu nebo život". Cílem klauzule bylo, že pokud má autor v podmínkách omezení šíření licence, povolení šířit dílo pouze v binární formě, tak by dílo nemělo být šířeno vůbec [32].

GNU General Public Licence v3.0

Verze 3 vyšla 29. června 2007 po veřejných diskuzích, je rozšířením verze 2. Byla přidána definice zdrojových kódů, vztahy k softwarovým patentům, řešení porušení autorských práv. Jedna z důležitých změn je možnost přidat další požadavky nebo podmínky na šíření díla [32].

GNU A□ero General Public License version 3

Affero licence vychází z GNU General Public licence, která je doplněna o využití přes počítačovou síť. Zdrojový kód musí být zpřístupněn uživateli i při užívání přes počítačovou síť [32], [33].

Apache License 2.0

Licence Apache 2.0 vyšla v roce 2004 a je alternativou k GNU General Public Licence v3.0. Licence umožňuje uživateli zachovat autorství a zároveň se zříká odpovědnosti za škody způsobené softwarem. Stejně jako u GPLv3 umožňuje licence používat software ke kopírování, změnám a distribuci díla [34].

Proprietární software

Proprietární licence se od ostatních zmiňovaných licencí liší tím, že zpravidla nemá dostupný zdrojový kód. Veškeré právní úpravy si autor upravuje ve smlouvě nebo v end user license agreement (EULA). Vývojář určuje cenu, může nabízet produkt zdarma nebo za poplatek, ať už placený jednorázově, za určité časové období nebo za přidružené služby. Uživatel nemůže do produktu zasahovat, nemůže ho distribuovat nebo jinak používat, pokud mu to neumožňuje EULA [35].

1.6 Shrnutí

V teoretické části byly porovnány nástroje spustitelné pomocí konzole nebo s vlastním grafickým rozhraním. Následně bylo porovnáno šest Android aplikací. U všech porovnávaných nástrojů byly vypsány protokoly, které nástroje používají pro detekci a licence, které byly popsány v samostatné kapitole. U zmíněných licencí byly vypsány nejdůležitější vlastnosti, pro přehlednost je lze vidět i v tabulce 1.1.

U konzolových nástrojů bylo vybráno devět zástupců pro porovnání. Byly vybrány na základě osobních zkušeností a především na základě doporučení externího konzultanta. Nástroje byly prozkoumány a bylo zjištěno, jaké protokoly používají, jaké operační systémy podporují a pod jakou licencí jsou dostupné. Výsledky jsou prezentovány v tabulce 1.2 a 1.3.

U nástrojů s grafickým rozhraním bylo vybráno devět zástupců pro porovnání jako u konzolových nástrojů. Opět bylo zjištěno jaké protokoly používají k detekci, jaké operační systémy podporují a pod jakou licencí jsou dostupné. Výsledky jsou pro přehlednost uvedeny v tabulce 1.4 a 1.5.

U Android aplikací bylo vybráno šest zástupců pro porovnání. Byly vybrány na základě definovaných požadavků (licence, programovací jazyk) z externí firmy. Ve vybraných aplikacích se vyskytují i takové, které nabízejí pouze základní skenování sítě a následně s více možnostmi jak oskenovat síť i samotné zařízení. Většina nástrojů nemá dostupnou dokumentaci nebo otevřený zdrojový kód. Nebylo tedy možné zjistit jaké protokoly jsou používány ke skenování sítě.

Tab. 1.1: Porovnání licencí [32] [33] [34].

	GNUv2	GNUv3	Affero3	Apache2	Proprietární
Otevřený software	✓	✓	✓	✓	
Komerční použití	✓	✓	✓	✓	EULA
Možnost změn	✓	✓	✓	✓	
Právo distribuce	✓	✓	✓	✓	
Vlastní použiti	✓	✓	✓	✓	EULA

Tab. 1.2: Podporované operační systémy u konzolových skenerů.

Nástroj	Windows	Linux	Mac
Dmitry	✓	✓	✓
MASSCAN	✓	✓	✓
Nikto		✓	
Netcat		✓	
Nmap	✓	✓	✓
OpenVAS		✓	
Sn0int	✓	✓	✓
Scapy	✓	✓	✓
ZMAP		✓	

Tab. 1.3: Podporované protokoly u konzolových skenerů.

Nástroj	TCP	UDP	SCTP	ICMP
Dmitry	✓			
MASSCAN	✓			
Nikto	✓			
Netcat	✓	✓		
Nmap	✓	✓	✓	✓
OpenVAS	✓	✓		✓
Sn0int	✓			✓
Scapy	✓	✓	✓	✓
ZMAP	✓	✓		✓

Tab. 1.4: Podporované operační systémy u skenerů s grafickým rozhraním.

Nástroj	Windows	Linux	Mac
Angry IP scanner	✓	✓	✓
Archery	✓	✓	✓
Dipiscan	✓		
Nessus	✓	✓	✓
Netcrunch	✓	✓	✓
NMAPSI4		✓	
PortScanner	✓		
Rumble	✓	✓	✓
Umit Network scanner	✓	✓	

Tab. 1.5: Podporované protokoly u skenerů s grafickým rozhraním.

Nástroj	TCP	UDP	SCTP	ICMP
Angry IP scanner	✓	✓		✓
Archery	✓	✓	✓	✓
Dipiscan	✓			
Nessus	✓			✓
Netcrunch	✓	✓		✓
NMAPSI4	✓	✓	✓	✓
PortScanner	✓	✓		✓
Rumble	✓			
Umit Network scanner	✓	✓	✓	✓

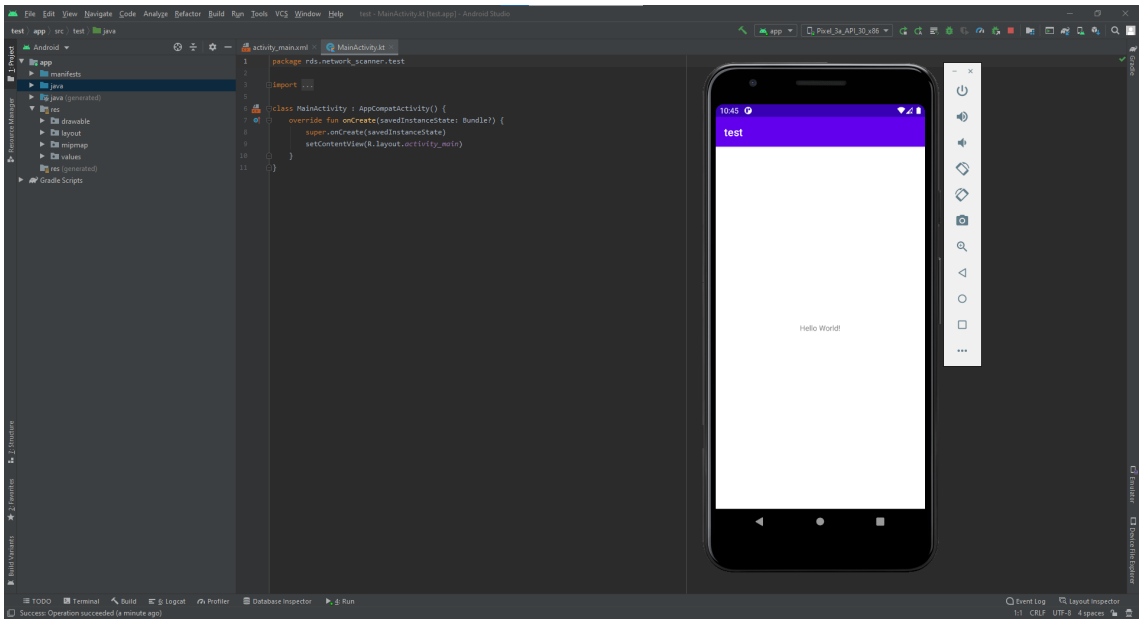
2 Praktická část

Praktická část práce začíná realizací experimentálního pracoviště, popsáním prvků nacházejících se na sítí, přípravou Android studia k vývoji a zprovoznění testovací aplikace. Následuje porovnání aplikací popsaných v kapitole 1.3, kde je porovnáno vytížení systému, dobu skenování a dostupné výsledky. Poslední a hlavní část je věnována vlastnímu návrhu aplikace a samotné implementaci s popisem nalezených problémů, důležitými částmi kódu a výsledky nakonec.

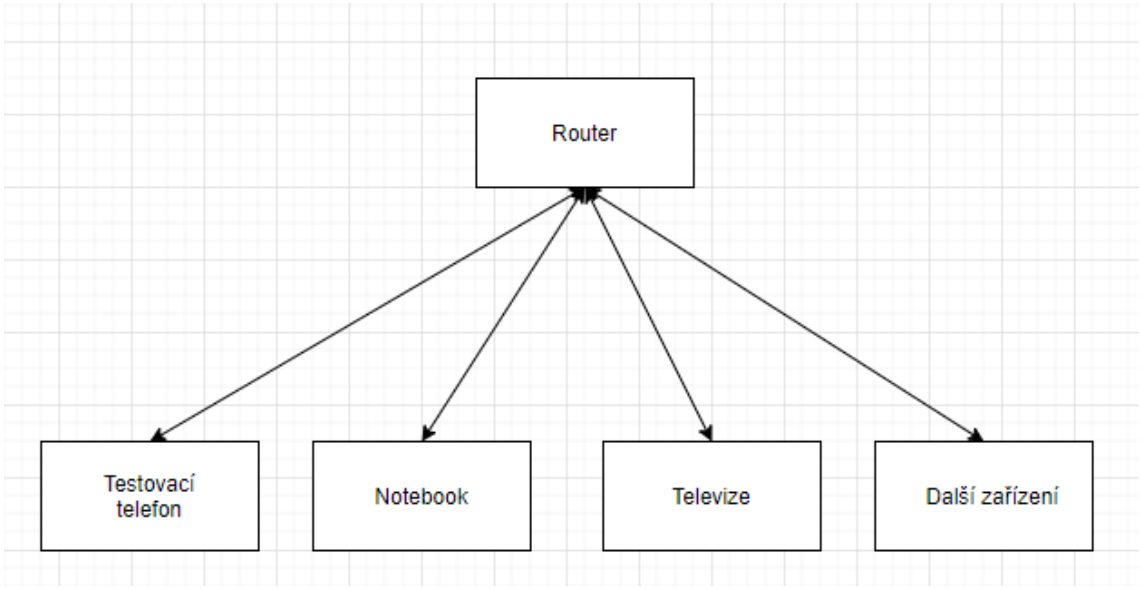
2.1 Realizace experimentálního pracoviště

Pro realizaci experimentálního pracoviště byla využita domácí síť. Na síti se nachází přibližně 13 zařízení, počet se měnil v závislosti na počtu osob v domácnosti. Mezi zařízení, která se stabilně nacházejí ve testovací WiFi síti jsou televize, notebook, mobilní telefon, kávovar, myčka a hostující router. Porovnání aplikací probíhalo na mobilním telefonu Redmi Note 8 od firmy Xiaomi s OS Android 10 (verze 10 QKQ1.200114.002). Telefon je vybaven 4GB RAM a 8 jádrovým procesorem Octa core Max 2.01 GHz.

Vývoj vlastní aplikace probíhal v Android studiu, které slouží jako vývojové prostředí pro programování aplikací pro zařízení s operačním systémem Android. Studio podporuje psaní aplikací ve dvou jazycích, konkrétně v Javě a Kotlinu. Vytvořené aplikace mají možnost být testovány buď na virtualizovaném telefonu nebo po propojení s fyzickým zařízením.



Obr. 2.1: Ukázka Android studia po vytvořeném novém projektu s puštěným virtuálním telefonem.



Obr. 2.2: Schéma připojených zařízení na síti.

Testovací aplikace

Za účelem ověření, že Android studio bylo správně zprovozněno a testovací telefon byl propojen k Android studiu, byla vytvořena jednoduchá testovací aplikace. Cíl aplikace bylo zjištění počtu kliknutí na tlačítko, výsledné číslo je vypisováno do textového pole na obrazovce. Zdrojový kód testovací aplikace lze vidět na obrázku 2.1.

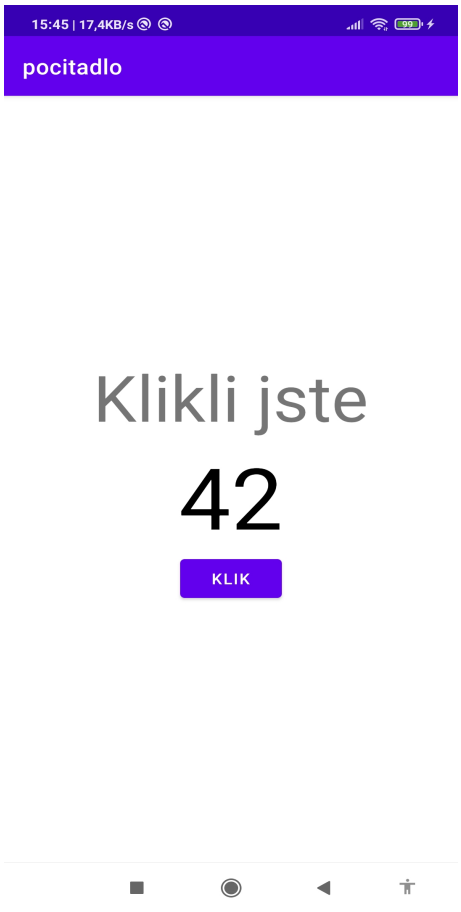
Testovací aplikace byla vytvořena v Android studio pomocí šablony *Empty activity*. Do šablony byl následně přidán kód aplikace. První úpravou bylo přidání tlačítka a dvou textových polí do souboru activity_main.xml. První textové pole slouží k vypsání textu klikli jste a druhé textové pole slouží k vypsání samotného čísla.

Po upravení uživatelského rozhraní bylo potřeba nastavit funkcionalitu tlačítka a druhého textového pole. Byl přidán kód na poslouchání co se má stát s tlačítkem po kliknutí a následná úprava textového pole, kterou lze vidět níže.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // Proměnná zachovávající počet kliknutí
        var pocetKliuti = 0
        // Namapování tlačítka
        val button = findViewById<Button>(R.id.btnIncrementByOne)
        // Kód co se má stát s tlačítkem po kliknutí
        button.setOnClickListener{
            // Namapování textového pole
            val text = findViewById<TextView>(R.id.txtCounter)
            // Připočítaní kliknutí
            pocetKliknuti++
            // Změna textového pole na počet aktuálních kliknutí
            text.setText(pocetKliknuti.toString())
}    }    }
```

Výpis 2.1: Zdrojový kód testovací aplikace (MainActivity.kt).

Obr. 2.3: Ukázka spuštěné testovací aplikace.



Obr. 2.4: Testovací aplikace po několika kliknutí.

2.2 Ověření funkčnosti pracoviště

Před samotným návrhem, resp. implementací bylo potřeba zmapovat trh dostupných aplikací a otestovat je. Vybrané nástroje, které byly popsány v kapitole 1.3 byly otestovány na fyzickém zařízení Xiaomi Redmi Note 8, popsaném výše. U aplikací nebyly hodnoceny pouze na technické parametry, ale také na možnost využití aplikací v komerčním použití ve vlastní aplikaci.

Testované nástroje byly dostupné v Android obchodě Google play, odkud byly staženy a nainstalovány na telefon. Nástroje byly dostupné zdarma. U některých byla možnost dokoupit funkce navíc, tyto neovlivňovaly výsledky porovnání, pouze rozšiřovaly možnosti skenování.

Aplikace byly porovnány ze dvou hledisek, první porovnání bylo z technického hlediska, zjištění jak dlouho trvá sken sítě a jaké je využití procesoru mobilního telefonu. Druhé hledisko bylo subjektivní, a to porovnání uživatelského rozhraní, jak se chová, ovládá a jaké je rozložení výstupů skenů.

U technického zpracování byly porovnány následující parametry RAM, doba skenování sítě, zda byla zjištěna IP a MAC adresa zařízení připojených do sítě, odezva zařízení a zjištění výrobce nalezených zařízení. Během porovnání byly zjištěny komplikace, jelikož Android přestává podporovat přístup k hardwaru zařízení a přesné hodnoty nelze získat bez velkého zásahu do telefonu. Bylo odzkoušeno několik návodů na zobrazení využití CPU i RAM Android zařízení, nicméně i přes zmíněné pokusy nedošlo k úplnému vyřešení nedostupnosti těchto systémových informací. Dále byly vyzkoušeny aplikace, které mají dané hodnoty zjistit, ale rovněž bez úspěchu.

Některé aplikace ukazovaly přibližné využití RAM aplikace. Kvůli komplikacím zmíněným v minulém odstavci byly hodnoty RAM a doba skenování získány pomocí deseti měření. Naměřené hodnoty byly zapsány do tabulky, výsledná hodnota byla zprůměrována. Dobu skenování nebylo také možné změřit zcela přesně a byla provedena měření obdobně jako při využití RAM. Bylo provedeno 10 měření na stejném zařízení, na stejné síti a výsledná hodnota byla zprůměrována.

Důležitý parametr při porovnávání bylo zjištění IP a MAC adresy a zjištění všech dostupných zařízeních, které se v síti nachází. Další parametry, které byly zjištěny byla odezva a výrobce zařízení při prvotním skenování. Některé aplikace jsou schopny zjistit dané hodnoty po rozkliknutí detailu zařízení a důkladnějším skenování.

Shrnutí naměřených výsledků

Výsledky testování jsou zobrazeny v tabulce 2.1. Všechny aplikace splnily hlavní podmínku a to zjištění IP a MAC adresy. U Network scannerů firmy Easy Mobile a First row byla zjištěna i odezva zařízení v jednotkách milisekund. Kromě skeneru Who's ony my wifi byl zjištěn i výrobce zařízení, ovšem ne u všech skenovaných zařízení.

Při porovnávání hodnot využití RAM nelze vidět velké rozdíly. Jediný větší rozdíl lze vyčíst z naměřených hodnot u aplikace Fing, která využívá RAM o přibližně 31 MB méně. U testování času doby skenování byly hodnoty více rozptýlené, nejrychlejší aplikace Who's on my wifi oskenovala síť za 3,2 sekundy, výsledek bude způsoben malým počtem zjištěných informací. Naopak nejdelší skenování bylo u aplikací s největším počtem zjištěných informací.

Největší nevýhodou aplikací je samotná licence, kterou nelze použít v komerčním použití. Žádná aplikace nenabízela otevřený zdrojový kód k nahlédnutí. Z tohoto důvodu bylo potřeba hledat další možnosti nebo vytvořit vlastní aplikaci.

Tab. 2.1: Porovnání aplikací na Redmi note 8.

	Network scanner (ES)	Network scanner (FR)	Fing	Who's on my wifi	WiFiman	WiFi Monitor
Využití RAM[MB]	~185	~192	~137	~161	~173	~163
Doba skenování[s]	8,7	10,4	7,2	3,2	6,3	4,8
Zjištěna IP adresa	✓	✓	✓	✓	✓	✓
MAC adresa	✓	✓	✓	✓	✓	✓
Odezva zařízení	✓	✓				
Výrobce zařízení	✓	✓	✓		✓	✓

2.3 Vlastní návrh aplikace pro skenování

Hlavní myšlenkou aplikace bylo nalézt všechna zařízení připojená k lokální počítačové síti. Nalezená zařízení identifikovat, zjistit IP a MAC adresu, popřípadě další informace, které lze získat. Samotná aplikace bude připravena jako modul pro aplikaci, která má sloužit k nalezení hrozeb v mobilním telefonu nebo na lokální síti. Proto má aplikace jednoduché uživatelské rozhraní, v budoucnu je plánováno využití výstupů tohoto modulu přes aplikační rozhraní.

Během analýzy dostupných nástrojů na trhu pro různé operační systémy a samotné Android zařízení byla nalezena jediná aplikace s otevřeným zdrojovým kódem, u které stále dochází k vývoji a je napsána požadovaném v jazyce kotlin. Zmíněná aplikace s otevřeným zdrojovým kódem se nazývá Ning. Další dvě aplikace s otevřeným kódem jsou obě napsány v Javě a nejsou nadále vyvíjeny. Aplikace dostupné pouze přes obchod Google Play nemají otevřený zdrojový kód a neumožňují používat jejich řešení v komerční podobě, díky licencím a nemožnosti zjistit jakým způsobem fungují.

Po zhodnocení těchto podmínek a pozdějším testováním aplikace Ning, která nebyla schopna zjistit ostatní zařízení na lokální síti se zapnutou izolací klientských stanic na routeru. Bylo tedy rozhodnuto pro napsání vlastní jednoduché aplikace, která tento problém dokáže obejít a zjistit zařízení i na takových sítích.

2.4 Vlastní implementace aplikace

Kapitola implementace je rozdělena na dvě části. První, menší část, zahrnuje popis grafického rozhraní a druhá, hlavní, se zabývá samotným skenováním, způsoby které jsem zkoušel nebo byly doporučovány návody na internetu. U různých řešení jsou napsány důvody proč nefungují, jaké mají nevýhody a podobně. Různé postupy implementací jsou sepsány v pořadí v jakém byly zkoušeny.

Během samotné implementace vlastní aplikace se vyskytlo několik problémů, které jsou rozebrány u jednotlivých řešení níže. Uvedená řešení nejsou dokonalá a obsahují nedostatky, kvůli nedostatku času nebo ne zcela funkčním řešením jako celku. U každého řešení je seznam výhod, nevýhod a nedostatků v kódu, které byly nalezeny během testování. Samozřejmě další testování by odhalilo další chyby k opravení.

Jako vývojové prostředí bylo vybráno Android studio a jazyk byl vybrán Kotlin, který vznikl upravením Javy. Kotlin má výhody jednodušších zápisů kódu než má Java, zvláště při psaní XML souborů pro uživatelská rozhraní, psaní funkcí nebo řešení typování proměnných. Android studio je volně dostupné a nabízí funkce ulehčující vývoj aplikací. Umožňuje zobrazení grafického rozhraní při psaní kódu nebo automaticky píše kód při využívání nástrojů pro jejich tvorbu.

Grafické rozhraní

Grafické rozhraní aplikace je velice jednoduché, protože hlavní cíl práce bylo zmapování dostupných aplikací a napsání modulu na detekci zařízení nacházejících se na lokální síti. Grafické rozhraní se skládá z následujících věcí:

- app_bar
- Button
- CoordinatorLayout
- ProgressBar
- RecyclerView
- SnackBar

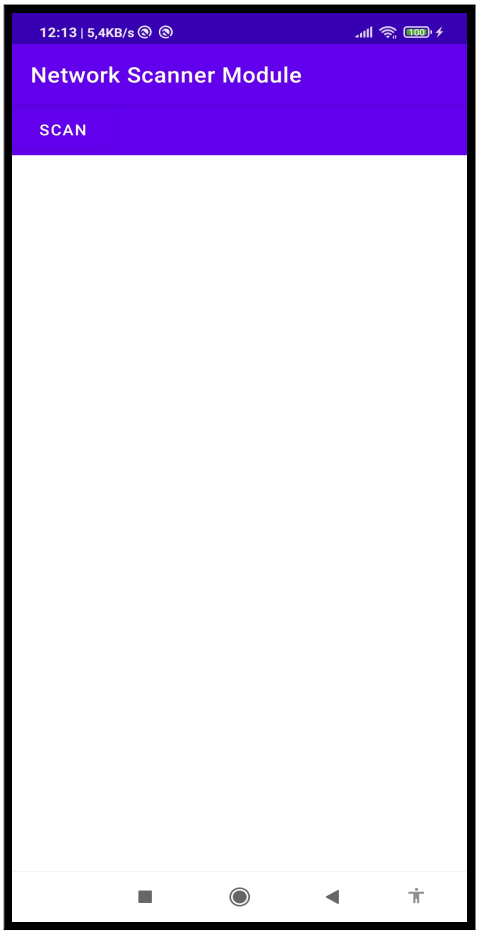
CoordinatorLayout je hlavní část grafického rozhraní, udržuje rozložení ostatních komponentů na místě určených při spouštění. Slouží k udržování vzdáleností komponent od sebe v závislosti na velikosti displeje. App bar slouží k uchycení tlačítka pro spouštění skenování, ale může sloužit k budoucímu vývoji aplikace a přidávání dalších funkcí. Samotné tlačítko slouží ke spouštění skenování.

Progress bar slouží k upozornění uživatele, že probíhá skenování sítě a po dokončení kolečko zmizí. Snackbar není vidět v grafickém rozhraní napsané v XML souboru, ale slouží k vypsání informace o úspěšném dokončení sítě nebo upozornění uživatele, že není připojen na WiFi.

Recycler view slouží k vypsání výsledků do graficky přívětivé podoby s možností prohlížení výsledků, které se nevejdou na obrazovku. Každý výsledek je uspořádán do vlastního boxu, který je definován v samostatném XML souboru viz příloha A.3. Recycler view obsahuje prostor na obrázek reprezentující typ zařízení a dvě textová pole. První textové pole vypisuje IP adresu, druhá MAC adresu a výrobce zařízení. Pokud je recycler view prázdný nelze vidět.

Zdrojové kódy uživatelského rozhraní jsou sepsány ve výpisech 2.3 a 2.4. Android studio dokázalo usnadnit psaní kódu, díky vestavěným funkcím. Studio dokáže zobrazit uživatelské rozhraní a umožňuje psát XML kód ručně nebo pomocí nástrojů. Posouvání jednotlivých prvků lze provést snadno přes vizualizaci rozhraní.

Výslednou aplikace lze vidět na obrázku 2.4 před zapnutím skenování. Po zapnutí skenování se v úspěšném případě vypíše do recycler view IP a MAC adresa nalezených zařízení. Vzhled aplikace se během zkoušení různých možností vlastní aplikace neměnila a proto je tu uvedena finální verze aplikace.



Obr. 2.5: Grafické rozhraní vlastní aplikace.

2.5 Skenování zařízení na síti

Skenování zařízení připojených na síť je dobré, zvlášť v dnešní době chytrých zařízení. Lidé dnes vlastní notebook, chytrý telefon, chytrou televizi nebo další zařízení, které jsou připojené k WiFi síti. Pokud majitel nemonitoruje vlastní síť může se nechtěný návštěvník bez povšimnutí připojit na jeho síť a využívat ji. V lepším případě připojený uživatel používá internet na věci, které nejsou zákonem zakázány. V horším případě je využije na trestnou činnost a pak majitel sítě těžko vysvětluje, že to nebyl on. Skenery v dnešní době mohou upozorňovat provozovatele sítě na neznámá zařízení a identifikovat je. Popřípadě i zablokovat přístup k síti.

Získání vlastní IP adresy

Jedna z prvních věcí, kterou bylo potřeba získat je IP adresa zařízení. K tomuto nám posloužila knihovna WiFimanager, která je dostupná v Android studiu. Knihovna získává informace o IP adrese pouze k připojené WiFi síti a nevrátí adresu v případě připojení k datům mobilního operátora. Knihovna obsahuje funkci pro zjištění, zda je zařízení připojené k síti *isWifiEnabled*, která vrací hodnoty *pravda* a *nepravda*.

Funkci, která vrací IP adresu lze vidět ve výpisu 2.5. Funkce má nedostatek, že vrátí prázdný řetězec v případě, že uživatel není připojen na WiFi. Později je funkce zakomponována do hlavní funkce, které volá *isWifiEnabled* pro zjištění, zda je uživatel připojen k WiFi síti.

```
1 private fun getIP() : String{
2     val wifiManager = getApplicationContext().getSystemService(
    Context.WIFI_SERVICE) as WifiManager
3     val ip =  Formatter.formatIpAddress(wifiManager.connectionInfo.
    ipAddress)
4     return ip
5 ]
```

Výpis 2.2: Funkce pro zjištění IP zařízení.

2.5.1 Nalezení zařízení pomocí ping příkazu

Prvním pokusem během zjištění informací o zařízení bylo použití klasického příkazu ping, který je dostupný v každém operačním systému. Vývojové prostředí Android studia nabízí knihovnu InetAddress [36], která obsahuje funkci *isReachable(int časový limit)*. Funkce funguje obdobně jako příkaz ping. Nejprve pošle ICMP žádost, kterou nahradí TCP Echo request v případě chybové hlášky. V případě, že se jeden z příkazů vrátí do časového limitu, je výsledek funkce *true*, v opačném případě vrátí pozitivní výsledek *false*. Časový limit je udáván v milisekundách.

Funkce potřebuje parametr IP adresy, aby vytvořila seznam zařízení k nalezení. Návratová hodnota je list zařízení, která byly dostupná v daném časovém limitu. List obsahuj cestu k obrázek a dvou textových polí, které slouží k zachycení informací o zařízení. Časový limit byl nastaven na hodnotu dvacet milisekund, která by měla být postačující pro odezvu mezi zařízeními na lokální síti. Parametr IP adresy se rozdělí na hodnoty rozdělené tečkou, které se později spojí a vytvoří základní adresu sítě. Předpokládaná maska sítě je *255.255.255.0*, pokud je jiná, funkce i tak počítá s danou hodnotou.

Po zjištění adresy podsítě je vyvolán cyklus pro hodnoty 1 až 254 a pro každé zařízení je vyslána ICMP výzva. V případě kladné odpovědi do časového limitu se hodnota IP adresy zapíše do výsledného seznamu. Není možné získat žádné dodatečné informace díky této metodě.

```
private fun ping_scan(ip: String): List<recycler_data_item> {
    val splitIP = ip.split(".")
    val subnet = splitIP[0] + "." + splitIP[1]+ "." + splitIP[2]
    val recyclerList = ArrayList<recycler_data_item>()
    for (i in 1..254){
        val ping_ip = subnet + "." + i
        if (InetAddress.getByName(ping_ip).isReachable(20)) {
            val item = recycler_data_item(
                R.drawable.ic_question_mark,
                "IP: $ping_ip",
                "Additional info: Unknown"
                )
            Log.i("Scanning", "IP is reachable ($ping_ip)")
            recycler_list += item
    } else {
            Log.i("Scanning", "IP not reachable ($ping_ip)")
        }
    }
return recyclerList
}
```

Výpis 2.3: Funkce pro zjištění dostupnosti zařízení pomocí ping příkazu.

Výhody a nevýhody

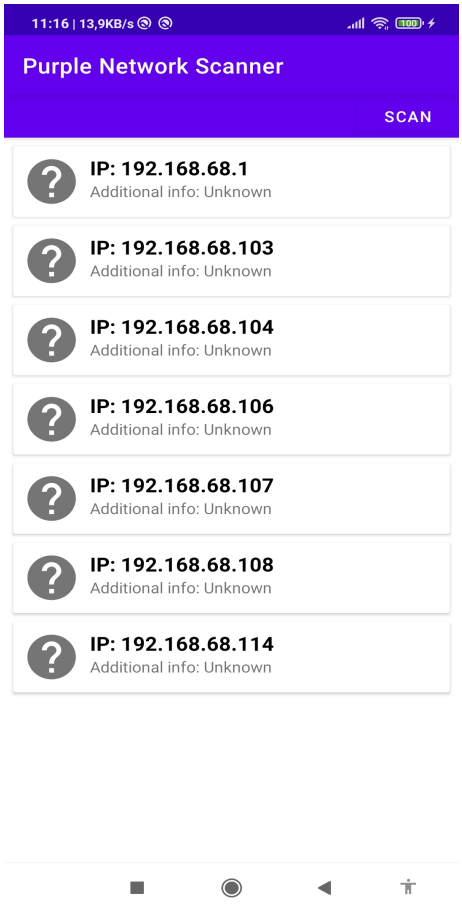
K největším výhodám tohoto způsobu patří jednoduchost zprovoznění, funkčnost na jakémkoliv Android zařízení a schopnost přizpůsobit kód i pro ostatní operační systémy.

Největší nevýhoda je nedostatek nalezených informací o zařízení. Další nevýhodou může být nestabilita počtu oskenovaných zařízení. U nižších hodnot časové

odezvy nemusí zařízení stihnout odpovědět do limitu a aplikace zařízení nezapíše do tabulky nalezených zařízení. Pokud funkce nepracuje na více vláknech doba skenování se značné prodlužuje. Zároveň některá zařízení mají mohou mít zapnutou funkci, která neodpovídá jiným zařízením na ping žádosti, díky tomu nebude možné zařízení detekovat.

Routery umožňují zapnout funkci, která odděluje uživatele na síti. Oddělením je myšleno zakázání komunikace mezi uživateli. Povolena je komunikace pouze mezi zařízením a přístupovým bodem. Pokud by se zařízení pokusilo komunikovat s ostatními zařízeními na síti, tak to router nepovolí a cílové zařízení se bude jevit jako nedostupné. Tato funkce se používá na firemních veřejných sítích nebo obecně na veřejných sítích pro snížení možnosti napadení uživatelů. Samozřejmě se dá komunikace stále odchytávat, zvlášť na veřejných WiFi sítích [37].

Výsledný sken sítě vypadá přibližně jako na obrázku 2.5.1. Výsledný počet zařízení se mohl měnit v závislosti na době, jakou jim trvala odezva. Zejména při prvních skenováních sítě chybělo více zařízení a bylo potřeba vícekrát oskenovat síť.



Obr. 2.6: Výsledek skenování použitím ping skenování.

2.5.2 Čtení ARP mezipaměťi

Jedním z dalších možných přístupu detekce zařízení bylo čtení mezipaměti nacházejícího se v souboru ARP s cestou */proc/net/arp*. Tabulka ARP slouží k ukládání záznamů o zařízení na síti a jejich informací jako je MAC adresa, na jakém rozhraní se nachází a podobně. Uloženou IP adresu překládá do MAC adresy, díky které můžou komunikovat [38].

Funkce pro získání informací z ARP tabulky by byla jednoduchá. Funkce by začínala rozesláním paketu s žádostí o odpověď na lokální multicastovou adresu. Odpovědi by se následně uložily do tabulky a pak by stačila funkce, která by daný soubor četla řádek po řádku a vybírala potřebné informace o zařízení do tabulky.

Problém nastává u novějších verzí operačního systému Android. Konkrétně od verze Androidu 10 s API verzí 29. Dochází k velkým bezpečnostním úpravám. Zejména zákaz spouštění aplikací na pozadí a zákaz k přístupu do složky */proc/net*. Složka slouží k uchovávání informací o různých připojeních k internetu, informací o IP a MAC adresách, se kterými zařízení komunikovalo. Tato změna se dotkla spousty aplikací, zejména těch poskytující virtuální soukromé sítě a proxy připojení. Některé dostupné skenery používaly tabulku ke zjišťování informací.

Díky tomuto nebylo možné nijak uskutečnit, alespoň částečně úspěšně zhotovené řešení. Pokaždé při pokusu o čtení z tabulky aplikace vyhodila chybovou hlášku, která upozorňuje, že uživatel nemá přístup k */proc/net/arp* a pokud nebyla ošetřena výjimka, tak se aplikace ukončila [39].

Výhody a nevýhody

Výhody a nevýhody se zde celkem špatně vyjmenovávají. Z teoretického hlediska by jedna z výhod mohla být snadná implementace, zejména na linuxových zařízeních, kde je povolený přístup k ARP tabulce. Funkce by mohla vypadat tak, že se pošle paket na multicastovou adresu a po chvíli by se mohla číst tabulka řádek po řádku a vybírat z toho vybrané hodnoty. Nevýhodou je z pohledu Android zařízení to, že je zakázaný přístup k tabulce a možné zakázání posílání paketů na multicastovou adresu ze strany routeru.

Jedna z dalších možných otázek by bylo, zda by to fungovalo na sítích se zapnutou funkcí pro oddělení uživatelů (Client Isolation), které znemožňuje komunikaci mezi sebou. Pravděpodobně by tato funkce zamezila komunikaci a nebyl by možný použít tento způsob pro veřejné sítě.

2.5.3 Používání NMAP nebo ARP příkazu

Dalším možným řešením, které bylo doporučováno a fungovalo na starších verzích operačního systému byly příkazy *nmap* a *arp*. Oba příkazy se standardně používají na Linuxu, slouží ke zjišťovaní informací o lokální síti. V případě *NMAP* dokáže skenovat i další sítě kromě připojené podsítě. Arp dokáže zjišťovat pouze základní informace o zařízení jako jsou IP a MAC adresa. Nmap dokáže díky možnosti přepínačů a podpoře skriptování zjistit daleko více o zařízeních.

Během zkoušení příkazů uvedených ve výpisech níže bylo zjištěno, že uvedené příkazy nejsou dostupné v základní verzi Androidu. Bylo by potřeba implementovat balíčky, které by obsahovaly dané příkazy. Během zjišťování informací o tom jaké balíčky přidat pro zprovoznění, bylo zjištěno v různých diskuzních fórech, že dané balíčky už jsou zastaralé a je lepší používat jiný příkaz. Tento příkaz byl použit ve finální verzi, která je rozebrána dále v práci.

Jako u předchozích způsobů, otázka je podobná a to, zda by uvedené příkazy fungovaly na veřejné síti se zapnutou funkcí na oddělení uživatelů, díky které nemůžou mezi sebou komunikovat. Bylo by potřebné udělat testy, které by to prokázaly nebo vyvrátily.

Uvedené příkazy slouží jako příklad, jak by mohly vypadat zprovozněné. V uvedené formě fungují na operačním systému Linux při použití v příkazovém řádku. Při spouštění příkazu na Android zařízení aplikace byla ukončena chybovou hlášku, který říkal, že daný příkaz neexistuje.

```
1 val proc = Runtime.getRuntime().exec("nmap -sP $ip/24")
```

Výpis 2.4: Příkaz nmap pro skenování sítě.

```
1 val proc = Runtime.getRuntime().exec("arp -a")
```

Výpis 2.5: Příkaz ARP pro skenování sítě.

Výhody a nevýhody

Jedna z možných výhod u Nmapu by mohla být zprovoznění skriptování a používání skriptů pro zjištění informací o cílových zařízeních. Nmap má velkou skupinu uživatelů a existuje spoustu článků, návodů a skriptů, které by mohly být použity pro skenování uživatelů.

Výhoda příkazu *arp* je jednoduchost a jasně rozlišitelné výsledky. Nevýhoda je, že během skenování není možnost zjistit více informací o zařízení. Nalezené informace budou vždy IP adresa, MAC a název rozhraní. Další menší nevýhodou ARP příkazu je, že už není nadále vyvíjen.

2.5.4 Použití network service discovery

Jeden z dalších způsobů jak detekovat ostatní zařízení na síti je použitím Network service discovery. Tato funkce slouží k detekování zařízení, které ji také podporují. Nejčastěji je podporována tiskárnami, servery a jinými mobilními telefony. Network service discovery funguje na principu *Domain name system service discovery*, díky kterému může zařízení žádat o služby ostatní zařízení na síti, pokud danou službu nabízejí.

Tato funkce může být vhodná pro chytré domácnosti, které mají zařízení nabízející služby např. tiskárny, kávovary, myčky, televize, lednice a podobně. Dále se může použít ke sdílení souborů mezi zařízeními nebo hraním videoher společně na různých zařízeních [40] [41].

Výhody a nevýhody

Uvedenou možnost skenování zařízení jsem nezkoušel, ale byla možná vidět při zkoumání aplikace Ning. Jednou z hlavních výhod je možnost komunikace se zařízeními a zjištění jakou službu nabízí.

Nevýhody jsou, že všechny zařízení nepodporují danou službu a nebylo by možné detekovat všechna zařízení na síti. Další nevýhodou je, že pokud je na síti zapnuta funkce na oddělení uživatelů, nemůžou se šířit nabídky služeb od ostatních zařízení a není možnost detekovat ostatní zařízení.

2.5.5 Použití ip příkazu

Poslední testovanou variantou pro detekci zařízení je příkaz *ip neighbour*. Příkaz funguje obdobně jako *arp*, který je jeho předchůdce. Příkaz slouží k vypsání známých spojení, ukládá je do ARP tabulky, do které není jinak přistup. Umožňuje přidávat nebo mazat záznamy do tabulky.

Příkaz je dostupný v základní verzi Androidu a není potřeba přidávat žádné další balíčky. Samotný příkaz je dostupný v základní výbavě operačních systémů Linux a to konkrétně v balíčku net-tools.

```
1 val proc = Runtime.getRuntime().exec("ip neighbour show")
2 proc.waitFor()
3 val stdInput = BufferedReader(InputStreamReader(proc.inputStream))
```

Výpis 2.6: Příkaz ip neighbour pro skenování sítě.

Výše uvedený příkaz slouží jako ukázka pro zprovoznění a je třeba ho obalit funkcí a dalším zpracováním. Proměnná *proc* slouží jako prostředník k puštění příkazu pro *ip neighbour*. Následuje funkce *waitFor()*, která čeká na dokončení procesu. Samotný

proces vypisuje obsah na standardní výstup, který předáváme do proměnné *stdInput*. Výstup je uložen v proměnné a skládá se z několika řádků, které obsahují informace o IP adrese, MAC adrese a rozhraní zařízení.

Výhody a nevýhody

Největší výhoda je možnost příkazu je možnost k přístupu do ARP tabulky a čtení jejich hodnot, které usnadňuje získávání informací. Další výhoda je, že příkaz je dostupný nejen na Android zařízeních, ale i v linuxových operačních systémech. Nevýhoda je, že nejsou nalezeny informace o vlastním zařízení.

2.6 Vlastní implementace

Tato kapitola se bude věnovat shrnutí vlastní implementace aplikace, sepsání důležitých částí kódu jako celku, vysvětlení proč jsou věci takto řešeny a končí možnými vylepšeními aplikace do budoucna, které nebylo čas dodělat. Části kódu budou rozebrány podle souboru v jakém se nacházejí. Budou vysvětleny propojení mezi jednotlivými soubory. Uživatelské rozhraní nebude v této kapitole zmíněno, protože má vlastní kapitolu výše a nachází se v souboru *activity_mmain.xml*.

MainActivity

Soubor *MainActivity* je základním stavebním kamenem aplikací, slouží při prvotním vytváření aplikace. Pokaždé když uživatel pustí jakoukoliv aplikace spustí se díky tomuto souboru. Při spouštění aplikace je zavolána funkce *onCreate*, která inicializuje celou aplikace a následně načte grafické rozhraní ze souboru *activity_mmain.xml*, který obsahuje hlavní grafické rozhraní aplikace.

```
1 override fun onCreate(savedInstanceState: Bundle?) {
2     super.onCreate(savedInstanceState)
3     setContentView(R.layout.activity_main)
```

Výpis 2.7: Spouštění aplikace.

Samotné spouštění aplikace už inicializuje první proměnné. Komponenty grafického rozhraní je nejlepší si vypsat do proměnných pro snazší manipulaci. Aplikace upravuje tři komponenty. *recycler_view*, který se naplňuje hodnotami ze skenování, *progress* k zobrazení zda funkce probíhá a *devices*, které slouží jako textové pole informující o počtu nalezených zařízení. Poslední je *scan* tlačítko, které slouží k začnš proces skenování sítě.

```
val RECYCLER_VIEW = findViewById<RecyclerView>(R.id.recycler_view)
val PROGRESS = findViewById<ProgressBar>(R.id.progressBar)
val DEVICES = findViewById<TextView>(R.id.textView)v
val SCAN = findViewById<Button>(R.id.scan)
```

Výpis 2.8: Načtení komponentů do proměnných.

Po načtení grafických komponentů je třeba určit co se s nimi má stát po nějaké akci. Zejmnéna je třeba nastavit tlačítko a říct mu, co má udelat po kliknutí. Tlačítko se uchovává v proměnné *Scan*. Po nastavení naslouchání tlačítka je lepší spustit informaci, že funkce probíhá ve formě načítacího točícího se kolečka. WiFiManger, zde slouží ke zjištění stavu WiFi sítě, zda jsme k ní připojeni, a zjištění IP adresy. Pokud uživatel není připojen k WiFi síti je vypsán text, že skenování nemohlo být provedeno, protože uživatel není připojen k síti.

V případě připojení k WiFi síti je spuštěno skenování, které zjistí počet zařízení na síti a jejich IP a MAC adresy. Zjištěné hodnoty jsou zapsané do proměnných *list*, obsahující IP a mac adresy zařízení, a *found*, která obsahuje počet nalezených zařízení. Samotné skenování se nachází ve funkci *Scan*, které volá funkci v souboru *Scanner* pro získání hodnot.

```
SCAN.setOnClickListener{
            PROGRESS.visibility = View.VISIBLE WifiManager
            val IP =  Formatter.formatIpAddress(WIFIMANAGER.
    connectionInfo.ipAddress)
            if (WIFIMANAGER.isWifiEnabled) {
                val LIST= scan(IP)
                val FOUND = LIST.size
                RECYCLER_VIEW.adapter = Scanner(LIST)
                RECYCLER_VIEW.layoutManager = LinearLayoutManager(
    this)
                RECYCLER_VIEW.setHasFixedSize(true)

                Snackbar.make(SCAN, "Finished at ${
    getCurrentDateTime()} ", Snackbar.LENGTH_LONG).show()
                DEVICES.setText("Number of found devices: $FOUND")
            }
            else{
                Snackbar.make(SCAN,
                    "WiFi is not enabled, please enable it to
    function",
                    Snackbar.LENGTH_LONG
                ).show()
            }
```

Výpis 2.9: Zdrojový kód stisknutí tlačítka

Funkce *scan* slouží k zavolání funkce skenování a navrácení hodnot ve formě listu IP a MAC adres doplněné o číslo nalezených zařízení.

```
    private fun scan(ip: String): List<Recycler_data_item> {
        val LIST = ArrayList<Recycler_data_item>()
        val SCANNED_LIST = Scanner(LIST).scan(ip)
        return SCANNED_LIST
    }
```

Výpis 2.10: Zdrojový kód ke spuštění skenování

Po dokončení skenování je zavolán *ScackBar*, který slouží ke zjištění informace o konci skenování s časovou známkou, která je získávána pomocí funkce *getCurrentDateTime*. Tato časová známka se zobrazuje na konci po dokončení skenování v dolní části obrazovky.

```
    private fun getCurrentDateTime(): String {
        val DATE = Calendar.getInstance().time
        val FORMATTER = SimpleDateFormat.getDateTimeInstance()
        val FORMATED_DATE = FORMATTER.format(DATE)
        return FORMATED_DATE
    }
```

Výpis 2.11: Zdrojový kód získání času

Recycler data item

Soubor *Recycler data item* obsahuje informaci o tom jaké parametry má obsahovat jeden box v recycler view. Je to vytvoření vlastního typu seznamu. Soubor je potřebný pro vytváření struktur pro vložení do recycler view.

```
data class Recycler_data_item (val imageResource: Int,
                val text1: String, val text2: String)
```

Výpis 2.12: Zdrojový kód souboru recycler data item

Scanner

Další soubor je *scanner.kt*, který obsahuje funkci pro získání IP a MAC adres. Funkce začíná deklarací návratových hodnot, kdy vrací list obsahující IP a MAC adresu všech nalezených zařízení.

```
val recycler_list = ArrayList<Recycler_data_item>()
```

Výpis 2.13: Zdrojový kód návratových proměnných.

Po inicializaci proměnných je možné spustit samotné skenování sítě. Skenování sítě provedeme příkazem *RUntime.getRuntime().exec("příkaz")*, kde spustíme příkaz *ip neighbour show*. Následně počkáme, až se příkaz dokončí a výsledek zapíšeme do proměnné *stdInput*.

```
val proc = Runtime.getRuntime().exec("ip neighbour show")
proc.waitFor()
val stdInput = BufferedReader(InputStreamReader(proc.inputStream))
```

Výpis 2.14: Zdrojový kód zpracování výsledek skenu.

Výsledek skenování načítáme řádek po řádku do proměnné *s*, která se následně rozděluje podle mezer do pole. Výsledné pole, lze považovat za vhodné pouze, pokud má velikost alespoň šesti hodnot. Po splnění podmínky doplníme přičteme do proměnné *found* hodnotu, pro navýšení počtu nalezených zařízení. V poli na prvním místě se nachází IP adresa a na čtvrtém místě se nachází MAC adresa zařízení. Výsledky si zapíšeme do seznamu pro výpis v recycler view. Tímto funkce končí a vrací výsledný seznam do funkce, které ji zavolala.

```
while (stdInput.readLine().also { s = it } != null) {
    (if (s != null) {
        val split = s!!.split(" ")
        if (split.size == 6) {
            found++
            val mac = split[4].toUpperCase()
            val scanned_ip = split[0]
            val item = Recycler_data_item(
                R.drawable.ic_question_mark,
                "IP: $scanned_ip",
                "MAC: $mac"
            )
            recycler_list += item
            }
        }
    )
}
```

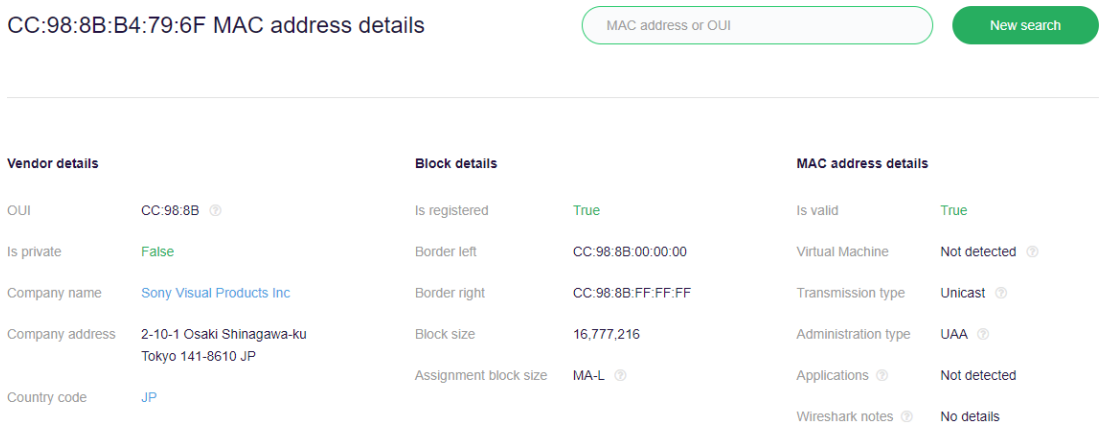
Výpis 2.15: Zdrojový kód spuštění skenování.

2.6.1 Detekce IoT zařízení

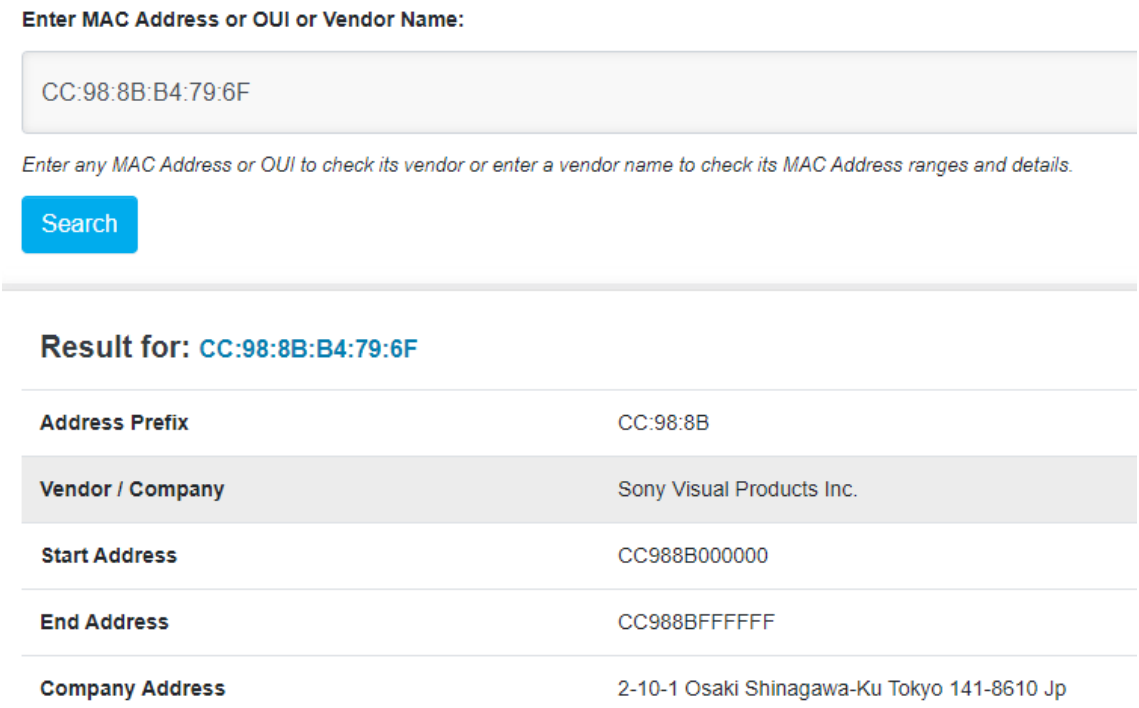
Po zjištění MAC adresy je potřeba zjistit o jaké IoT zařízení se jedná. Zjištěni zda se jedná o IoT zařízení nebo o klasické zařízení je obtížné. Výrobci dostávají přidělené první tři oktety MAC adresy a následně přidělují poslední tři oktety vyrobeným zařízením. Tedy většinou nelze najít přesný model zařízení.

Byla udělána analýza dvou dostupných webových stránek (*macaddress.io* a *dnschecker.org*), které nabízejí možnost zjištění výrobce zařízení pomocí MAC adresy pomocí Application programming interface (API). Obě stránky nabízejí podobné informace, pouze stránka *macaddress.io* nabízí informace navíc o zda je MAC adresa zaregistrována a z jakého rozsahu pochází. Výsledky hledání MAC adresy televizoru Sony jsou vidět na obrázcích 2.7 a 2.8.

Během vlastní implementace byly pokusy o zprovoznění získávání jména výrobce z daných stránek, ale vyskytovaly se chybové hlášky. Ze začátku při pokusu o navázání spojení a později v návratové hodnotě funkce. Části kódu, kterými bylo pokoušeno připojení nejsou součástí kódu. Obě stránky mají nevýhodu, že nabízejí pouze omezený počet žádostí denně, které by nemohlo být použito, protože na každou žádost je potřeba vlastní API klíč. V budoucnu se aplikace bude dotazovat na databázi pro zjištění výrobce zařízení.



Obr. 2.7: Výsledek hledání MAC adresy pomoci *macaddress.io*.



Obr. 2.8: Výsledek hledání MAC adresy pomoci *dnschecker.org*.

2.6.2 Návrhy na rozšíření aplikace

Samotná aplikace je spíše důkazem, že se dané věci zjistit dají a jaké informace lze zjistit. Aplikace bude sloužit jako modul nebo předlohu zjišťování informací o zařízeních na síti, takže věci nejsou plně optimalizované a pro plnohodnotnou aplikaci by bylo potřeba dodělat další změny. První dodělávkou by mohlo být zapisování výsledků do lokální tabulky. Hodnoty z tabulky by se mohly stáhnout pro zkoumání na později nebo naopak nahrát hodnoty z předešlých skenování. Dále by mohla být možnost pojmenovávání zařízeních.

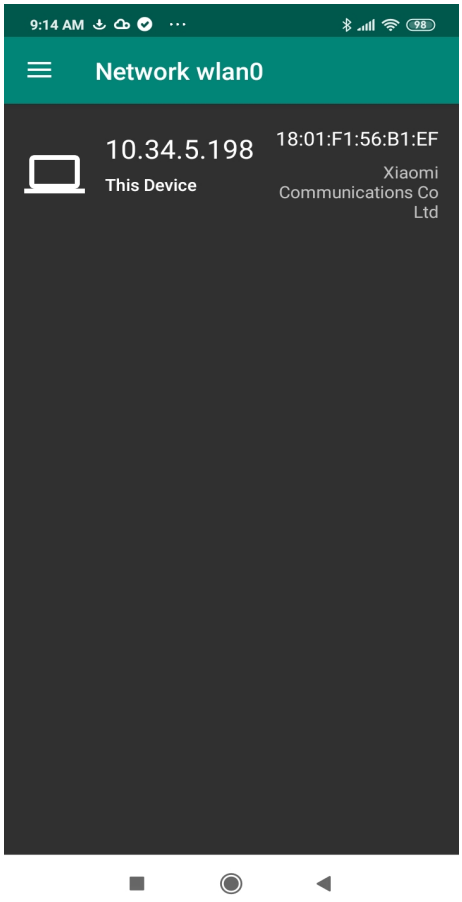
V případě, že na routeru není zapnutá funkce oddělení uživatelů, mohla by aplikace zjišťovat informace ohledně otevřených portů nebo získávání informací o odezvě mezi zařízeními. Popřípadě se dají zjistit i další informace o zařízeních dalšími způsoby, které zjišťují bezpečnostní aplikace pro zjištění otevřených portů a dalších zranitelností.

2.7 Porovnání aplikace Ning s vlastní aplikací

Jediná aplikace, která byla vybrána ke srovnávání, je Ning. Aplikace má otevřený zdrojový kód s licencí umožňující použití v jiných komerčních řešeních. Zároveň aplikace je napsána v jazyce kotlin, která byla podmínkou pro jazyk v jakém má být

aplikace napsaná. Ostatní aplikace s otevřeným zdrojovým kódem jsou napsané v jazyce Java a zároveň u nich nedochází k dalšímu vývoji. Aplikace, které byly nalezeny v obchodě Google play nemají otevřený zdrojový kód a jejich licence neumožňuje využití v komerčních řešeních.

Největším nedostatkem aplikace bylo nalezeno na sítích, které mají zapnutou funkci oddělení uživatelů. To neumožňuje komunikace zařízení komunikaci mezi sebou, funguje pouze komunikace zařízení s výchozí bránou. Tudíž výsledkem skenování bylo nalezení pouze vlastního zařízení.



Obr. 2.9: Výsledek skenování aplikací Ning při zapnuté funkci oddělení uživatelů.

Používané typy detekce zařízení

Pro porovnání aplikací a zjištění důvodu, proč aplikace nedovoluje nalézt zařízení na síti. Bylo potřeba nejdříve zjistit jaké způsoby detekce zařízení aplikace používá. Zjištěné protokoly byly použity k poučení, které se nemají použít pro skenování na sítích se zapnutou funkcí oddělení uživatelů a bylo potřeba vybrat jiné. Použitá Verze aplikace Ning je 1.2.4, vydaná 8.9.2020, v době psaní této práce to byla nejnovější verze aplikace.

Aplikace používá dva způsoby ke zjištění zařízeních na síti. První typ skenování je pomocí příkazu *ping*, konkrétně její variantu v Android knihovně *isReachable()*. Funkce zjišťuje zda je zařízení dostupné a pokud ano, uloží si výsledek pro další zpracování. Zjištěná informace je pouze IP adresa.

Druhý způsob používá *Network service discovery*, kdy aplikace vytvoří službu, kterou hledá nabízené služby na síti. Aplikace umožňuje detekování 13 služeb, protože Android API nepodporuje objevování všech služeb. Po vyčkání na odpovědi od ostatních uživatelů sítě, je opět zjištěna pouze IP adresa u nalezených zařízení.

Po zjištění IP adres zařízení na síti je potřeba zjistit jejich MAC adresa. Její zjišťování probíhá čtením ARP tabulky, která není dostupná na novějších verzích Android zařízení. Druhým způsobem bylo použití příkazu *ip neighbour*, který je používán ve vlastní aplikaci. Bohužel je chybná implementace a nejsou nalezena žádná zařízení na síti se zapnutou funkcí oddělení uživatelů.

Poslední zjištěnou informací je výrobce zařízení. Informace je zjištěna z MAC adresy, pomocí první tří bytů. Hodnota se pak porovnává se seznamem MAC adres výrobců a pokud je nalezena mezi hodnotami, vypíše se výrobce. Aplikace má vlastní databázi výrobců a jejich MAC adres, ve které ji hledá. Tabulka obsahuje 29 228 různých MAC adres pro výrobce, kdy celkový počet výrobců je o menší kvůli duplicitám [28].

Porovnání aplikací

Aplikace byly porovnány na základě základního oskenování, které obsahovalo dobu skenování sítě a nalezených informací. Výsledky hodnot porovnání jsou v tabulce 2.2, u porovnání se řeší zda je zapnuta funkce oddělení uživatelů (CL - Client Isolation) nebo ne, protože to mění výsledky skenování aplikací. Grafické zobrazení výsledků je možně vidět na obrázku 2.7, které jsou vyfoceny při vypnutém oddělení uživatelů na routeru. Pokud je zapnuté je vidět výsledek na obrázku v předchozí kapitole.

Porovnávané hodnoty byly čas doby skenování sítě a zjištěné informace o zařízeních. První věc, která byla zjišťována byla doba skenování. Doba skenování se zjišťovala pomocí deseti měření, kdy výsledná hodnota byla zprůměrována. Výsledek skenování vlastního aplikace vychází na 0,6 sekundy, což je asi 15 sekund rychlejší než celková doba skenování u aplikace Ning. Nutno zmínit, že aplikace Ning zjistila většinu informace o dostupných zařízení za dvě sekundy, ale síť byla nadále skenována a hledány další zařízení. Doba skenování u případu se zapnutou funkcí oddělení uživatelů nebo u vypnuté funkce, byla stejná.

Po zjištění doby skenování je potřeba se podívat na nalezené výsledky viz tabulka 2.2. Obě aplikace zjistily IP a MAC adresy zařízení pokud bylo vypnuté oddělení uživatelů, kdy aplikace Ning zjistila i výrobce zařízení oproti vlastní aplikaci. Naproti

tomu při zapnuté funkci oddělení uživatelů aplikace Ning nezjistila žádné zařízení kromě vlastního. Vlastní aplikace byla schopna zjistit IP a MAC adresy zařízení, pokud tato funkce byla zapnutá. Další informace, která není vidět u aplikace Ning je celkový počet nalezených aplikací, které lze jasně vidět ve vlastní aplikaci.

Tab. 2.2: Porovnání vlastní aplikace a Ning.

	Vlastní aplikace	Ning
Doba skenování[s]	~0,6	~15,7
Celkový počet nalezených zařízení	✓	
Zjištěna IP adresa (CL off)	✓	✓
MAC adresa (CL off)	✓	✓
Výrobce zařízení		✓
Zjištěna IP adresa (CL on)	✓	
MAC adresa (CL on)	✓	

Obr. 2.10: Výsledek skenu pomocí vlastní aplikace.

Obr. 2.11: Výsledek skenu pomocí Ning.

Závěr

Hlavním cílem práce bylo zmapování dostupných skenovacích nástrojů, vytvoření vlastního návrhu a implementace aplikace pro detekování zařízení na síti pro operační systém android zadané od externí firmy. Výsledná aplikace bude sloužit jako modul aplikace externí firmy pro detekování IP a MAC adres, které se můžou podrobit dalšímu zjišťování informací.

Teoretická část mapuje trh dostupných nástrojů pro různé operační systémy. Začíná skenery, které lze ovládat pomocí konzole nebo grafického rozhraní pro operační systémy Windows, Linux a Mac. Pak se zaměřuje na nástroje pro android zařízení a to z obchodu Google play nebo s otevřeným zdrojovým kódem. Následuje popis licencí, které byly použity u zkoumaných nástrojů. Tato část končí shrnutím, kde jsou porovnány protokoly a podporované operační systémy.

Praktická část se dělí na tři části. První část je věnována realizaci experimentálního pracoviště. Následně je pracoviště otestováno a jsou porovnány aplikace dostupné v obchodě Google play. Výsledky měření jsou porovnány a zobrazeny v přehledné tabulce. Z výsledků nevzešel žádný nástroj, který by vyčníval nad ostatní. Druhá část se zaměřuje na vlastní návrh a implementaci aplikace. Byla vytvořena testovací aplikace za účelem ověření propojení fyzického zařízení s vývojovým prostředím. Během implementace vlastního řešení bylo prozkoumáno několik různých přístupů k detekování a identifikaci zařízení na síti, způsoby byly vyzkoušeny a u každé metody byly vypsány výhody a nevýhody každého řešení. Po zhodnocení možných přístupů k řešení detekce bylo vybráno právě jedno řešení do vlastní implementace, a to konkrétně příkaz *ip neighbour show*. Identifikace zařízení je pak řešena pomocí rozpoznání MAC adresy prostřednictvím dostupných služeb na síti internet. Implementace končí prezentací důležitých částí zdrojového kódu.

Závěr praktické části se věnuje srovnání vlastní aplikace s aplikací Ning, která má otevřený zdrojový kód a licenci umožňující využití v komerčních řešeních. Porovnávání probíhalo podobně jako u aplikací z Google play. Bylo provedeno 10 měření, u kterých byla zjištěna doba skenování a zjištěné informace.

Cílem práce bylo zmapování trhu dostupných skenerů pro zařízení na lokální síti, které bylo splněno. U skenerů byl popsán protokol, licence a informace zjištěné o skenovaných zařízeních. Následně byla navrhnuta a implementována vlastní aplikace. Vlastní implementace byla porovnána s aplikací Ning, která patří mezi aktivně vyvíjené aplikace potenciálně splňující komerční použití. Avšak Ning obsahuje zásadní nedostatek při skenování sítí se zapnutou funkcí izolace klientských stanic. V takovém případě nezjistí žádné jiné zařízení, kromě svého. Tento nedostatek byl ve vlastní aplikaci odstraněn. Nakonec jsem se věnoval problematice identifikace zařízení IoT přítomných na lokální síti. Všechny cíle bakalářské práce byly splněny.

Literatura

[1] Galov N. *How Many IoT Devices Are There in 2021? [All You Need To Know]* [online]. [cit.3̇1. 01. 2021]. Dostupné z URL: <https://techjury.net/blog/how-many-iot-devices-are-there>.

[2] Maayan G. *The IoT Rundown For 2020: Stats, Risks, and Solutions* [online]. [cit.3̇1. 01. 2021]. Dostupné z URL: <https://securitytoday.com/Articles/2020/01/13/The-IoT-Rundown-for-2020.aspx>.

[3] Arnold D. *What is a Network Scanner? - Definition & Use* [online]. [cit.3̇1. 01. 2021]. Dostupné z URL: <https://study.com/academy/lesson/what-is-a-network-scanner-definition-use.html>.

[4] Greig J. *DMitry* [online]. [cit.3̇1. 01. 2021]. Dostupné z URL: <https://mor-pah.net/software/dmitry-deepmagic-information-gathering-tool>.

[5] Graham R. *MASSCAN: Mass IP port scanner* [online]. [cit.0̇1. 02. 2021]. Dostupné z URL: <https://github.com/robertdavidgraham/masscan>.

[6] Sullo Ch. *Nikto2 official website* [online]. [cit.0̇6. 02. 2021]. Dostupné z URL: <https://cirt.net/Nikto2>.

[7] *Netcat* [online]. [cit.2̇2. 02. 2021]. Dostupné z URL: <https://sectools.org/tool/netcat>.

[8] Lyon G. *Nmap the Network Mapper - Free Security Scanner* [online]. [cit.3̇1. 01. 2021]. Dostupné z URL: <https://nmap.org>.

[9] Greenbone Networks *OpenVAS - Open Vulnerability Assessment Scanner* [online]. [cit.0̇9. 02. 2021]. Dostupné z URL: <https://www.openvas.org>.

[10] *Sn0int - semi-automatic OSINT framework* [online]. [cit.1̇0. 02. 2021]. Dostupné z URL: <https://github.com/kpcyrd/sn0int>.

[11] Biondi P. *Scapy - Packet crafting for Python2 and Python3* [online]. [cit.13 02. 2021]. Dostupné z URL: <https://scapy.net>.

[12] *ZMap: The Internet Scanner* [online]. [cit.14. 02. 2021]. Dostupné z URL: <https://github.com/zmap/zmap>.

[13] *Angry IP Scanner Fast and friendly network scanner* [online]. [cit.19. 02. 2021]. Dostupné z URL: <https://angryip.org>.

[14] Tiwari A. *Archery a security tool* [online]. [cit.23. 02. 2021]. Dostupné z URL: <https://www.archerysec.com>.

[15] Ponnelle D. *Dipisoft* [online]. [cit.23. 02. 2021]. Dostupné z URL: <https://www.dipisoft.com>.

[16] *Nessus documentation* [online]. [cit.23. 02. 2021]. Dostupné z URL: <https://www.tenable.com/products/nessus>.

[17] *Efficient Network Monitoring With NetCrunch* [online]. [cit.23. 02. 2021]. Dostupné z URL: <https://www.adremsoft.com>.

[18] *NmapSi4* [online]. [cit.23. 02. 2021]. Dostupné z URL: <https://github.com/nmapsi4/nmapsi4>.

[19] *SolarWinds: Port Scanner* [online]. [cit.23. 02. 2021]. Dostupné z URL: <https://www.solarwinds.com/free-tools/port-scanner>.

[20] *Rumble documentation* [online]. [cit.23. 02. 2021]. Dostupné z URL: <https://www.rumble.run/docs>.

[21] Marques A. *Umit Network Scanner* [online]. [cit.23. 02. 2021]. Dostupné z URL: <https://github.com/umitproject/network-scanner>.

[22] *Network scanner by Easy Mobile* [online]. [cit.17. 03. 2021]. Dostupné z URL: <https://play.google.com/store/apps/details?id=com.easymobile.lan.scanner>.

[23] *Network scanner by First Row* [online]. [cit.17. 03. 2021]. Dostupné z URL: <https://play.google.com/store/apps/details?id=com.myprog.netscan>.

[24] *Fing app* [online]. [cit.İ7. 03. 2021]. Dostupné z URL: <https://www.fing.com/products/fing-app>.

[25] *WHO'S ON MY WIFI - NETWORK SCANNER* [online]. [cit.İ7. 03. 2021]. Dostupné z URL: <https://play.google.com/store/apps/details?id=com.magdalm.wifinetworkscanner>.

[26] *WiFiman* [online]. [cit.İ7. 03. 2021]. Dostupné z URL: <https://play.google.com/store/apps/details?id=com.ubnt.usurvey>.

[27] *WiFi Monitor: analyzer of WiFi networks* [online]. [cit.İ7. 03. 2021]. Dostupné z URL: <https://signalmonitoring.com/en/wifi-monitoring-description>.

[28] Csiky C. *Ning - Scan local network for active devices.* [online]. [cit.Ż6. 04. 2021]. Dostupné z URL: <https://github.com/csicar/Ning>.

[29] Rollings M. *Android Network Tools.* [online]. [cit.Ż6. 04. 2021]. Dostupné z URL: <https://github.com/stealthcopter/AndroidNetworkTools>.

[30] Jean-Baptiste A. *Network Discovery.* [online]. [cit.Ż6. 04. 2021]. Dostupné z URL: <https://github.com/rorist/android-network-discovery>.

[31] Berman D. *What Is a Software License? 5 Types of Software Licenses You Need to Know About* [online]. [cit.Ů6. 03. 2021]. Dostupné z URL: <https://snyk.io/learn/what-is-a-software-license>.

[32] *GNU Operating system* [online]. [cit.Ů7. 03. 2021]. Dostupné z URL: <https://www.gnu.org/licenses>.

[33] Assay Matt *Don't believe the hype, AGPL open source licensing is toxic and unpopular* [online]. [cit.Ů8. 03. 2021]. Dostupné z URL: <https://www.techrepublic.com/blog/10-things/dont-believe-the-hype-agpl-open-source-licensing-is-toxic-and-unpopular>.

[34] *APACHE LICENSE, VERSION 2.0* [online]. [cit.Ů8. 03. 2021]. Dostupné z URL: <https://www.apache.org/licenses/LICENSE-2.0>.

[35] Rankin T. *What You Need to Know About Open-Source and Proprietary Licenses* [online]. [cit.Ů8. 03. 2021]. Dostupné z URL:

<https://www.maketecheasier.com/open-source-vs-proprietary-license>.

[36] *InetAddress | Android Developers* [online]. [cit.ů8. 05. 2021]. Dostupné z URL: <https://developer.android.com/reference/java/net/InetAddress>.

[37] Hoffman, Ch *Lock Down Your Wi-Fi Network With Your Router's Wireless Isolation Option* [online]. [cit.í2. 05. 2021]. Dostupné z URL: <https://www.howtogeek.com/179089/lock-down-your-wi-fi-network-with-your-routers-wireless-isolation-option>.

[38] *What is Address Resolution Protocol (ARP)?* [online]. [cit.ů8. 05. 2021]. Dostupné z URL: <https://www.fortinet.com/resources/cyberglossary/what-is-arp>.

[39] *Privacy changes in Android 10* [online]. [cit.ů8. 05. 2021]. Dostupné z URL: <https://developer.android.com/about/versions/10/privacy/changes>.

[40] *Use network service discovery* [online]. [cit.í3. 05. 2021]. Dostupné z URL: <https://developer.android.com/training/connect-devices-wirelessly/nsd>.

[41] Cheshire s. *NS Service Discovery (DNS-SD)* [online]. [cit.í3. 05. 2021]. Dostupné z URL: <http://www.dns-sd.org>.

Seznam symbolů a zkratek

API Application programming interface

EULA End user licence agreement

FTP File transfer protocol

GPL General public licence

HTTP Hypertext transfer protocol

ICMP Internet control message protocol

IMAP4 Internet message access protocol

IoT Internet of things

IP Internet protocol

IT Information technology

MAC Medium Access Control

MIT Massachusetts institute of technology

RAM Random access memory

POP3 Post office protocol

RDP Remote desktop protocol

SCTP Stream control transmission protocol

SMB Server message block

SMTP Simple mail transfer protocol

SNMP Simple network management protocol

SSH Secure shell

SSL Secure sockets layer

TCP Transmission control protocol

TCP Transmission control protocol/internet protocol

Telnet Teletype network

UDP User Datagram Protocol

VNC Virtual network computing

Seznam příloh

A Přiložené soubory **67**

B Zdrojové kódy grafického rozhraní **68**

C Zprovoznění aplikace **71**

C.1 Instalace aplikace pomocí APK 71

C.2 Zprovoznění aplikace v android studiu 71

C.3 Ukázka aplikace bez instalace 71

A Přiložené soubory

- ProchazkaAplikace.zip - obsahuje zdrojový kód aplikace.
- ProchazkaAplikace.apk - obsahuje vygenerovanou aplikaci pro instalaci.
- ProchazkaAplikace.mp4 - obsahuje ukázkové video spuštěné aplikace.

B Zdrojové kódy grafického rozhraní

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:id="@+id/linearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="60sp"
        android:text="Klikli jste" />

    <TextView
        android:id="@+id/txtCounter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textAppearance="@style/TextAppearance.AppCompat.
    Display2"
        android:textColor="#000"
        android:textSize="80sp" />

    <Button
        android:id="@+id/btnIncrementByOne"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Klik" />
</LinearLayout>
```

Výpis B.1: Zdrojový kód grafického rozhraní testovací aplikace (activity_main.xml).

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:layout_width="match_parent" android:layout_height="
    match_parent">

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:visibility="gone" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/purple_500"
            android:orientation="horizontal">

            <Button
                android:id="@+id/scan"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/scan" />

            <TextView
                android:id="@+id/textView"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:gravity="center_horizontal|center_vertical"
                android:textSize="18sp"
                android:textColor="@color/white"
                android:text="" />

        </LinearLayout>


        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recycler_view"
            android:clipToPadding="false"
            android:padding="4dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </LinearLayout>
</RelativeLayout>
```

Výpis B.2: Zdrojový kód grafického rozhraní aplikace (activity_main.xml).

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.
    android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_margin="4dp">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="8dp">
        <ImageView
            android:id="@+id/image_view"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_marginEnd="8dp"
            android:src="@drawable/ic_android" />
        <TextView
            android:id="@+id/text_view_1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toEndOf="@id/image_view"
            android:text="Line 1"
            android:textColor="@android:color/black"
            android:textSize="18sp"
            android:textStyle="bold" />
        <TextView
            android:id="@+id/text_view_2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/text_view_1"
            android:layout_toEndOf="@id/image_view"
            android:text="Line 2" />
    </RelativeLayout>
</androidx.cardview.widget.CardView>
```

Výpis B.3: Zdrojový kód grafického rozhraní aplikace pro recyclerView.

C Zprovoznění aplikace

C.1 Instalace aplikace pomocí APK

Zprovoznit aplikaci na fyzickém zařízení lze dosáhnout pomocí následujícího návodu:

1. Stáhnout přiložený APK soubor (ProchazkaAplikace.apk).
2. Nalézt stáhnutý soubor na mobilním zařízení s operačním systémem android.
3. Kliknout na APK soubor.
4. Aplikace se možná zeptá, zda má instalovat aplikaci z neznámého zdroje, to povolíme a dáme instalovat.
5. Chvíli počkáme a aplikaci můžeme spustit pomocí nově vytvořené ikonky s názvem Network Scanner Module.

C.2 Zprovoznění aplikace v android studiu

Tento způsob zprovoznění nedoporučuji. Aplikace nefunguje správně na virtuálním telefonu a nalezne pouze vlastní zařízení. Veškeré testování bylo uskutečněno na fyzickém zařízení. Aplikace nachází zařízení pouze v síti vytvořené android studiem, kde žádné jiné zařízení není.

1. Stáhnout přiložený ZIP soubor (ProchazkaAplikace.zip).
2. Rozbalit soubor.
3. V Android studiu kliknout na *File* a *open*.
4. Nalézt rozbalenou aplikaci a kliknout na *OK*.
5. Chvíli počkáme a android studio nainstaluje potřebné balíčky pro fungování aplikace.

C.3 Ukázka aplikace bez instalace

V příloze této práce se nachází video, které ukazuje jak aplikace funguje a jaké výsledky lze získat. Jméno videa *ProchazkaAplikace.mp4*