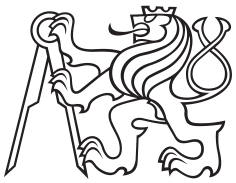


České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Mobilní aplikace pro ovládání IDS/IPS bezpečnostní sondy SonIoT

Diplomová práce

Bc. Michaela Jurková

Vedoucí práce: Ing. Bc. Marek Neruda, Ph.D.
Studijní program: Inteligentní budovy
Květen 2020

Poděkování

Děkuji vedoucímu práce Ing. Bc. Markovi Nerudovi, Ph.D za vstřícný přístup při tvorbě diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května 2020

.....

Abstrakt

Diplomová práce se zabývá vývojem a testováním mobilní aplikace na platformě Android sloužící pro správu IDS/IPS bezpečnostní sondy SonIoT. Aplikace umožňuje zabezpečenou komunikaci se sondou pomocí SSH, včetně autentizace s použitím certifikátu, automatickou aktualizaci bezpečnostních incidentů ze sondy, jejich zobrazení a filtrování, zjištění a zobrazení zařízení v síti a poskytuje uživatelské rozhraní pro správu blokace webových stránek sondou.

Klíčová slova: Android, Java, mobilní aplikace, SonIoT

Vedoucí práce: Ing. Bc. Marek Neruda, Ph.D.

Abstract

The diploma thesis describes the development and testing of the Android mobile application for the SonIoT IDS/IPS security probe management. The application provides a secure communication with the probe via SSH, including a certificate authentication, automatic update of the security incidents from the probe and its displaying and filtering, it facilitates a network scanning for displaying the devices inside the network and provides an interface for the management of blocked websites by the probe.

Keywords: Android, Java, mobile application, SonIoT

Title translation: Mobile Application to Control IDS/IPS Security Probe SonIoT — Diploma thesis

Obsah

Obrázky

Tabulky

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jurková** Jméno: **Michaela** Osobní číslo: **435004**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Inteligentní budovy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Mobilní aplikace pro ovládání IDS/IPS bezpečnostní sondy SonIoT

Název diplomové práce anglicky:

Mobile Application to Control IDS/IPS Security Probe SonIoT

Pokyny pro vypracování:

V návaznosti na předchozí projekty rozšiřte mobilní aplikace pro ovládání IDS/IPS (Intrusion Detection/Prevention System) bezpečnostní sondy SonIoT. Navrhněte, zrealizujte a otestujte mobilní aplikaci v OS Android pro správu IDS/IPS bezpečnostní sondy SonIoT pro verzi Home a Industry.
Navrhněte, zrealizujte a otestujte především tyto funkce: zjištění a zobrazení TCP/IP zařízení, správa blokace webových stránek a jejich kategorií, databáze webových stránek, kategorií a profilů, správa bezpečnostních incidentů včetně databáze a filtrů těchto incidentů, správa sondy SonIoT včetně možnosti připojení pomocí hesla, veřejného klíče a certifikátu.
Provedené testy zdokumentujte, popište případně vzniklý problém, jeho prioritu, návrh řešení a samotné řešení.

Seznam doporučené literatury:

[1] L. Mejzrová a kolektiv, Projekt: Vývoj sondy pro preventivní ochranu IoT zařízení před pokusy o jejich převzetí, interní projektová dokumentace, FEL-ČVUT v Praze
[2] Joshua J. Drake, Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A. Ridley, and Georg Wicherski. 2014. Android Hacker's Handbook (1st. ed.). Wiley Publishing

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Marek Neruda, Ph.D., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **12.02.2020** Termín odevzdání diplomové práce: **22.05.2020**

Platnost zadání diplomové práce:
do konce letního semestru 2020/2021

Ing. Marek Neruda, Ph.D.	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Podpis studentky

Datum převzetí zadání

Kapitola 1

Úvod

Předmětem této diplomové práce je návrh, realizace a testování mobilní aplikace na platformě Android, která slouží jako doplněk pro správu bezpečnostní sondy SonIoT, která je vyvíjena na katedře telekomunikační techniky FEL ČVUT v Praze. Jedná se o systém detekce či prevence průniku, který je vyvíjen ve verzích Industry a Home a přidává úroveň zabezpečení pro firemní i domácí sítě, ve kterých je zapojen. Mobilní aplikace je určena pro správu obou těchto verzí.

Pro použití s nástroji pro monitorování sítě existuje řada webových rozhraní, které uživatelům poskytují přehled o detekovaných síťových hrozbách. Mobilní aplikace SonIoT kromě přehledu bezpečnostních incidentů svým uživatelům poskytuje i další funkce jako je správa blokace webových stránek či skenování zařízení připojených v síti. Všechny tyto funkce jsou v aplikaci rychle přístupné. Jelikož mobilní zařízení uživatelé mohou mít vždy snadno při ruce, hodí se aplikace pro zběžnou kontrolu nových bezpečnostních incidentů či rychlou a snadnou správu blokovaných stránek.

Kromě zobrazení seznamu bezpečnostních incidentů, detekovaných sondou za posledních 30 dní, a detailních informací o jednotlivých incidentech aplikace umožňuje v seznamu těchto incidentů také filtrovat dle zadaných parametrů, omezit počet vypsaných incidentů v seznamu a také vybrat mezi řazením seznamu dle času či závažnosti. Pro správu blokace webových stránek na sondě aplikace uživateli umožňuje jednotlivé webové stránky přiřadit do kategorií a vytvořit profily pro blokaci dle vybraných kategorií, včetně možnosti vybrat pravidelné intervaly, ve kterých mají být webové stránky v blokovaných kategoriích sondou blokovány. Zabezpečená komunikace mezi sondou a aplikací probíhá pomocí protokolu Secure Shell s využitím autentizace pomocí certifikátu.

Diplomová práce je rozdělena na teoretickou a praktickou část. První kapitola teoretické části se zabývá vývojem pro operační systém Android a popisuje jeho základní komponenty, verze, architekturu, způsoby práce s daty a jejich ukládání a zabývá se jeho bezpečností. Druhá kapitola obsahuje informace o bezpečnostní sondě SonIoT, včetně popisu systémů detekce a prevence průniku a popisu nástroje Suricata pro detekci

bezpečnostních hrozeb, který sonda SonIoT využívá. Třetí kapitola teoretické části se zabývá úvodem do kryptografie a popisem protokolu Secure Shell pro zabezpečenou komunikaci mezi zařízeními v síti.

Praktická část je věnována vyvinuté aplikaci. V první kapitole je obsažen popis uživatelského rozhraní aplikace, použitých knihoven a jednotlivých funkcí aplikace včetně jejich implementace. Ve druhé kapitole je pak popsán způsob testování aplikace a jeho výsledky.

Část I

Teoretická část

Kapitola 2

Android

Android je open-source operační systém (OS), který je založený na linuxovém jádře a je využíván širokým spektrem zařízení [1]. Je celosvětově nejrozšířenějším operačním systémem. V dubnu 2020 činil celosvětový tržní podíl zařízení s OS Android přes 39%. Následován byl OS Windows s 33% zařízeními na trhu [2]. Pokud jde o mobilní telefony, podíl OS Android na celosvětovém trhu byl 70,7%, následovaný iOS s 28,8% zařízeními. Podíl ostatních operačních systémů pro mobilní zařízení je zanedbatelný [3]. V České republice je podíl OS Android na mobilních telefonech oproti celosvětovému trhu ještě vyšší a v posledních letech se drží kolem 79%. Téměř 4/5 mobilních telefonů tedy v České republice využívají OS Android [4].

2.1 Základní komponenty

Základní komponenty aplikace na platformě Android jsem popisovala ve své bakalářské práci [5]. Tato kapitola poskytuje krátké shrnutí těch nejdůležitějších.

Nejzákladnější komponentou je aktivita, která představuje jednu obrazovku aplikace. Jednotlivé aktivity jsou na sobě nezávislé a systém je spravuje v zásobníku, na jehož vrcholu je aktuálně zobrazená aktivita. Aktivita od svého vytvoření do svého zániku prochází čtyřmi stavy. Tyto stavy a přechody mezi nimi reprezentují životní cyklus aktivity [5]:

- Při vytvoření aktivity jsou vždy volány metody onCreate(), onStart() a onResume(), kterými se aktivita dostane do aktivního stavu.
- Do pozastaveného stavu se aktivita dostane při volání metody onPause(). Aktivita je v pozastaveném stavu částečně viditelná, ale uživatel s ní nemůže interagovat. Při navrácení se k aktivitě (přechod zpět do aktivního stavu) je opětovně volána metoda onResume().
- V zastaveném stavu je aktivita zcela překryta, ale zůstává v paměti. Do tohoto stavu se dostane z pozastaveného stavu voláním metody onStop().

Při navrácení se k aktivitě ze zastaveného stavu jsou volány metody onRestart(), onStart() a onResume().

- V ukončeném stavu je aktivita odstraněna z paměti. Do tohoto stavu se dostane ze zastaveného stavu zavoláním metody onDestroy(), případně pokud aktivita nebyla aktivní a byla ukončena z důvodu nedostatku paměti. V takovém případě pro navrácení se do aktivního stavu musí být znova vytvořena voláním metod onCreate(), onStart() a onResume().

Součástí aktivity mohou být fragmenty, které představují část jejího uživatelského rozhraní UI (User Interface) či chování. Jeden fragment může být součástí více aktivit.

Další základní komponenty jsou služby, které se používají pro běh aplikace na pozadí, přijímače, které umožňují do aplikace přenášet události ze zařízení, a poskytovatele obsahu, které se používají pro sdílení obsahu mezi aplikacemi [5].

Nedílnou součástí každé aplikace je soubor manifestu obsahující základní informace o aplikaci, které systém potřebuje před jejím spuštěním. Jedná se například o všechna oprávnění, která aplikace vyžaduje pro své fungování, deklaraci minimální podporované úrovně Android API (Application Programming Interface) či deklarace všech komponent aplikace, jako jsou aktivity, služby, přijímače a poskytovatele obsahu [5].

Projekt aplikace obsahuje složku pro zdroje (resources), které jsou přístupné pomocí jejich definovaného ID. Většina zdrojů jsou soubory ve formátu XML (Extensible Markup Language). Jednotlivé zdroje jsou uchovávány v podsložkách animator a anim (animace), color (barvy), drawable (bitmapové soubory či definice tvarů), mipmap (ikona aplikace), layout (definice layoutů UI), menu (definice menu UI), raw (libovolné soubory v surové podobě), values (definice použitých řetězců znaků, barev, stylů apod.) a xml (libovolné XML soubory) [5].

Mezi základní komponenty UI patří textové pole TextView, editační pole EditText, obrázek ImageView, tlačítko Button, přepínač Switch či zaškrtávací pole CheckBox. Tyto prvky mohou být zobrazeny v různých typech layoutů. Často je využíván LinearLayout, ve kterém jsou prvky uspořádány buď v jednom sloupci pod sebou, nebo v jednom řádku vedle sebe, RelativeLayout, ve kterém je umístění prvků dáno na základě definice jejich vzájemných pozic a pozic k rodičovskému layoutu, či FrameLayout, který obsahuje jen jedinou komponentu View. Pro dynamický obsah je možné využít layout, který dědí ze třídy AdapterView. Příkladem je ListView, který vytvoří seznam s položkami, jejichž data jsou k němu vázána pomocí adaptéru, což umožňuje obsah layoutu měnit za běhu aplikace [5].

2.2 Popis novějších verzí

Tato podkapitola představuje některé přínosy verzí OS Android od verze 5.0, Lollipop. Podrobný popis jednotlivých verzí je součástí dokumentace pro Android vývojáře [6].

- ***Lollipop**, verze 5.0 a 5.1 (API 21 a 22)* - verze 5.0 přinesla mnoho nových vlastností a funkcí, poskytla mimo jiné rozšířené nástroje pro snadnou integraci návrhových principů uživatelského rozhraní Material design, vylepšila notifikace, aby byly více viditelné, přístupné a konfigurovatelné, představila podporu pro nové typy senzorů, nástroje pro nahrávání a sdílení obrazovky a nástroje pro optimalizaci využití baterie na základě plánování asynchronních operací [7]
- ***Marshmallow**, verze 6.0 (API 23)* - verze představila automatické zálohování dat aplikací, nové funkce pro autentizaci uživatelů, včetně skenování otisků prstů, nový model oprávnění, kdy jsou uživeteli spravována přímo za chodu aplikace, a také optimalizovala využití baterie představením režimů Doze (režim spánku, s periodickým návratem do normálního režimu pro synchronizaci a vykonání nevyřízených operací) a App Standby (nastává po období, kdy uživatel s aplikací neinteraguje, a zajistí pozastavení synchronizací a operací a zákaz přístupu k síti) [8],[9]
- ***Nougat**, verze 7.0 a 7.1 (API 24 a 25)* - verze 7.0 představila podporu pro zobrazení více oken, přidala Just-in-Time kompilátor pro zlepšení výkonu a zrychlení instalace aplikací a systémových aktualizací, představila režim Doze on the Go (vylepšení k režimu Doze), přidala rozšíření nástrojů pro optimalizaci operací na pozadí a představila nový nástroj pro ověření bezpečnosti použití klíčů z hardwarového úložiště klíčů i pro rootovaná zařízení [10]
- ***Oreo**, verze 8.0 a 8.1 (API 26 a 27)* - verze 8.0 poskytla nová vylepšení notifikací, zjednodušení vyplňování formulářů, vylepšila podporu nestandardních fontů a možnosti optimalizace textu na různých obrazovkách, přidala možnost deklarovat aplikaci v manifestu její kategorii, představila adaptivní ikony, přidala podporu pro správu oken na zařízeních s více displeji a přidala další omezení pro vykonávání operací na pozadí a nástroje pro vypořádání se s těmito omezeními [11], verze 8.1 představila API pro neuronové sítě a optimalizaci využití paměti pro zařízení s malou RAM (Random Access Memory) 1 GB či menší [12]
- ***Pie**, verze 9 (API 28)* - přidala podporu pro Wi-Fi Round-Trip-Time pro sledování polohy v budovách s přesností na 1-2 metry, podporu pro displeje s vykrojením pro fotoaparát či reproduktor a podporu pro využití výstupů z několika fotoaparátů současně, představila další vylepšení pro notifikace, další rozšíření API pro neuronové sítě, přidala nové funkcionality pro zálohu dat a přidala podporu hardwarového bezpečnostního modulu StrongBox Keymaster a přídavné zabezpečení pro dešifrování klíčů v hardwarovém úložišti klíčů [13]
- ***Android 10**, verze 10.0 (API 29)* - přidala možnost použití univerzálních gest pro navigaci na zařízení, změnila namapování spustitelných segmentů systémových binárních souborů a knihoven v paměti tak, že se dají pouze spustit, ale není možné je číst, pro

zabezpečení proti útokům opětovného použití kódu, dále přinesla nová vylepšení kryptografických operací a umožnila uživatelům přepnout jednotlivé aplikace do černobílého barevného režimu a přepnout jednotlivé aplikace do rušivého režimu, ve kterém jsou potlačeny veškeré jejich notifikace [14]

- ***Android 11*** - v době psaní diplomové práce (květen 2020) je tato verze OS Android vydána v první beta verzi [15] a mimo jiné přináší podporu pro displeje přetékající přes hranu zařízení, přidává vizuální indikátory pro 5G, přináší mnoho změn a omezení pro zajištění uživatelského soukromí a dále rozšiřuje API pro neuronové sítě [16]

Podíl jednotlivých verzí na trhu

V tabulce 2.1 je zobrazen podíl jednotlivých verzí OS Android na celosvětovém a českém trhu za duben 2020. Na trhu je momentálně nejrozšířenější Android Pie s 42% na českém a 37% na celosvětovém trhu. Následují verze Android 10 a Android Oreo, které mají obě podíl na českém i celosvětovém trhu mezi 15% a 20%. U starších verzí pak podíl na trhu postupně klesá. Ze statistik za posledních několik měsíců je pak patrné, že podíl Android 10 na trhu postupně sílí, především na úkor Android Pie, ale i ostatních verzí, u kterých ale markantní pokles není patrný [17],[18].

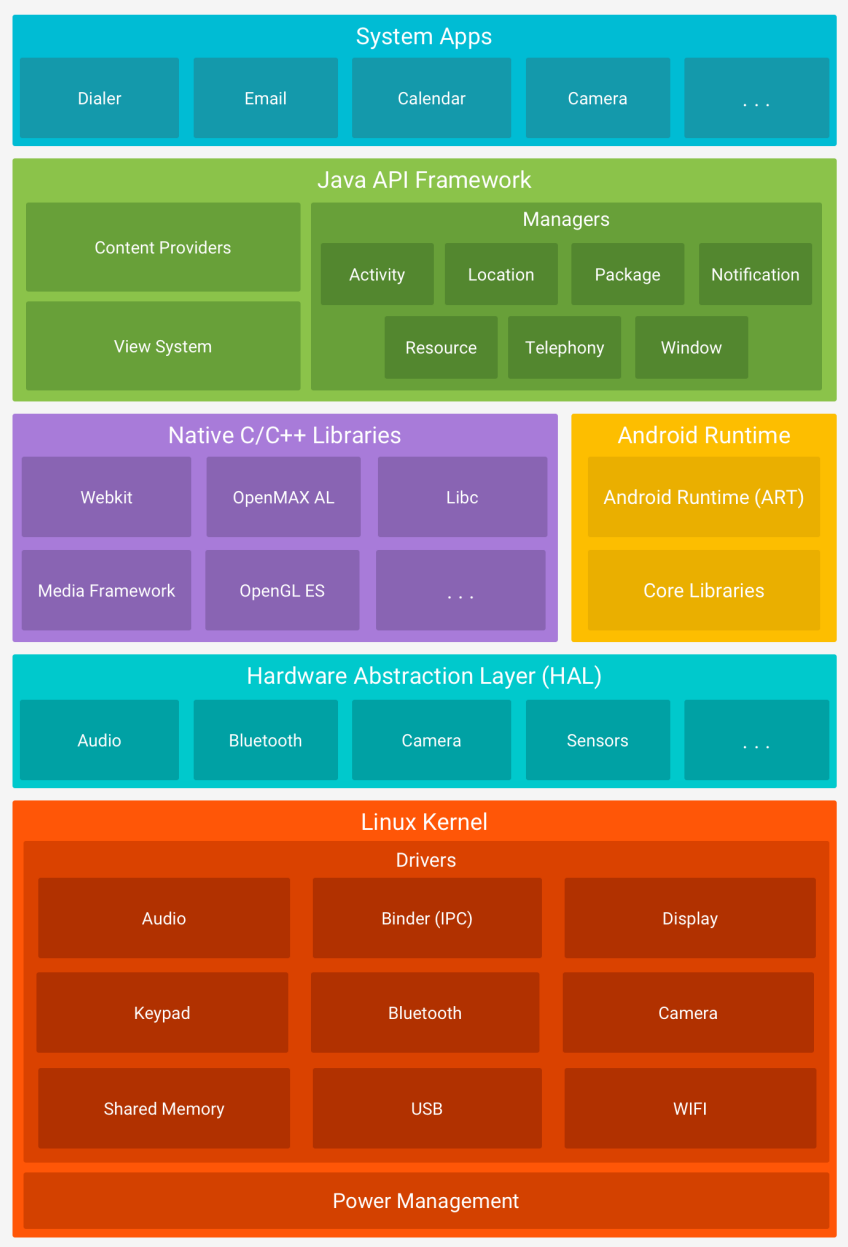
Tabulka 2.1: Porovnání podílu verzí OS Android na celosvětovém a českém trhu v dubnu 2020 dle dat StatCounter [17],[18]

Verze OS Android	Celosvětový trh	Český trh
10.0 10	16,1%	15,0%
9.0 Pie	37,4%	42,0%
8.1 Oreo	11,3%	6,1%
8.0 Oreo	7,37%	10,2%
7.1 Nougat	4,3%	4,0%
7.0 Nougat	6,2%	8,2%
6.0 Marshmallow	8,7%	6,8%
5.1 Lollipop	4,8%	3,9%
5.0 Lollipop	1,21%	1,3%
ostatní	2,5%	2,7%

2.3 Architektura

Softwarový stack platformy Android je znázorněn na obrázku 2.1. Základem je linuxové jádro, nad nímž je hardwarová abstrakční vrstva (HAL = Hardware Abstraction Layer), dále Android Runtime a soubor nativních C/C++ knihoven, Java API framework a na nejvyšší vrstvě jsou systémové

aplikace [1].



Obrázek 2.1: Architektura platformy Android [1]

Linuxové jádro

Na nejnižší vrstvě softwarového stacku platformy Android je upravené linuxové jádro, které spravuje základní systémové funkce, jako je správa procesů, paměti či bezpečnosti, a obsahuje sadu ovladačů pro ovládání hardwarových komponent zařízení (např. displeje, fotoaparátu či Bluetooth modulu). Různé verze OS Android mají různé verze linuxového jádra [19]. Verze 8.0 Oreo představila modulární jádro, které výrobcům umožňuje

pracovat v izolovaných stromech specifických pro zařízení a pozměňovat konfiguraci jádra pro přidání vlastních funkcí a ovladačů ve formě modulů [20].

Hardwarová abstrakční vrstva (HAL)

HAL se skládá z několika modulů, které poskytují rozhraní pro různé typy hardwarových komponent zařízení, a umožňuje tak Java API frameworku na vyšší vrstvě stacku s jednotlivými komponentami pracovat [1]. Například součástí audio modulu mohou být primární, USB (Universal Serial Bus) i Bluetooth audio zařízení a celý HAL modul pak abstrahuje práci s různými audio zařízeními. Výrobci hardwarových komponent tak nemusejí zasahovat do vyšších úrovní systému pro implementování dané funkcionality. Moduly jsou uloženy jako soubory s příponou .so (sdílená knihovna) a dynamicky propojeny systémem, když jsou potřeba [21]. Když si API Java frameworku vyžádá přístup k nějaké hardwarové komponentě, systém načte příslušný modul knihovny [1]. Ve verzi 8.0 Oreo byla nižší vrstva stacku (jádro) pozměněna a zařízení s touto či vyšší verzí musí podporovat HAL napsané v HIDL (HAL Interface Definition Language) [22], který poskytuje nové stabilní rozhraní pro oddělení implementace funkcí hardwarových komponent a Android frameworku. To usnadňuje výrobcům zařízení přechod na novou verzi OS Android, jelikož při aktualizaci systému už nemusejí aktualizovat HAL moduly [23].

Android Runtime (ART)

Runtime (běhové prostředí) je zodpovědný za spouštění a běh aplikací na zařízení. Starší verze OS Android využívaly Dalvik Runtime, kde aplikace běžely v DVM (Dalvik Virtual Machine), který spouštěl Dalvik bytekód, přeložený z Java bytekódu pomocí Dex kompilátoru. Od verze 5.0 Lollipop je Dalvik nahrazen Android Runtime (ART), který spouští přímo Java bytekód [19]. ART je navržen pro optimalizaci využití paměti při běhu více JVM (Java Virtual Machine). Umožňuje kompilaci Ahead-Of-Time (AOT) i Just-In-Time (JIT), poskytuje optimalizovaný garbage collector a přináší vylepšenou podporu debugování. Součástí jsou také základní knihovny běhového prostředí, poskytující funkce programovacího jazyka Java. Každá aplikace běží ve svém vlastním procesu a má vlastní instanci ART [1].

Nativní C/C++ knihovny

Mnoho systémových komponent a služeb jako ART či HAL je v Androidu postaveno na kódu vyžadujícím nativní C/C++ knihovny. Přes API Java frameworku jsou funkce některých těchto knihoven dostupné i aplikacím [1]. Jedná se například o knihovny jako SQLite pro správu dat v databázi či Surface Manager pro správu grafiky [19].

Java API framework

Veškeré funkce OS Android jsou aplikacím poskytovány přes API napsaná v programovacím jazyce Java. To mimo jiné zahrnuje správu aktivit, notifikací, zdrojů či přístup k poskytovatelům obsahu [1].

Systémové aplikace

Na nejvyšší úrovni softwarového stacku platformy Android je soubor aplikací, se kterými uživatel přímo interaguje. Aplikace uživatel může nainstalovat např. z obchodu Google Play a systém obsahuje i řadu předinstalovaných aplikací [19], např. aplikaci pro zasílání zpráv, pro správu kontaktů, kalendář, internetový prohlížeč či klávesnici. Až na výjimky (např. aplikace Nastavení) tyto předinstalované aplikace nemají oproti uživatelsky instalovaným aplikacím žádné speciální postavení a uživatel je může nahradit jinými (např. uživatel nainstaluje nový internetový prohlížeč, kterým nahradí defaultní systémovou aplikaci internetového prohlížeče) [1].

2.4 Práce s daty

2.4.1 Způsoby uchovávání dat

Android obsahuje několik tříd pro ukládání údajů a má implementovánu databázi SQLite [24].

SharedPreferences

Třída SharedPreferences poskytuje jednoduchý, rychlý a efektivní způsob uložení dat, který ovšem umožňuje ukládat pouze primitivní datové typy (boolean, float, long, integer, string) [25]. Data jsou ukládána ve formátu klíč-hodnota, přičemž uvnitř každého SharedPreferences souboru může být uchována kolekce několika párů klíč-hodnota. Soubory umožňují soukromý i sdílený přístup, dle potřeby aplikace [26]. Využívá se k dlouhodobému ukládání dat, například konfiguračních údajů. Ukládaná data jsou perzistentní, jsou tedy dostupná i po ukončení a restartu aplikace [24].

Soubor v interní paměti

Čtení a zápis dat přidružených k interní paměti aplikace. K souborům uloženým v interní paměti může přistupovat pouze daná aplikace. Nejsou přístupné jiným aplikacím, ani uživateli. V případě odinstalování aplikace jsou tyto soubory automaticky smazány [25]. Interní úložiště se používá zpravidla pro ukládání menších objemů dat, dočasných údajů či pro interní potřeby aplikací [24].

Soubor v externí paměti

Ukládání souborů do externí paměti, nejčastěji na paměťovou kartu. Používá se pro ukládání větších objemů dat, například obrázků či hudby. Je možné i využití pro uložení dočasných souborů, které systém automaticky smaže v případě nedostatku místa v úložišti [24]. Uložená data jsou přístupná ostatním aplikacím i uživateli, úložiště tedy není vhodné pro ukládání privátních údajů [25].

Databáze SQLite

SQLite je relační databáze, která pracuje bez serveru a není nutné ji instalovat [24]. Vytvořená SQLite databáze v aplikaci na platformě Android je přístupná ze všech tříd dané aplikace a není přístupná jiným aplikacím [25]. Různé verze OS Android využívají různé verze SQLite. Verze Androidu od 5.0 Lollipop používají verzi SQLite databáze 3.8 nebo novější [27]. Jelikož přístup k databázi zahrnuje přístup do souborového systému, je doporučeno operace s databází provádět asynchronně [28].

Ukládáním dat do SQLite databáze se blíže zabývá následující podkapitola 2.4.2.

Externí databáze

Ukládání dat na vzdáleném serveru mimo vlastní zařízení Android. Vyžaduje internetové připojení. Uložená data jsou získávána přes síť pomocí HTTP (Hypertext Transfer Protocol) požadavků. Data v externí databázi mohou být uchována i po odinstalování aplikace či změně zařízení a je možné v databázi zpracovávat data získaná od různých uživatelů na různých zařízeních [25].

2.4.2 Databáze SQLite

Úvod do relačních databází

Relační model databáze vytvořil v roce 1970 E. F. Codd. Díky jednoduchosti dotazování se stal nejpopulárnějším modelem. Data jsou v něm uchovávána ve formě tabulek (tzv. relací), jejichž řádky reprezentují jednotlivé entity a sloupce jejich atributy. Vztahy mezi tabulkami jsou implementovány pomocí společného atributu [29]. Všechny hodnoty v relační tabulce musí být elementární, tedy dále nedělitelné na další údaje. Sloupce, kterým je přiřazen název (atribut) a ve kterých jsou vždy údaje stejného druhu, mohou být v relační tabulce v libovolném pořadí. Stejně tak nezáleží na pozici řádků, které ale musí být jednoznačně rozlišitelné – nesmí existovat více stejných řádků. Pokud některý z požadavků není splněn, nejedná se o relační tabulku [30].

Atribut, který jednoznačně určuje řádky v tabulce se nazývá tzv. primární klíč. Pokud je k jednoznačnému určení řádků potřeba více sloupců, jedná se

o tzv. složený klíč [30]. Propojení mezi daty ve dvou relačních tabulkách zajišťuje tzv. cizí klíč, který zpravidla odkazuje na primární klíč cizí tabulky. Pokud by tedy například v jedné tabulce byl seznam učitelů a primárním klíčem by bylo unikátní ID učitele, ve druhé tabulce se seznamem vyučovaných předmětů by byl cizím klíčem atribut přiřazující ID učitele k daným předmětům. Pomocí cizích klíčů je možné specifikovat akci pro smazání či aktualizaci odkazované entity, takže je například možné společně se smazáním konkrétního učitele z první tabulky vymazat i všechny předměty vyučované tímto učitelem z tabulky druhé [31].

Relační model umožňuje 3 typy vztahů mezi jednotlivými tabulkami:

- *Vztah 1:1* - řádku z jedné tabulky je přidružen jediný záznam z tabulky druhé a naopak - používá se pro oddělení krátkodobých informací, pro jednodušší vymazání záznamů, dat vztažených jen k podmnožině dat hlavní tabulky, oddělení tabulky z bezpečnostních důvodů či rozdělení tabulky s mnoha sloupci [32]
- *Vztah 1:N* - nejpoužívanější vztah, kdy jeden řádek z první tabulky může mít více odpovídajících řádků v druhé tabulce, nicméně jednomu řádku z druhé tabulky musí odpovídat vždy pouze jediný řádek z první tabulky [32]
- *Vztah N:M* - řádku z první tabulky může odpovídat více řádků z tabulky druhé a naopak - vztah je vytvořen definováním třetí, tzv. vazební tabulky, jejíž řádky obsahují cizí klíče obou tabulek [32]

Relační model přinesl nové pohledy na databázové modelování [33]. Hlavními výhodami relační databáze je jednoduchý popis entit ve formě tabulek, snadná modifikace dat a neprocedurální manipulace s daty [30]. S tabulkami relační model manipuluje prostředky predikátového kalkulu 1. řádu, popřípadě operacemi relační algebry [33] (základními operacemi jsou projekce, selekce, spojení, sjednocení, rozdíl, průnik a kartézský součin [30]), upravené v dotazovací jazyk. Způsob dotazování se zcela liší od prostředků síťového či hierarchického modelu databáze [33].

SQLite

SQLite patří mezi světově nejvíce rozšířené softwarové knihovny, denně jsou na světě používány desítky miliard SQLite databázových souborů [34]. Díky své nezávislosti je velmi výkonná a má široké použití na různých platformách [24]. Celý kód i dokumentace SQLite jsou poskytovány jako volné dílo [35].

Databáze je vhodná pro embedded zařízení a internet věcí, většinu webových stránek, datovou analýzu, archivování souborů či pro tvorbu interních a dočasných databází, ale není vhodná při práci s velkými objemy dat či při potřebě zapisování dat v jednom okamžiku více procesy [36].

SQLite má implementovány téměř všechny běžné vlastnosti SQL (Structured Query Language) jazyka [37]. Podporuje například tvorbu více tabulek, triggerů pro jednotlivé řádky tabulky, funkci indexování či vytváření

pohledů pro čtení [24],[37]. Triggery definují operace, které se vykonají ve chvíli, kdy nastane určitá událost (např. smazání řádku v tabulce). Indexy umožňují vytvořit datovou strukturu, ve které je hodnotám indexovaného atributu přiřazen odkaz na příslušný řádek tabulky, a tím urychlují získávání dat z databáze, jelikož není potřeba prohledávat všechny hodnoty tabulky. Pohledy jsou složeny z SQL dotazů a jsou velmi podobné tabulkám s tím rozdílem, že ale nejsou fyzicky uloženy v databázi. Plní funkci předdefinovaných SQL dotazů a usnadňují práci s komplexními SQL dotazy [38].

Data SQLite databáze jsou uložena v souboru, ke kterému přistupuje přímo klientská aplikace, která tak zastává funkci klienta i serveru současně [24]. Do souboru je vždy zapisována změněná část databáze, nepřepisuje jej celý, takže je zápis vykonán rychleji. Transakce jsou atomické, zápisy do databáze se buď vykonávají celé, nebo vůbec, takže databáze v případě selhání aplikace, systému či baterie není poškozena [34]. V průběhu zápisu jsou informace, potřebné k obnově databáze do původního stavu, uloženy ve druhém souboru, který se nachází ve stejné složce jako soubor databáze. Pokud se nepodaří dokončit zápis, při následujícím otevření dané databáze je přítomnost tohoto druhého souboru detekována a databáze je obnovena do stavu před nepovedeným zápisem [39]. Více procesů může k databázi přistupovat zároveň, ovšem provádět v databázi změny může v jeden okamžik vždy pouze jediný proces. Pokud nějaký proces provádí zápis do databáze, uzamkne soubor databáze na dobu potřebnou pro její aktualizaci a ostatní procesy čekají, než je opět odemčen. Obvykle tato operace trvá několik milisekund [40].

Od verze 3.7.0 SQLite podporuje WAL (Write-Ahead Log) soubory, které používají opačný princip, kdy jsou změny v databázi zapisovány do WAL souboru, zatímco původní soubor databáze je v nezměněné podobě přístupný pro čtení. Tento přístup umožňuje ve většině případů čtení z databáze z několika procesů i při provádění zápisu [41].

Každá SQLite databáze obsahuje tabulku SQLITE_MASTER, která definuje schéma databáze. Obsahuje názvy tabulek a indexů obsažených v databázi, kromě dočasných tabulek a jejich indexů. Z tabulky je možné pouze číst a je automaticky aktualizována při vytvoření či smazání tabulky či indexu v databázi [40].

Soubor databáze se skládá z jedné či několika stejně velkých stránek a jeho maximální velikost může být až cca 140 terabytů. Velikost databází je standardně několik kilobytů až několik gigabytů. Prvních 100 bytů zaplňuje hlavička databázového souboru [39].

Dotazovací jazyk SQL

SQL (Structured Query Language) je standardizovaný programovací jazyk pro správu relačních databází a práci s jejich daty, vyvinutý v 70. letech 20. století. SQL příkazy se dělí do čtyř hlavních kategorií [42]:

- *Manipulace s daty* - získávání dat z databáze a provádění operací s daty [42] (např. SELECT, INSERT, UPDATE, DELETE [43])
- *Definice dat* - definování a úprava struktury databáze [42] (např. CREATE, ALTER, DROP [43])
- *Řízení transakcí* - kontrola správného zpracování transakcí a zajištění vrácení do původního stavu v případě chyby [42] (např. COMMIT, ROLLBACK [43])
- *Bezpečnostní opatření* – správa přístupu k databázi a přiřazování uživatelských rolí a oprávnění [42] (např. GRANT, REVOKE [43]) – tyto příkazy v SQLite implementovány nejsou [37]

SQL dotazy se provádějí pomocí příkazu SELECT, který z databáze vybírá záznamy, kde každý řádek obsahuje stejné sloupce. Příkaz často doplňují následující klauzule [44]:

- *FROM* - specifikuje vstupní data (tabulka, spojení více tabulek či jiný SELECT příkaz), mezi výrazy SELECT a FROM jsou specifikovány názvy sloupců které mají být vybrány, případně znak „*“, kterým jsou vybrány všechny sloupce [44]
- *WHERE* - umožňuje data filtrovat na základě logického výrazu, který je pro každý řádek vyhodnocován, a výstupem jsou pak pouze řádky, pro které je výraz pravdivý [44]
- *GROUP BY* - seskupí záznamy, které mají v určitém sloupci stejné hodnoty - používá se spolu s agregačními funkcemi, které z dané skupiny vyberou jediný záznam (např. funkce max() porovná hodnoty jiného atributu tabulky a vybere z každé skupiny záznam, pro který je hodnota nejvyšší) [44]
- *HAVING* - doplňuje klauzuli GROUP BY a filtruje její výsledky na základě zadaného logického výrazu, ponechány jsou pouze záznamy každé skupiny, pro které je výraz platný [44]
- *ORDER BY* - zajistí seřazení řádků (např. na základě hodnot určitého sloupce), přičemž pomocí klíčových slov ASC a DESC lze specifikovat, zda jsou jako první vráceny hodnoty menší či větší [44]
- *LIMIT* - specifikuje maximální počet řádků, které mají být z tabulky vybrány [44]

Knihovna Room

Knihovna Room je součástí kolekce knihoven Architecture Components [45]. Vytváří abstrakční vrstvu nad SQLite databází a zjednodušuje tak její vytvoření a správu. Defaultně neumožňuje vykonávat dotazy v hlavním vlákně, čímž předchází jeho zatěžování a zhoršení výkonu aplikace. Použitím knihovny je možné vytvořit databázi s využitím menšího množství kódu, jelikož se o často opakující se části kódu postará sama. Navíc ověřuje SQL dotazy v době kompilace, takže nedochází k selhání za běhu aplikace [28].

Hlavními komponentami knihovny jsou databáze, sloužící jako hlavní přístupový bod k datům, entita, reprezentující tabulku v databázi, a Data Access Object (DAO), obsahující metody pro přístup k databázi, namapované na SQL dotazy [45]. Jednotlivé třídy a rozhraní jsou označeny příslušnými anotacemi @Database, @Entity a @Dao [28].

Pro každou entitu je v databázi vytvořena tabulka, v jejíž sloupcích jsou atributy definované v dané entitě, přičemž alespoň jeden z atributů musí být definován jako primární klíč. Room umožňuje nastavit pro primární klíč i automaticky generované ID. Pokud některý z atributů definovaných ve třídě entity nemá být uchován v databázi, označí se anotací @Ignore [47]. Pomocí anotace @Embedded je možné vnořit atributy jedné entity do druhé entity. Tabulka druhé entity pak bude obsahovat i sloupce tabulky první entity [48].

Knihovna Room zakazuje, aby na sebe entity vzájemně odkazovaly, jelikož tak dochází k zatěžování hlavního vlákna aplikace. Je ale možné vytvořit POJO (Plain Old Java Object), obsahující dané entity a napsat dotaz spojující jejich tabulky [49].

Pro přístup k datům používá Room komponenty DAO, které mohou být implementovány buď jako rozhraní, nebo jako abstraktní třída. Uvnitř DAO jsou definovány metody pro práci s databází. Metody mohou být označny anotacemi @Insert pro přidání dat do databáze, @Update pro aktualizaci jedné či více entit porovnáním s primárními klíči entit v databázi, @Delete pro smazání určité jedné či více entit z databáze vyhledáním jejich primárních klíčů a @Query pro vykonání SQL dotazu, který je ověřován v době kompilace a v případě chyby zobrazí chybovou hlášku [50].

Room dále také umožňuje definovat metody pro konverzi datových typů, anotované @TypeConverter. Metody vykonávají převod požadovaných datových typů (např. typ Date) na primitivní datové typy, které je možné v databázi uchovat (např. typ Long), a primitivních datových typů zpět na požadované [49].

Aby se uživatelské rozhraní aktualizovalo automaticky se změnou dat, je možné jako návratový typ metody použít LiveData. Knihovna Room pak vygeneruje potřebný kód pro aktualizaci objektu LiveData [50].

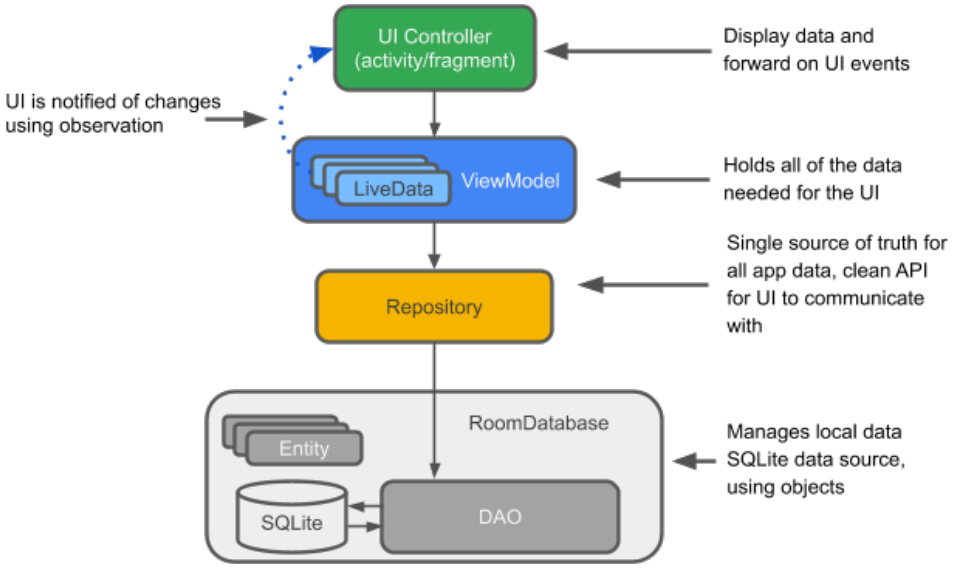
Zachování původních dat při aktualizaci na novější verzi aplikace, která obsahuje změny ve třídách entit, zajišťuje knihovna Room pomocí tříd Migration, které spravují změny mezi dvěma různými verzemi. Pokud vývojář tuto třídu nepoužije, Room databázi vytvoří znovu a dojde ke ztrátě původních dat [51].

2.4.3 Zobrazení dat

Správu životního cyklu komponent uživatelského rozhraní (UI) a zajištění perzistence dat usnadňují knihovny z kolekce Architecture Components, mezi které patří Room, LiveData či ViewModel [52].

Návrh architektury, která využívá knihoven Architecture Components, je znázorněn na obrázku 2.2. Databáze Room používá k přístupu k entitám

v SQLite databázi DAO rozhraní, které poskytuje funkce namapované na SQL dotazy. Třída Repository může spravovat data z více zdrojů (např. z databáze Room a z externí databáze) a poskytuje čisté API pro snadný přístup k těmto datům z jiných částí aplikace. ViewModel zprostředkovává komunikaci mezi Repository a UI, které data zobrazuje. UI komunikuje pouze se třídou ViewModel, takže se už nemusí starat o to, jak data získat. LiveData v sobě vždy drží aktuální verzi dat z databáze a umožňuje notifikovat UI v případě jejich změny. Tím je zajištěno, že UI zobrazuje vždy aktuální data [45].



Obrázek 2.2: Model aplikace využívající Architecture Components [45]

LifecycleOwner a LifecycleObserver

LifecycleOwner je rozhraní, které udává, že třída která jej implementuje, má životní cyklus. Životní cyklus těchto tříd umožňuje pozorovat rozhraní LifecycleObserver. Toto umožňuje jiným třídám aplikace pracovat s ohledem na životní cyklus komponent jako je aktivita či fragment [53].

LiveData

LiveData je objekt držící data jakéhokoliv typu, který respektuje životní cyklus ostatních komponent aplikace (aktivit, fragmentů či služeb). Tyto komponenty mají implementováno rozhraní LifecycleOwner, se kterým je spárována třída Observer (pozorovatel). Při změně dat, která jsou uchována v objektu LiveData, dojde ke spuštění registrovaných pozorovatelů a uživatelské rozhraní je takto automaticky aktualizováno [54].

Příslušný pozorovatel může být automaticky smazán, pokud spárovaný životní cyklus změní svůj stav na DESTROYED. Objekt LiveData aktualizuje vždy pouze pozorovatele, které mají aktivní stav životního cyklu

(tedy stav STARTED či RESUMED). Pokud životní cyklus komponenty přejde do neaktivního stavu, obdrží aktualizaci dat až po opětovném přechodu do aktivního stavu, pokud za tuto dobu došlo ke změně dat. Pokud dojde ke znovuvytvoření komponenty kvůli konfiguračním změnám (např. změna orientace obrazovky), dojde ihned k aktualizaci dat [54].

ViewModel

Třída ViewModel slouží k uchování a správě dat zobrazovaných v UI s ohledem na životní cyklus příslušné aktivity či fragmentu. Tato data jsou často držena uvnitř LiveData objektů. ViewModel také může zajišťovat komunikaci příslušné aktivity či fragmentu se zbytkem aplikace a komunikaci mezi fragmenty v rámci jedné aktivity. Pokud dojde k ukončení a znovuvytvoření aktivity či fragmentu z důvodu konfigurační změny (např. změny orientace obrazovky), nová instance této aktivity či fragmentu bude připojena k existujícímu objektu ViewModel, ve kterém data zůstala zachována. Tímto je zajištěno, že data přežijí konfigurační změny [55].

Paging

Za pomoci knihovny Paging je možné načíst a zobrazit jen určitou malou část dat, kterou UI aktuálně potřebuje. Hlavní komponentou knihovny je třída PagedList, která načítá potřebné části dat a v příděe změny dat vytvoří novou instanci třídy PagedList. Načtená data jsou UI prezentována za pomoci LiveData objektů, které drží PagedList objekty. Knihovna zajišťuje načtení a zobrazení dat, které UI aktuálně vyžaduje, předběžné načtení dalších částí dat (např. nových položek v seznamu, které se uživateli zobrazí při rolování) a animaci změn zobrazovaných dat [56].

2.5 Bezpečnost

Otevřená platforma Android má několik vrstev zabezpečení, které snižují pravděpodobnost útoků na systém a zároveň v případě úspěšného útoku omezují jeho škody. Android poskytuje zabezpečení jednotlivých aplikací, systémových zdrojů a uživatelských dat na zařízení a izoluje aplikace od systému či jiných aplikací. Využívá přitom robustního zabezpečení linuxového jádra, povinného sandboxování aplikací, bezpečné komunikace mezi procesy, podepisování aplikací a správu oprávnění. Každá vrstva architektury OS Android spoléhá na spravné zabezpečení nižších vrstev [57].

OS Android staví svou bezpečnost na zabezpečení linuxového jádra, které je mnoho let široce využíváno v prostředích, které vyžadují vysoké zabezepčení. Linuxové jádro poskytuje Androidu model uživatelských oprávnění, izolaci procesů, rozšiřitelné mechanismy pro zabezpečení komunikace mezi procesy či možnost odstranění nepotřebných či potenciálně nedůvěryhodných částí jádra [58].

Od verze 6.0 Android poskytuje ověření startu zařízení, kdy je při spuštění Android zařízení ověřena integrita a autenticita jednotlivých částí systému. Také je ověřeno, že je spuštěna správná verze OS Android. Tím je zajištěno, že všechen spuštěný kód pochází z věrohodného zdroje a nebyl poškozen. Verze OS Android do 6.0 varovaly uživatele o poškození zařízení, ale dovolily jeho spuštění. Novější verze již v takovém případě neumožní, aby poškozené zařízení bylo spuštěno [59].

2.5.1 Root oprávnění

Uživatelé a aplikace, které mají root oprávnění, mohou provádět změny operačního systému, jádra a jakékoliv aplikace a mají přístup ke všem aplikacím i jejich datům. S root oprávněními v OS Android defaultně běží pouze jeho jádro a malá část systémových aplikací. Uživatelé mohou získat přístup k root oprávnění odemknutím bootloaderu. Pro ochranu uživatelských dat je při jeho odemknutí vyžadováno, aby všechna existující uživatelská data na zařízení byla smazána [60].

2.5.2 Oprávnění využití systémových zdrojů

Některé systémové zdroje, které aplikace využívají, jsou chráněny oprávněními. Pokud chce mít například aplikace přístup k fotoaparátu, GPS (Global Positioning System) datům, Bluetooth či síti, musí mít udělena daná oprávnění, která specifikuje v souboru manifestu [61]. Uživatelé vidí, jaké oprávnění každá aplikace vyžaduje, a mohou tato oprávnění kontrolovat [57]. Ve starších verzích OS Android uživatelé tato oprávnění aplikacím udělovala při jejich instalaci. Ve verzi 6.0 byla přidána možnost udělovat oprávnění aplikacím za jejích běhu a možnost je vypínat a zapínat [62]. Pokud uživatel oprávnění nevypne, je aplikaci oprávnění uděleno do doby, než je odinstalována [61]. Některé zdroje jsou dostupné pouze předinstalovaným aplikacím na zařízení, například přístup k SIM (Subscriber Identity Module) kartě [61].

2.5.3 Aplikační sandbox

Systém Android je navržen tak, aby zabránil zneužití či poškození aplikace jinými aplikacemi a částmi systému [58]. Všechny aplikace na Androidu běží ve vlastním zabezpečeném prostředí (aplikačním sandboxu). Ten mezi sebou jednotlivé aplikace izoluje a ochraňuje tak aplikace i systém před škodlivými aplikacemi [58]. Defaultně mezi sebou aplikace nesmí interagovat a mají jen omezený přístup k operačnímu systému [63]. Každé aplikaci je udělen přístup pouze do části souborového systému, do které daná aplikace může ukládat soukromá data [57]. Aplikace nemají přístup k souborům jiných aplikací, pokud k tomu jejich vývojář nedá přímé svolení [58].

Je ale možné, aby v jednom aplikačním sandboxu běželo více aplikací.

Každá aplikace musí být podepsána vývojářem. Podepsáním je identifikován autor aplikace, čímž je usnadněna i její aktualizace. Při instalaci aplikace je ověřováno, zda byla opravdu podepsána certifikátem, který prezentuje. Pokud jsou aplikace podepsány stejným klíčem, mohou na zařízení, na kterém jsou nainstalovány, bežet ve sdíleném aplikačním sandboxu, pokud si to jejich vývojář přeje [61].

Aplikační sandbox se nachází na úrovni jádra, takže veškerý software ve vrstvách nad jádrem (systémové knihovny, aplikační framework, aplikační běhové prostředí, všechny aplikace) běží v aplikačním sandboxu. Pro porušení aplikačního sandboxu by tak útočník musel poškodit bezpečnost linuxového jádra [63].

Před verzí 4.3 byly aplikační sandboxy definovány vytvořením unikátního linuxového uživatelského ID pro každou aplikaci v době instalace [64]. Každá aplikace běžela jako zvláštní uživatel systému [58]. V novějších verzích je pro posílení tohoto bezpečnostního modelu použit SELinux (Security-Enhanced Linux), který vynucuje povinnou kontrolu přístupu všem procesům, včetně root procesů. Tím je poskytnuto vyšší zabezpečení systémových služeb, kontrola přístupu k datům aplikací a systémovým logům, je omezen vliv škodlivého softwaru a je zajištěna ochrana uživatelů před chybami v kódu. Vše, co v SELinux není explicitně povoleno, je zakázáno [64].

2.5.4 Bezpečnost uložených dat

Celý souborový systém Android zařízení je šifrován a zabezpečen symetrickým šifrovacím klíčem, který je chráněn heslem zařízení. Uživatelská data nejsou cizím uživatelům dostupná ani úpravou bootloaderu či operačního systému [58]. Do verze 7.0 Android šifroval celý disk jediným klíčem. Od verze 7.0 je možné šifrovat jednotlivé soubory různými šifrovacími klíči. Tím je umožněn rychlejší přístup k některým částím dat [65].

Systémová část disku, která obsahuje jádro, systémové knihovny, aplikační běhové prostředí, aplikační framework a systémové aplikace, je nastavena pouze pro čtení. Uživatel může do této části disku tedy manuálně nahrát další aplikace pouze tehdy, pokud zapne Android zařízení v nouzovém režimu [58].

Data, která jsou šifrována pomocí klíčů uložených na zařízení, nejsou chráněna proti přístupu root uživatelů. Vrstva zabezpečení může být přidána použitím klíčů uložených mimo zařízení či chráněných uživatelským heslem. To poskytne dočasnou ochranu, dokud klíč není poskytnut aplikaci, kdy je již dostupný i root uživateli. Robustnější ochrana je možná implementováním hardwarových řešení, které omezují přístup ke specifickým typům dat, například údajům používaným pro platby [58].

2.5.5 Šifrování

Aplikační framework OS Android zahrnuje robustní implementace běžně používaných kryptografických algoritmů pro ochranu dat. Třída Cipher poskytuje implementace algoritmů jako AES (Advanced Encryption Standard) či RSA (Rivest-Shamir-Adleman). Dále poskytuje SecureRandom generátor pro inicializování kryptografických klíčů generovaných pomocí třídy KeyGenerator [66].

Pokud je nutné klíč uschovat pro opakované použití, Android poskytuje systém zvaný Android Keystore, který poskytuje mechanizmus pro dlouhodobou úschovu kryptografických klíčů a znesnadňuje získání těchto klíčů ze zařízení. Android Keystore chrání klíče před neautorizovaným přístupem a znemožňuje extrakci klíčů z aplikačního procesu i celého zařízení, aby nedošlo k jeho neoprávněnému použití. Aplikace používající Android Keystore musí specifikovat autorizované použití daných klíčů. Specifikován je algoritmus, operace či účely jeho použití (šifrování, dešifrování, podepsání, ověření), časový interval, kdy je klíč platný, a další parametry. Android Keystore poskytuje nástroj pro omezení toho, jak a kdy jsou klíče použity, například vyžadováním uživatelské autentizace pro použití klíče, či kontrolu toho, aby byly použity jen ve specifikovaných kryptografických režimech [67].

Klíče jsou v Android Keystore chráněny tím, že samotný klíč nikdy nevstoupí do aplikačního procesu. Pokud aplikace vykonává kryptografickou operaci, která vyžaduje klíč uvnitř Android Keystore, samotná kryptografická operace je uskutečněna uvnitř Android Keystore. I při poškození aplikačního procesu tedy útočník nemůže získat tento klíč. Pokud navíc bezpečnostní hardware zařízení (např. TEE (Trusted Execution Environment) či SE (Secure Element)) podporuje danou kombinaci parametrů klíče, může být klíč vázán na něj a nedochází tak k jeho odhalení mimo tento hardware, tedy ani když útočník získá přístup k internímu úložišti Android zařízení [67].

2.5.6 Řešení bezpečnostních problémů OS Android

Android má vlastní tým lidí, kteří se zabývají bezpečností a vyhledávají potenciální hrozby, navrhují jejich řešení a poskytují pravidelné měsíční aktualizace pro Android zařízení [57]. Tým se zabývá vlastním interním výzkumem potenciálních hrozeb, ale také problémy, které byly nalezeny třetími stranami (např. akademické vědecké práce, reportované problémy či veřejně známé problémy sdílené na blozích a sociálních sítích). Kdokoliv může bezpečnostní tým Androidu upozornit na nalezené bezpečnostní riziko. Bezpečnostní tým následně vyhodnotí, která komponenta platformy Android je rizikem zasažena a přiřadí problému závažnost [60].

Při klasifikaci nalezených problémů jsou rozlišovány lokální a vzdálené útoky. Vzdálené útoky jsou takové, které nepotřebují mít fyzický přístup k zařízení, ani na napadené zařízení instalovat aplikaci. Lokální útoky naopak vyžadují, aby byla na zařízení nainstalována škodlivá aplikace nebo

aby měl útočník přímý přístup k zařízení [60].

Před tím, než je aplikace přijata do Google Play, je skenována pro ověření bezpečnosti. Prostřednictvím služby Google Play jsou také aplikacím předávány důležité bezpečností aktualizace kritických knihoven [57]. Aplikace nahrané v Google Play jsou opakovaně skenovány pro nové bezpečnostní hrozby [68]. Kromě toho, že jsou pro nové bezpečnostní hrozby poskytovány záplaty, Google Play při opakovaném skenování všech aplikací odstraní ty, které se snaží nalezené bezpečnostní hrozby zneužít. Služba Google Play navíc poskytuje i nástroj pro kontrolu aplikací, které byly nainstalovány z jiných zdrojů [60].

Kapitola 3

Bezpečnostní sonda SonIoT

Mobilní aplikace vyvíjená v rámci této diplomové práce slouží jako doplněk pro správu bezpečnostní sondy SonIoT, IDS/IPS (Intrusion Detection System/Intrusion Prevention System) zařízení vyvíjeného na katedře telekomunikační techniky fakulty elektrotechnické ČVUT v Praze. Sonda je k dispozici ve dvou verzích - Industry a Home. Obě tyto verze běží na linuxovém operačním systému a využívají open-source IDS/IPS nástroj Suricata pro detekci a blokaci hrozeb [69].

3.1 IDS/IPS

3.1.1 Funkce

Účelem IDS, tedy systému detekce průniku, je poskytnout sledování, kontrolu, analýzu a hlášení podezřelé aktivity v síti. IPS, tedy systém prevence průniku, k těmto schopnostem přidává možnost detekovanou síťovou hrozbu zastavit a poskytnout tak ochranu vnitřní sítě před vnikem potenciálních hrozeb [71].

Všechny IDS/IPS mají implementovány následující funkce:

- *Zaznamenávání informací o pozorovaných událostech* - obvykle se jedná o lokální záznam, který ale může být poslán dalšímu oddělenému systému, například centralizovanému logovacímu serveru či systému pro správu událostí a bezpečnostních informací [72]
- *Upozornění bezpečnostního administrátora o důležitých detekovaných událostech (tzv. alert)* - notifikace může probíhat např. e-mailem, zprávou přes uživatelské rozhraní systému, syslog zprávou či uživatelskými programy a skripty, přičemž zaslaná zpráva obvykle obsahuje jen základní informace o události [72]
- *Vytváření reportů* - shrnují detekované události a poskytují k nim detailní informace [72]

IPS na detekované hrozby mohou reagovat a pokusit se útokům zabránit. Odpověď na detekované hrozby může mít různou formu. Je možné rozlišit 3 skupiny odpovědí [72]:

- *IPS zastaví samotný útok* - například ukončí síťové spojení nebo uživatelskou relaci použitou pro útok, zablokuje přístup k uživatelskému účtu, IP adrese či jinému atributu útočníka [72]
- *IPS změní bezpečnostní prostředí* - změní konfiguraci dalších bezpečnostních složek, aby útok přerušila - například změní konfiguraci síťového zařízení jako je firewall, router či switch, aby blokovalo provoz od útočníka či směrem k cílovému zařízení [72]
- *IPS změní obsah útoku* - některé IPS technologie umožňují smazat či nahradit škodlivé části útoku tak, aby byl neškodný - například smazat infikovanou přílohu souboru v e-mailu a samotnou zprávu e-mailu propustit [72]

Některé systémy průniku jsou také schopny změnit svůj bezpečnostní profil při detekci nové hrozby, například po detekování konkrétní události může pro následné události upravit jejich důležitost [72].

IDS/IPS systémy nedokáží poskytnout přesnou detekci hrozeb. Část neškodných aktivit je nesprávně detekována jako hrozba a naopak. Jsou tak produkovány falešně pozitivní a falešně negativní události. Výskyt nesprávně detekovaných událostí není možné naprosto eliminovat. Obecně vzato, redukce falešně pozitivních událostí vede k většímu množství falešně negativních a naopak [72]. Je tak potřeba správně nakonfigurovat tzv. šedou zónu, která některé potenciální hrozby propustí [71].

3.1.2 Základní části

Typické komponenty systémů průniku jsou senzory, které monitorují a analyzují aktivitu, server pro správu, který přijímá a spravuje informace ze senzorů, databázový server, na kterém jsou uchovávány informace zaznamenané senzory, a řídicí program, který poskytuje rozhraní pro uživatele a administrátory pro konfiguraci, aktualizaci a monitorování. Tyto komponenty spolu mohou být propojeny přes standardní síť organizace, nebo přes oddělenou síť, vyvinutou speciálně pro správu bezpečnostního softwaru [72].

Jelikož systémy IDS/IPS často obsahují citlivé informace jako je uživatelská konfigurace a identifikované zranitelnosti sítě, které mohou být užitečné pro plánování dalších útoků, je důležité, aby veškeré jejich komponenty byly dostatečně zabezpečeny a byly udržovány aktualizované. Doporučené je také vytvořit oddělené účty pro jednotlivé uživatele a administrátora systému s různým přiřazením oprávnění, omezit přístup ostatních síťových zařízení k IDS/IPS tak, aby jejich přístup byl možný pouze tam, kde je opravdu potřeba, a zajistit dostatečné zabezpečení komunikace pro správu IDS/IPS, buď fyzickým či logickým oddělením, nebo

zašifrováním komunikace. Vhodné je také použití silné autentizace pro vzdálený přístup k IDS/IPS komponentám, použít lze například vícefaktorovou autentizaci [72].

3.1.3 Rozdělení a umístění v síti

IDS/IPS systémy je možné rozdělit dle typu monitorovaných událostí a způsobů jejich nasazení na síťově orientované, které monitorují síťový provoz určité části sítě či zařízení a analyzují aktivitu síťových a aplikačních protokolů, bezdrátové, které monitorují provoz v bezdrátové síti včetně analyzování jejích protokolů, systémy založené na analýze chování v síti, které analyzují síťový provoz pro identifikaci neobvyklých datových toků, a systémy uživatelské, které monitorují aktivity určitého hostitelského zařízení. Pro komplexní zabezpečení sítě je možné kombinovat více typů IDS/IPS systémů [72].

V síťově orientovaných systémech může být umístění senzorů buď tzv. inline, nebo pasivní.

V případě inline umístění síťový provoz prochází přímo přes senzor, což systému umožňuje zastavit útoky. V síti jsou obvykle umístěny za firewallem a přidávají další vrstvu ochrany před vstupem hrozeb do sítě a vylepšují tak schopnost firewallu [71]. Mohou být ale zapojeny i před ním za účelem poskytnutí ochrany a menší zátěže pro jiné zařízení v síti [72]. Inline zapojení je typické pro IPS systémy, které pak zmírňují riziko útoku tím, že potenciálně škodlivé pakety nepropustí dál do sítě. Jejich důležitou vlastností je spolehlivost - schopnost vykonávat stanovené operace za jakýchkoliv podmínek. Řešení tak vyžaduje deterministický přístup, což ale znemožňuje dělat nad pakety složité rozhodování. Požadavkem na senzor je vysoká dostupnost (musí ustát chod v nejnáročnějším síťovém prostředí), vysoký výkon (každý paket musí umět analyzovat bez postřehnutelného rozdílu v průchodu dat - musí mít vysokou propustnost a nízkou latenci) a ovladatelnost a škálovatelnost (musí umožňovat svou správu tak, aby při tom nebyl ovlivněn procházející tok dat) [71].

Pasivní senzory monitorují kopii skutečného síťového provozu a nemohou tedy detekované hrozby zastavit. Typicky jsou použity k monitorování klíčových částí sítě, například pro sledování aktivity v demilitarizované zóně [72]. Pasivní umístění je typické pro IDS systémy. Vzhledem k tomu, že v tomto umístění není neovlivněn chod dat do sítě, je možné na procházejících paketech vykonávat operace, které vyžadují větší výpočetní výkon a jsou zdlouhavější, a využívat několika různých detekčních technik. Je tím umožněna inteligentní analýza paketů procházejících do sítě, která může pro detekci hrozeb využívat i nedeterministické metody, například statistickou analýzu provozu [71].

3.1.4 Detekce událostí

IDS/IPS mohou využívat různé metody detekce událostí, které se mezi sebou dají i kombinovat [72].

Detekce založená na signaturách

Signaturou se rozumí vzor, který odpovídá známé hrozbě. Metoda porovnává signatury oproti procházejícím datům za účelem identifikovat potenciálních hrozby. Metoda je velmi efektivní v detekování známých hrozeb, ale neumí si dostatečně dobře poradit s novými hrozbami a novými variantami známých hrozeb. Neodhalí ani většinu útoků, které sestávají z více aktivit. Jedná se o nejjednodušší metodu, která porovnává aktuální aktivitu oproti seznamu signatur pomocí textových řetězců [72].

Detekce založená na anomáliích

Metoda porovnává definice aktivit, které jsou považovány za obvyklé (tzv. profily reprezentující normální chování), s aktuální aktivitou za účelem identifikace významných odchylek v chování. Profily jsou vytvářeny na základě monitorování typické aktivity v síti za určitý čas. Porovnávání s těmito profily probíhá na základě statistických metod. Na rozdíl od detekce založené na signaturách má tento způsob schopnost detekovat i dosud nepoznané hrozby. Problém v použití této metody spočívá v komplikovaném vytváření přesných profilů, především v riziku neúmyslného zahrnutí škodlivé aktivity do vytvořeného profilu jako normální aktivitu, ale i v nezahrnutí běžné aktivity, která ale probíhá pouze jednou za čas [72].

Detekce založená na stavové analýze protokolů

Metoda zjišťuje odchylky aktuální aktivity oproti profilům obecně přijatých definic neškodných aktivit protokolů pro všechny jejich stavy. Nevyužívá specifické profily monitorované sítě jako metoda založená na anomáliích, ale univerzálně vytvořené profily specifikující použití konkrétních protokolů. Porovnávání probíhá s ohledem na stav sítě, transportu a protokolu. Metoda dokáže identifikovat neočekávané sekvence příkazů. Nevýhodou metody je vysoká složitost analýzy a nutnost sledovat stav pro mnoho simultánních relací. Analýza navíc nedokáže odhalit útoky, které neporušují standardní chování protokolů. Nesprávnou detekci může zapříčinit také nekonzistence modelu protokolu v jeho různých verzích pro různé aplikace a operační systémy [72].

3.2 Suricata

3.2.1 Nástroj Suricata

Suricata je open-source nástroj pro detekci síťových hrozeb, který umožňuje fungovat v režimech IDS i IPS. Je vyvíjen neziskovou organizací Open Information Security Foundation (OISF) [73], která je tvořena mezinárodní skupinou vývojářů a konsorciem předních firem z oblasti kyberbezpečnosti [74]. Jedním z partnerů je například bezpečnostní autorita Agence Nationale de la Securité des Systémes d'Information (ANSSI) [75]. Cílem je vytvořit IDS/IPS engine nové generace [74].

Suricata podporuje operační systémy Linux, FreeBSD, OpenBSD, macOS i Windows [76]. Hlavní výhodou Suricaty oproti konkurenčnímu nástroji Snort, který vlastní korporace Cisco a je taktéž open-source [75], je její plná podpora vícevláknového režimu, která umožňuje zpracování mnohem většího objemu dat, a schopnost pravidla jednoduše aplikovat na protokoly aplikační vrstvy použitím klíčového slova, které specifikuje daný protokol. Dokáže pak tyto protokoly snadno rozpoznat, i když jejich komunikace neprobíhá na obvyklých portech [77].

Nedílnou součástí Suricaty jsou pravidla, která pomáhají odhalit síťové hrozby [78]. Procházející pakety jsou porovnávány s předem nastavenými pravidly a v případě shody jsou detekovány a zaznamenány události jako je neobvyklá komunikace, podezřelá aktivita či pokus o útok. Správné nastavení těchto pravidel je tedy klíčové pro to, aby nástroj dokázal potenciální hrozby správně detekovat. Je možné využít existující sady pravidel, například sady Emerging Threats či pravidla Snortu, včetně možnosti předplatit si zpoplatněné sady pravidel, ale je možné i manuálně vytvořit vlastní pravidla. Jednotlivé zdroje pravidel je pak možné spolu libovolně kombinovat [75]. Pro správnou funkci je nutné, aby pravidla byla pravidelně aktualizována. Pro usnadnění aktualizace těchto pravidel poskytuje Suricata nástroj Suricata-Update [79].

Pravidla sestávají ze tří částí - akce, hlavičky a vlastností pravidla. Akce udává, jaká operace se má provést s paketem, u kterého je detekována shoda s daným pravidlem. Hlavička obsahuje protokol, zdrojové a cílové IP adresy a porty a směr pravidla. Vlastnosti pravidlo blíže specifikují. Nastavení jednotlivých vlastností probíhá na základě použití klíčových slov. Jednotlivé vlastnosti jsou pak obaleny v závorkách a odděleny středníkem [78]. Při porovnávání procházejících paketů oproti načteným pravidlům Suricata vytváří interní skupiny pravidel, aby jej porovnávala pouze oproti takovým pravidlům, u kterých to má smysl. Takže pokud například procházející paket využívá protokol TCP (Transmission Control Protocol), neporovnává jej oproti pravidlům pro UDP (User Datagram Protocol) protokol. Suricata nicméně vytváří jen omezené množství těchto skupin, aby zbytečně nezvyšovala nároky na paměť. Při vytváření skupin pravidel se řídí tím, aby byly vyváženy nároky na paměť a výkon [80].

Účelem některých pravidel může být pouze poskytnout informaci, jiná

varují před vážným bezpečnostním rizikem. Pro rozlišení těchto pravidel je součástí Suricaty klasifikační soubor, ve kterém je pro pravidla nastavený název, popis a priorita s hodnotou od 1 do 4, kde pravidla s prioritou 1 jsou nejdůležitější a s prioritou 4 nejméně důležitá. Tyto informace pak obsahuje i log události při nalezení shody s konkrétním pravidlem. [80].

Výstupy Suricaty jsou uchovávány ve složce /var/log/suricata, kde je možné nalézt logy s detekovanými událostmi [80]. Nejdůležitějším výstupním souborem Suricaty je log Eve ve formátu JSON (JavaScript Object Notation). Obsahuje upozornění (vytvořené záznamy událostí při shodě pravidel), anomálie (detekované pakety s neočekávanými hodnotami), metadata, informace o souboru a specifické záznamy protokolů [81]. Pro práci s jeho daty je vhodné použít nástroj jq, pomocí kterého je například možné vybrat z logu pouze určitý typ detekovaných hrozeb nebo zjistit počet paketů, které byly Suricatou zpracovány [79].

3.2.2 Grafická rozhraní pro analýzu událostí

Pro Suricatu je k dispozici několik různých nástrojů poskytujících webový front-end pro analýzu detekovaných událostí v grafickém rozhraní. Mezi nejznámější z nich se řadí Snorby [82], BASE (Basic Analysis and Security Engine) [83], Squert [84] či EveBox [85].

Všechny tyto aplikace umožňují zobrazit přehled detekovaných událostí. Je možné zobrazit všechny detekované události, nebo pouze nejnovější události za určitý časový úsek. Zobrazované informace o událostech v přehledu se u každé aplikace liší, ale všechny obsahují časovou značku, vstupní a výstupní adresu a signaturu události. Taktéž umožňují mezi událostmi filtrovat dle jejich specifických vlastností, například zobrazit jen události s konkrétní signaturou. Aplikace dále umožňují analyzovat vybrané události a zobrazit informace o daném paketu a pravidle, na základě něhož byla událost generována. Nejmodernější grafické uživatelské rozhraní ze zmíněných aplikací poskytuje Snorby, které události barevně rozděluje dle jejich závažnosti a vytváří přehledné grafy, zobrazující statistiku vzniklých událostí za vybraný časový úsek, seskupených dle jejich specifických vlastností [86].

Pro analýzu událostí je pro bezpečnostní sondu SonIoT navíc možné použít mobilní aplikaci vyvíjenou v rámci této diplomové práce, která mimo jiné obsahuje databázi detekovaných upozornění (událostí typu alert) za posledních 30 dní a umožňuje jejich filtraci. Aplikace je blíže popsána v kapitole Mobilní aplikace 5.

3.3 Sonda SonIoT

Projektová dokumentace k bezpečnostní sondě SonIoT [69] rozlišuje 2 verze sondy - Industry a Home. Pro verzi Industry je zařízení postaveno na embedded platformě Up Board s OS Ubuntu. Verze Home je postavena na embedded platformě Raspberry Pi s OS Raspbian Buster. Obě verze

využívají nástroj Suricata ve verzi 4.1.4, která je automaticky spuštěna při každém spuštění sondy. Zároveň se při startu vždy provádí také obnovení nastavení iptables [87], linuxového nástroje pro nastavení operací pro procházející IP datagramy.

Sonda podporuje funkci v režimech IDS (detekuje hrozby v síti) i IPS (automaticky blokuje neobvyklou aktivitu v síti). Mezi těmito režimy je možné přepínat. Od toho se pak odvíjí i její umístění v síti. V IPS režimu, kdy komunikace ze vstupního síťového portu směřuje přes Suricatu do výstupního síťového portu, je předávání provozu mezi porty závislé na nastavení iptables. V IDS režimu je na sondu síťový provoz pouze přeposílán pomocí síťového přepínače a funkce zrcadlení portů [69].

Defaultně sonda využívá otevřenou sadu pravidel Emerging Threats, kterou automaticky aktualizuje pomocí nástroje Suricata-Update či podobného nástroje Oinkmaster. Načtení nových pravidel Suricata umožňuje bez nutnosti službu restartovat, takže aktualizace pravidel může na sondě probíhat bez ztráty její schopnosti detekovat síťové hrozby. Aktualizace je nastavena denně na 3 hodiny ráno, tedy čas, ve kterém je síťový provoz obvykle minimální, aby se zabránilo zbytečnému zatěžování sondy při plném síťovém provozu [69].

Pro správu bezpečnostních incidentů je pro obě verze sondy možné využít mobilní aplikaci pro OS Android, která se k sondě připojuje a komunikuje pomocí protokolu Secure Shell (SSH), a pro verzi Industry také webovou aplikaci EveBox. Obě aplikace čerpají z logu Suricatou detekovaných událostí Eve JSON [69].

Další informace o sodně jsou dostupné i z oficiální webové stránky [70].

Kapitola 4

Zabezpečená komunikace

4.1 Úvod do kryptografie

Kryptografie poskytuje mechanizmy pro šifrování a podepisování dat. Šifrování zajišťuje důvěrnost dat a podepisování jejich integritu a autentizaci. Symetrická kryptografie, která využívá jediného sdíleného klíče, je vhodná pro šifrování velkých dat, nicméně vyžaduje, aby se jednotlivé entity shodli na použitém klíči. Pro zabezpečení dohody sdíleného klíče je možné použít asymetrickou kryptografii, která používá pár veřejného a soukromého klíče. Kombinováním obou typů kryptografie je tak poskytnuto silné zabezpečení komunikace [88].

4.1.1 Symetrická kryptografie

V symetrické kryptografii je jediný šifrovací klíč použit pro šifrování i dešifrování předávaných zpráv. Vyžaduje nižší výpočetní výkon a proto se používá pro šifrování velkých bloků dat. Využívá se například pro šifrování celého datového toku SSH spojení [89].

Existují 2 hlavní typy symetrických šifer - blokové a proudové. Zatímco prodové šifry zpracovávají data po jednotlivých bitech, blokové šifry, které využívají například algoritmy AES (Advanced Encryption Standard) či DES (Data Encryption Standard), zpracovávají data po blocích. Jednotlivé bloky mají konstantní počet bitů, který musí být nižší než velikost šifrovacího klíče [88].

Algoritmus AES pracuje s bloky o velikost 128 bitů a délkami klíčů 128, 192, nebo 256 bitů. Algoritmus využívá sítě substitučních a permutačních bloků, které jsou na šifrovaný text aplikovány několikrát s různými hodnotami těchto bloků [88].

Blokové šifry se používají v kombinaci s volbou režimu, který rovněž udává kvalitu zabezpečení šifrovaného textu. Mezi základní režimy patří ECB (Electronic Code Book), CBC (Cipher Block Chaining) a GCM (Galois Counter Mode). Nejjednodušší a nejméně bezpečný režim je ECB, který

každý blok šifruje zvlášť a bloky se stejnými bity jsou tak na výstupu šifrovány totožně. V CBC režimu jsou jednotlivé bloky zřetězeny a způsob šifrování následující bloku je vždy závislý na výstupu předchozího bloku. První blok pro svůj vstup využívá náhodně generovaný inicializační vektor. Pokročilejší režim GCM bloky zpracovává po vzoru proudové šifry a umožňuje zajistit integritu dat [88].

Limitujícím faktorem bezpečnosti je velikost šifrovacího klíče. Malé klíče mohou být snáze kompromitovány brute-force útoky [88].

4.1.2 Asymetrická kryptografie

Asymetrická kryptografie pro šifrování a dešifrování používá 2 rozdílné klíče - veřejný a soukromý. Zatímco veřejný klíč může být veřejně známý, k soukromému klíči musí mít přístup vždy pouze entita, které patří. Asymetrická kryptografie je založena na předpokladu, že odvození soukromého klíče ze znalosti veřejného klíče je při použití současných výpočetních technologií extrémně časově náročné a vypočíst jej v rozumném čase je prakticky nemožné [88].

Text, který je zašifrován pomocí veřejného klíče, lze dešifrovat pomocí soukromého klíče a naopak. Veřejný klíč je předán jiným entitám, které jím zašifrují zprávu, kterou pak může dešifrovat pouze první entita, která drží příslušný soukromý klíč. Veřejný klíč je tedy použit pro zašifrování zprávy a soukromý klíč pro její dešifrování. Asymetrická kryptografie zároveň umožňuje ověřit identitu odesílatele zprávy. Pokud je daná zpráva šifrována soukromým klíčem, kterákoliv entita, která drží příslušný veřejný klíč, může ověřit, že daná zpráva byla opravdu odeslána první entitou [89].

Nejrozšířenějším algoritmem, který využívá principů asymetrické kryptografie je RSA [88].

4.1.3 RSA klíče

Algoritmus byl vyvinut v roce 1977 a nese jména jeho tvůrců. Algoritmus je výpočetně velmi náročný a proto se obvykle nepoužívá pro přímé šifrování zpráv. Je ale používán pro šifrování procesu generování sdíleného klíče (např. AES) mezi dvěma entitami, které pak následující komunikaci mezi sebou šifrují tímto vytvořeným sdíleným klíčem. Algoritmus je vhodný i pro podepisování dat [88].

Generované RSA klíče mohou být uchovány v různých formátech, mezi kterými lze převádět. Nejrozšířenějším je formát PEM (Privacy-enhanced Electronic Mail), jehož původním účelem bylo šifrování e-mailů. Záhlaví a zápatí PEM formátu obsahuje definici obsahu těla souboru, kterým může být například soukromý klíč, soukromý klíč chráněný heslem či X.509 certifikát. Pokud je PEM formát použit pro uchování kryptografického klíče, je použit formát zvaný PKCS #8 (Public Key Cryptographic Standards #8), který pro popis obsahu používá jazyk ASN.1 (Abstract Syntax Notation

One). Tělo PEM formátu vždy začíná znaky *MII*. První verze standardu PKCS byla vyvinuta přímo pro uchovávání RSA klíčů. V záhlaví a zápatí je při použití této verze specifikován název algoritmu RSA. Generování klíčů v těchto formátech umožňuje OpenSSL [90].

Soukromé klíče generované pomocí OpenSSH používají formát PKCS #1. Veřejné klíče používají SSH formát, který v kryptografickém světě není standardní. SSH formát je rozdělen na 3 části, ve kterých je specifikován algoritmus klíče, samotný klíč a případný komentář. Všechny veřejné RSA klíče v SSH formátu začínají znaky *AAAAB3NzaC1y* [90].

4.2 Secure Shell

SSH zabezpečuje komunikaci mezi dvěma zařízeními šifrováním jejich vzájemné komunikace. Umožňuje vzdálené zabezpečené vykonávání příkazů na cizím zařízení, čímž poskytuje zabezpečenou alternativu k protokolům jako Telnet či RSH (Remote Shell), a vzdálené kopírování souborů, čímž poskytuje zabezpečenou alternativu pro protokoly jako FTP (File Transfer Protocol) a RCP (Remote Copy Protocol) [91].

SSH používá tradiční klient-server architekturu, ve které se SSH klienti připojují k SSH serverům [91]. SSH server zpracovává příkazy od SSH klientů, vykonává je a odpověď příkazu odesílá zpět klientovi [92].

První verze SSH1 byla vyvinuta v roce 1995 ve Finsku a byla poskytnuta jako volný software. Bezpečnostní problémy a omezení první verze daly v roce 1996 vzniknout verzi SSH2 [92], ve které byl protokol kompletně přepsán. Protokol SSH2 poskytuje vyšší zabezpečení, výkon a flexibilitu [91] Volně dostupnou implementaci protokolu SSH2 poskytuje OpenSSH, které je součástí mnoha operačních systémů [92].

Pro připojení se k SSH serveru potřebují SSH klienti znát pouze jeho IP (Internet Protocol) adresu či jméno a port, na kterém poslouchá. Ačkoliv SSH služba může běžet na jakémkoliv portu, nejčastěji poslouchá na portu 22. Vyžadovaná autentizace závisí na konfiguraci SSH serveru [91].

Všechna přenášená data mezi serverem a klientem jsou šifrována. SSH umožňuje několik možností autentizace a integraci autentizace s jinými systémy jako Kerberos či SecureID [92]. Typ vyžadované autentizace závisí na konfiguraci SSH serveru. Integrita dat je zajištěna digitálním podpisem přenášených dat, kterým je ověřeno, že data nebyla změněna či poškozena [91].

Základem SSH jsou SSH klíče, které jsou použity jako parametry v SSH algoritmech pro šifrování i autentizaci. Asymetrické klíče jsou použity pro vytvoření a výměnu krátkodobých symetrických klíčů. V SSH jsou použity trvalé uživatelské a hostitelské klíče, které pomáhají vzájemně identifikovat klientské zařízení a server, a soukromé krátkodobé klíče relace, které jsou unikátní pro dané SSH spojení a jsou použity pro šifrování přenášených dat mezi klientem a serverem a ověření integrity dat. Jejich platnost končí při ukončení spojení dané relace [92].

Pro přenášení souborů SSH poskytuje protokol SCP (Secure Copy Protocol) nebo SFTP (Secure File Transfer Protocol). Vzdálené přenášení souborů je podobné vzdálenému vykonávání příkazů. SFTP je podobný FTP, nicméně pro přenos používá šifrované spojení vytvořené mezi SSH klientem a serverem [91].

4.2.1 Protokoly

Protokol SSH operuje nad TCP vrstvou a tvoří jej 4 základní protokoly - SHH Transport Layer Protocol, který navazuje a spravuje počáteční spojení, SSH Authentication Protocol, který se stará o autentizaci klienta, SSH Connection Protocol, který spravuje vytvořené SSH spojení a SFTP, který se stará o vzdálený přístup k souborům a jejich přenos [92].

Transportní protokol se stará o počáteční výměnu klíčů pro zabezpečení komunikace, která většinou vychází z Diffie-Hellmanovy výměny klíčů. Pro zvýšení zabezpečení je proces výměny klíčů obvykle během vytvořené relace několikrát opakován, obvykle každou hodinu či po každém přeneseném gigabytu dat. SSH umožňuje nastavit, aby komunikace nebyla šifrována, nicméně tato možnost je určena pouze a jenom pro účely testování. Po počáteční výměně klíčů mohou být vyjednány bezpečnostní parametry daného spojení, kdy jsou dohodnuty způsoby zajištění důvěrnosti, integrity a komprese dat. Protokol také zajišťuje autentizaci serveru [92].

Autentizační protokol zajišťuje autentizaci SSH klienta pomocí veřejného klíče, hostitelského klíče či hesla, popř. jiného autentizačního mechanizmu. Klient serveru posílá žádost o autentizaci, která obsahuje jméno uživatelského účtu, jméno serveru, název použité autentizační metody a příslušná data pro autentizaci, např. heslo či veřejný klíč. Server následně pošle klientovi odpověď, zda byla autentizace úspěšná [92].

Protokol spojení zajišťuje kontrolu datového toku a vytváří logické kanály nad transportní vrstvou, identifikované číslem, které mohou být různých typů, například pro vykonávání vzdálených příkazů či přesměrování portů [92]. Více kanálů je multiplexováno do jednoho SSH spojení, v rámci kterého je možné otevírat a zavírat více kanálů, přičemž kanál může být otevřen jak na straně klienta, tak na straně serveru. Mezi odesílatelem a příjemcem je s každým kanálem nakládáno jako s nezávislým spojením [89].

4.2.2 Autentizace

Účelem procesu autentizace je ověřit identitu připojovaného zařízení. SSH poskytuje několik autentizačních metod v závislosti na typu a konfiguraci SSH serveru. SSH pro autentizaci může například vyžadovat uživatelské jméno a heslo či veřejný klíč. Tyto možnosti mohou být použity i společně pro dvoufázové ověření. Další posílení zabezpečení může být použití SecureID či certifikátů [91].

Proces autentizace začíná tím, že SSH klient poskytne autentizační údaje SSH serveru. Proces, kdy klient serveru prezentuje uživatelské jméno a heslo nebo veřejný klíč, je kompletně šifrovaný. Pokud je autentizace úspěšná, započne SSH relace, ve které je veškerá komunikace šifrovaná [91].

Heslo

Autentizace pomocí hesla je nejjednodušší forma autentizace. Ačkoliv je používána nejčastěji, pro přenos citlivých dat není vhodná, jelikož hesla jsou často slabá, je snadné jejich sdílení, mohou být na ně použity brute-force útoky, je obtížné rozeznat, že heslo bylo kompromitováno, a často jsou uživateli používána pro více aplikací, které mohou mít slabší zabezpečení a být tak kompromitována odtud. SSH server může být nakonfigurován tak, aby autentizaci na základě hesla nepřijímal [91].

Hostitelský klíč

Další možná metoda autentizace je pomocí hostitelského klíče, který je unikátní pro konkrétní zařízení. Tímto může i SSH klient ověřit identitu SSH serveru, ke kterému se připojuje. Při první autentizaci k SSH serveru klientovi server pošle veřejný hostitelský klíč, aby se mu identifikoval. Klient si tento klíč může uložit do databáze hostitelských klíčů a použít jej pro ověření identity SSH serveru při příštím připojení, kdy klient porovná serverem prezentovaný klíč s klíči uloženými v databázi. V případě, že se klíče neshodují, může se jednat o potenciální man-in-the-middle útok či jinou formu útoku a klient může odmítnou připojení k serveru. Hostitelský klíč může být použit také pro autentizaci klienta serveru, pokud klient svůj veřejný hostitelský klíč překopíruje na server, který pak autentizuje klienta na základě prezentovaného klíče, který porovná s uloženým klíčem. SSH server musí být nakonfigurován tak, aby podporoval autentizaci na základě hostitelského klíče [91].

Veřejný klíč

Autentizace pomocí veřejného klíče je jedna ze silnějších metod autentizace, které SSH poskytuje. Autentizace pomocí klíčů využívá veřejného a soukromého páru SSH klíčů. Klient nejprve generuje klíčový pár, přičemž generovaný soukromý klíč může být chráněn heslem. Generovaný veřejný klíč je pak nahrán na server, obvykle do jeho domovské složky, a je připojen do souboru se seznamem autorizovaných klíčů. Každý klient pak musí pro autentizaci prezentovat veřejný klíč [91].

Každý uživatel musí mít vlastní pár veřejného a soukromého klíče, kterým se serveru autentizuje. Generování tohoto klíče probíhá přímo na klientském zařízení, na kterém jsou klíče uloženy. Na SSH server je poslána a uložena kopie veřejného klíče. Pro samotnou autentizaci klient potřebuje veřejný i soukromý klíč. Soukromým klíčem je ověřeno, že klient je skutečný majitel prezentovaného veřejného klíče. Veřejný klíč je při autentizaci poslán SSH

serveru, který jej porovná s klíčem, který má na serveru uložený pro daného uživatele. Pokud se klíče shodují, klient je autentizován [91].

SSH server je defaultně nakonfigurován tak, aby autentizaci pomocí veřejného klíče podporoval. Výhodou použití tohoto typu autentizace je, že klíče nemohou být sdíleny tak snadno jako hesla, nefungují brute-force útoky a soukromý klíč navíc může být chráněn heslem, které přidává další vrstvu zabezpečení. Aby se útočník mohl autentizovat SSH serveru, musel by získat kopii veřejného a soukromého klíče a znát případné heslo soukromého klíče [91].

Certifikát

Autentizace pomocí certifikátu poskytuje silnou formu zabezpečení SSH, která navíc poskytuje zjednodušení celého procesu, jelikož nevyžaduje, aby veřejné klíče klientů byly ukládány na serveru. Certifikát obsahuje veřejný klíč, jméno a další informace jako životnost a oprávnění certifikátu a je podepsaný soukromým klíčem certifikační autority. Server je pak nakonfigurován tak, aby důvěřoval všem certifikátům, které jsou podepsány konkrétní certifikační autoritou. Pro ověření toho, že prezentovaný certifikát je podepsaný danou certifikační autoritou, je na serveru uložen veřejný klíč této certifikační autority a cesta k němu je specifikována v konfiguračním souboru. Podobným způsobem může být implementována i autentizace serveru klientskému zařízení [93].

Část II

Praktická část

Kapitola 5

Mobilní aplikace SonIoT

Mobilní aplikace SonIoT slouží pro správu obou verzí bezpečnostní sondy SonIoT, která je blíže popsána v kapitole 3.

Minimální verze OS Android, kterou aplikace podporuje, je verze 6.0 Marshmallow (API 23). Dle dat z 10. dubna tedy aplikace podporuje 84,9% aktuálně používaných zařízení [94]. Tato verze byla pro aplikaci vybrána jako minimální, jelikož nižší verze OS Android nebyly kompatibilní s některými použitými knihovnami. Zároveň zvolená verze pokrývá dostatečně velké množství zařízení a dle dostupných dat distribuce jednotlivých verzí [94] je verze 6.0 druhá nejpoužívanější.

5.1 Verze aplikace

Na vývoji mobilní aplikace jsem začala pracovat v rámci Projektu 2 v magisterském studiu. Poté jsem aplikaci otestovala a chyby nalezené při testování jsem opravila, čímž vznikla první verze aplikace. Následně jsem velkou část kódu upravila a přidala jsem některé funkce, a tak vznikla druhá verze aplikace, která je výstupem této diplomové práce.

Nejzásadnější rozdíl mezi verzemi aplikace je ve způsobu autentizace při připojování aplikace k sondě. První verze aplikace využívala pro první přihlášení autentizaci pomocí hesla a pro následující přihlášení autentizaci veřejným klíčem. Druhá verze aplikace místo autentizace veřejným klíčem využívá autentizaci pomocí certifikátu a při prvním přihlášení pomocí hesla se automaticky odpojí a zpět připojí s autentizací pomocí certifikátu.

Mezi další přínosy druhé verze patří:

- Možnost připojení a odpojení sondy přímo z hlavní obrazovky
- Zobrazení stavu připojování k sondě uvnitř tlačítka pro připojení
- Odebrání možnosti sondu vypnout
- Možnost omezit počet zobrazených položek v seznamu bezpečnostních incidentů

- Použití Paging knihovny pro seznam bezpečnostních incidentů pro jeho rychlé načtení
- Zobrazení tří teček na konci řádku signatury v položce bezpečnostního incidentu v seznamu, pokud je signatura příliš dlouhá
- Možnost manuálně spustit skenování zařízení v síti
- Zobrazení nalezených zařízení v síti při probíhajícím skenování
- Změna API pro získání jména výrobce zařízení nalezeného v síti
- Řazení nalezených zařízení v síti v seznamu dle jejich IP adresy
- Použití knihovny EventBus pro předávání událostí v kódu (změna stavu připojení sondy, nalezení zařízení při skenování, detekování problému na sondě)

Pokud neuvedu jinak, následující text bude popisovat druhou verzi aplikace.

5.2 Popis uživatelského rozhraní

Uživatelské rozhraní aplikace je rozděleno na 4 hlavní části, mezi kterými je možné přecházet prostřednictvím bottom navigation baru. Jedná se o část *Home*, která zobrazuje hlavní obrazovku a poskytuje správu připojení sondy, *Security*, která zobrazuje bezpečnostní incidenty z databáze, *Devices*, která zobrazuje seznam zařízení v síti a část *Blocking*, která poskytuje správu blokace webových stránek.

5.2.1 Home

V části *Home* se nachází hlavní obrazovka aplikace, která tlačítkem v pravé části app baru umožňuje vstup na obrazovku nastavení komunikace.

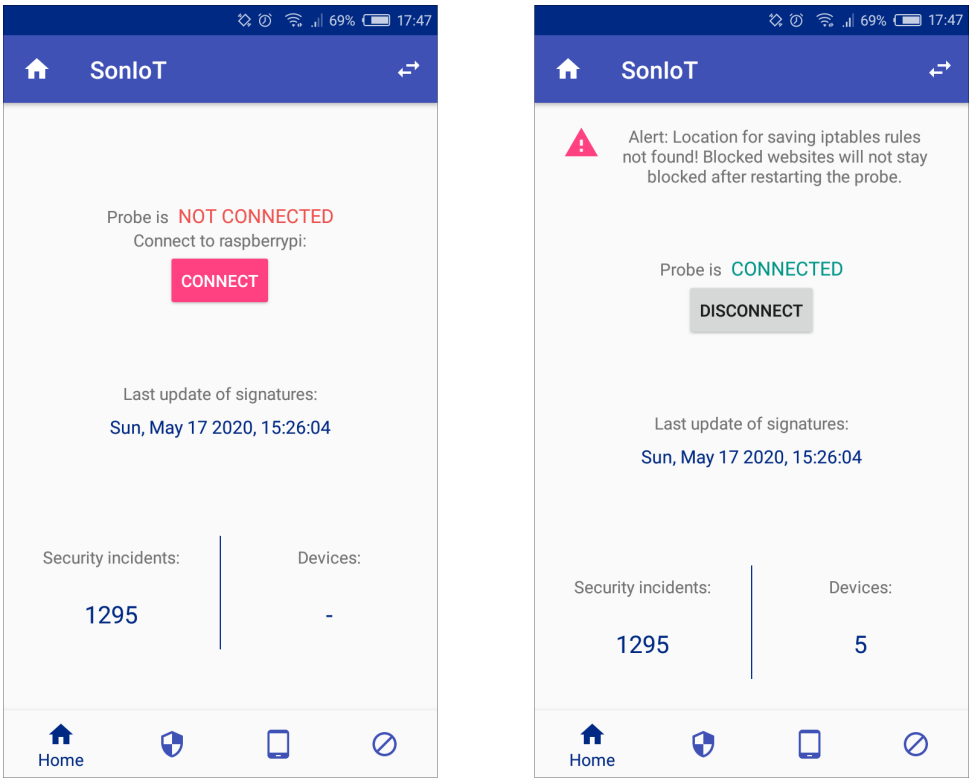
Hlavní obrazovka

Na hlavní obrazovce, která je na obrázku 5.1, je zobrazen stav sondy (zda je připojena), tlačítko pro připojení či odpojení sondy, datum a čas poslední aktualizace bezpečnostních incidentů, počet všech bezpečnostních incidentů v databázi, počet nalezených zařízení v síti a pokud je sonda připojená a byly nalezeny nějaké problémy (nebyl nalezen soubor s logy bezpečnostních incidentů či pro ukládání pravidel iptables, nebo byl detekován rozdílný čas na sondě a na zařízení), je zobrazeno i příslušné upozornění.

Pokud je sonda připojena, je zobrazeno tlačítko pro odpojení. Pokud je sonda odpojena, je zobrazeno tlačítko pro připojení. Pokud jsou v aplikaci uloženy údaje použité pro poslední připojení k sondě, při stisknutí tlačítka dojde k pokusu o připojení se k sondě s těmito údaji a nad tlačítkem se zobrazí uložené jméno (popř. IP adresa) zařízení, ke kterému se mobilní zařízení bude připojovat. Pokud se jedná o první připojení (aplikace nemá uložené údaje pro připojení), je spuštěna aktivita nastavení komunikace.

Pokud připojení nebylo úspěšné, je zobrazena chybová hláška.

Jelikož pro aktualizaci počtu bezpečnostních incidentů jsou použita LiveData, je zobrazené číslo aktualizováno vždy, když se počet bezpečnostních incidentů uložených v databázi aplikace změní. Podobně i počet nalezených zařízení v síti je aktualizován vždy, když je nalezeno nové zařízení. Pokud bylo skenování sítě provedeno dříve, je před dokončením nového skenování zobrazen počet zařízení v síti z minulého skenování a při nalezení nového zařízení v síti je aktualizováno jen tehdy, pokud je počet zařízení v síti vyšší, než při minulém skenování. Pokud vyšší není, je počet aktualizován až při dokončení celého skenování. Skenování zařízení v síti je spuštěno vždy při spuštění aktivity, pokud při jejím spuštění již neprobíhá a pokud je zařízení připojeno k síti.



Obrázek 5.1: Hlavní obrazovka před připojením (vlevo) a po připojení se zobrazeným upozorněním (vpravo)

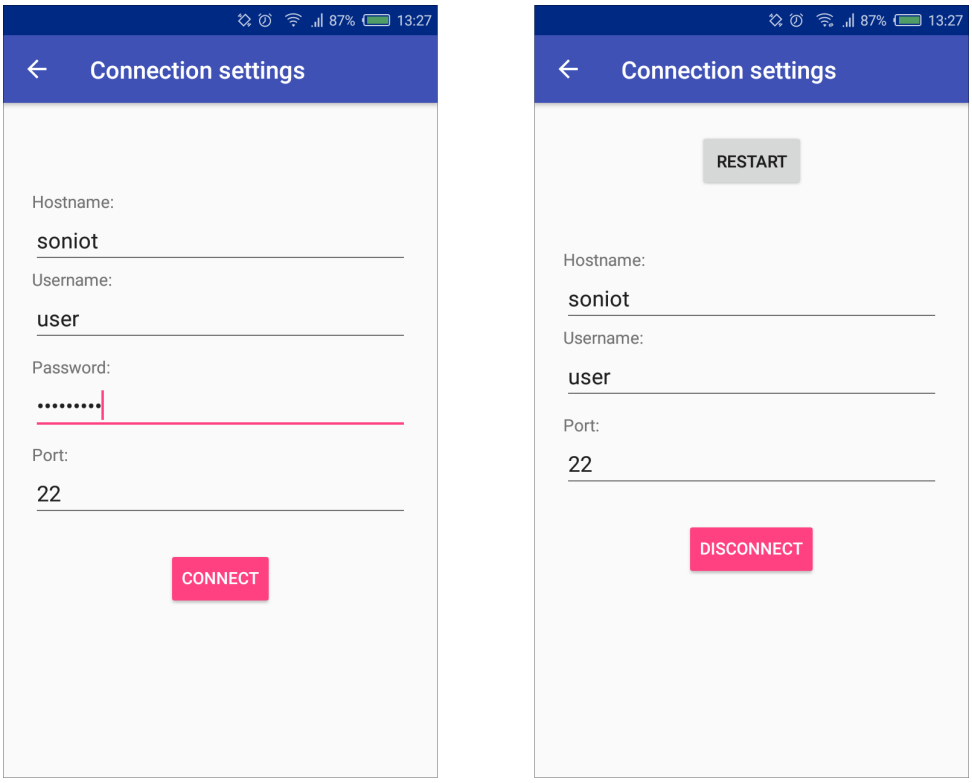
Nastavení komunikace

Před prvním připojením mobilního zařízení k sondě jsou na obrazovce nastavení komunikace 4 editační pole pro zadání jména či IP adresy sondy v síti, jména uživatelského účtu, příslušného hesla a port, na které se aplikace bude připojovat. Tento stav je zobrazen na obrázku 5.2 vlevo. Při dalších připojeních se už pole pro zadání hesla nezobrazuje, jelikož je použita autentizace pomocí certifikátu. Zároveň jsou pak všechna 3 pole

předvyplněna údaji použitými pro poslední připojení, které jsou v aplikaci uchovány uvnitř `SharedPreferences`.

Po stisknutí tlačítka pro připojení se k sondě je spuštěn proces připojení a autentizace a pole s údaji použitými pro připojení přestanou být editovatelná a tlačítko pro připojení stisknutelné. Pokud bylo připojení úspěšné, tlačítko připojení se změní na tlačítko odpojení a zobrazí se nové tlačítko pro restartování sondy. Tento stav je zobrazen na obrázku 5.2 vpravo.

V případě, že připojení úspěšné nebylo, je na obrazovce vypsána chybová hláška a pokud byla použita autentizace pomocí certifikátu, je zobrazen dialog, který umožňuje uživateli pokusit se o připojení s autentizací pomocí certifikátu znova, nebo se vrátit k použití autentizace pomocí hesla a vytvořit nový pár RSA klíčů a certifikát pro autentizaci.



Obrázek 5.2: Obrazovka nastavení komunikace před prvním připojením (vlevo) a po připojení (vpravo)

5.2.2 Security

V části *Security* se nachází obrazovka zobrazující seznam bezpečnostních incidentů. Stisknutím konkrétního bezpečnostního incidentu v seznamu je zobrazena obrazovka detailu daného bezpečnostního incidentu. Tlačítkem v pravé části app baru také umožňuje vstup na obrazovku filtrování bezpečnostních incidentů, která umožňuje filtrovat incidenty zobrazené

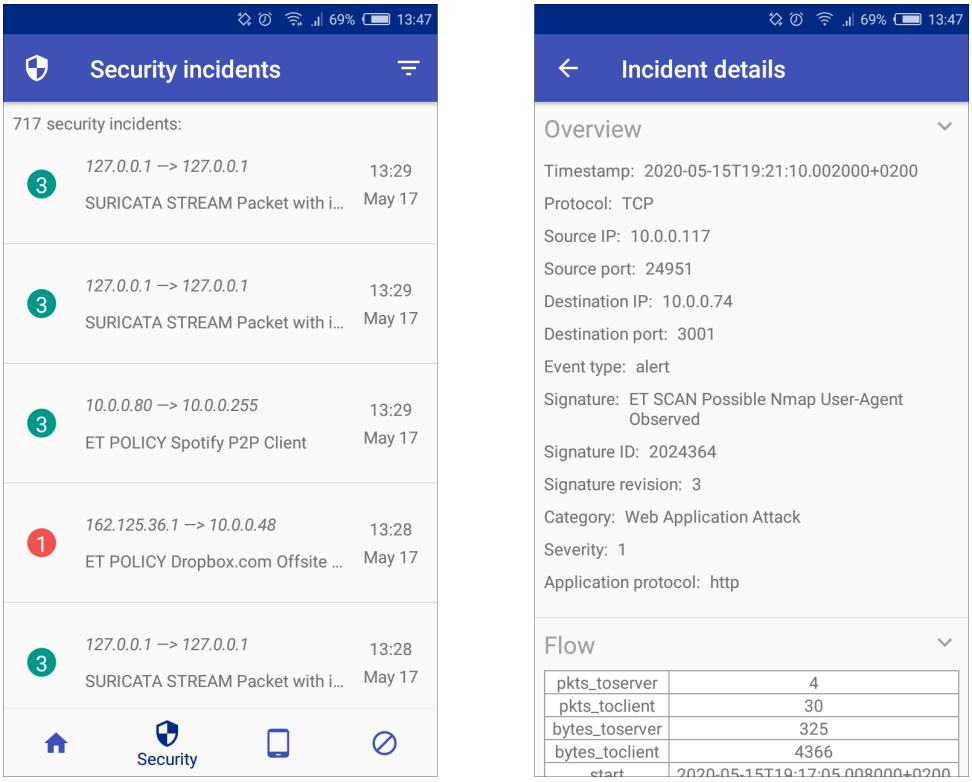
v seznamu.

Seznam bezpečnostních incidentů

Na obrazovce, jejíž příklad je zobrazen na obrázku 5.3 vlevo, je pouze seznam bezpečnostních incidentů v databázi, který obsahuje buď všechny záznamy v databázi, nebo jen záznamy, které odpovídají uživatelem nastavenému filtru. Nad seznamem je zobrazen počet bezpečnostních incidentů, které jsou v seznamu aktuálně zobrazeny.

Každá položka seznamu zobrazuje závažnost 1-3, zdrojovou a cílovou IP adresu, signaturu a časovou značku příslušného bezpečnostního incidentu. Rozdílné hodnoty závažností incidentů jsou v seznamu barevně odlišeny. Závažnost 1, která reprezentuje nejvyšší závažnost, má červenou barvu, závažnost 2 má modrou barvu a závažnost 3 barvu zelenou.

Pro zobrazení dat v seznamu aktivita využívá třídu `PagedListAdapter` [95], která se stará o animaci změn v seznamu při načítání `PagedList` objektů, které drží část zobrazovaných dat z databáze. Při přidání nového bezpečnostního incidentu do seznamu je seznam vždy rolován na první pozici, takže má uživatel přehled o nových bezpečnostních incidentech.



Obrázek 5.3: Obrazovka seznamu bezpečnostních incidentů (vlevo) a obrazovka detailu bezpečnostního incidentu (vpravo)

Detail bezpečnostního incidentu

Obrazovka zobrazuje detailní informace o konkrétním bezpečnostním incidentu v databázi aplikace. Informace jsou rozděleny do dvou rozbalovacích částí - přehled a datový tok. Při spuštění aktivity je přehled rozbalen a datový tok zabalen. Záhlaví každé části obsahuje obrázek šipky, která ukazuje směrem vlevo, pokud je část zabalena, a dolů, pokud je rozbalena. Přechod mezi těmito stavy je animován otočením šipky pomocí třídy `RotateAnimation` [96]. Pro funkci rozbalování je v aplikaci použita knihovna `ExpandableRelativeLayout` [97], která se stará i o příslušnou animaci rozbalení či zabalení. Příklad obrazovky detailu bezpečnostního incidentu je na obrázku 5.3 vpravo.

Filtrování bezpečnostních incidentů

Obrazovka umožňuje nastavit filtry pro bezpečnostní incidenty, které budou zobrazeny na obrazovce seznamu bezpečnostních incidentů, a také vybrat, zda v seznamu budou řazeny dle času či závažnosti. Při řazení dle času jsou nejdříve zobrazeny nejnovější bezpečnostní incidenty a při řazení dle závažnosti jsou jako první zobrazeny nejzávažnější bezpečnostní incidenty.

V horní části obrazovky je zobrazen počet aktuálně aplikovaných filtrů dle uložených hodnot a tlačítko *"Reset all"* pro resetování všech filtrů i řazení do původního stavu - tedy bez filtrů s řazením dle času. Změnu je vždy nutné potvrdit stisknutím tlačítka na pravé straně app baru. Nastavené filtry jsou po jeho stisknutí uloženy do `SharedPreferences`.

Obrázek 5.4: Obrazovka filtrování bezpečnostních incidentů

Bezpečnostní incidenty je v aplikaci možné filtrovat dle závažnosti, zdrojové a cílové IP adresy, signatury, času a je možné omezit maximální počet položek v seznamu. U pole pro výběr limitu a času od a do je, pokud je uživatelem zadána hodnota pro filtr, zobrazen křížek pro snadné resetování filtru.

Pro filtrování závažností, zdrojových a cílových IP adres a signatur je uživateli zobrazen dialog se zaškrtávacími poli, pomocí kterých uživatel zaškrtne, zda daná hodnota parametru má být zobrazena. Ve výběrovém dialogu jsou zobrazeny všechny rozdílné hodnoty parametru, které jsou v databázi bezpečnostních incidentů obsaženy.

Pro filtrování dle času uživatel v aplikaci může specifikovat datum a čas od a do kterého budou bezpečnostní incidenty vybrány. Vybraný čas je porovnáván s časovou značkou incidentu a je možné v aplikaci specifikovat i pouze počáteční či pouze konečný čas. Pro výběr dne je zobrazen `DatePickerDialog` [98] a pro výběr času `TimePickerDialog` [99].

Příklad obrazovky je zobrazen na obrázku 5.4.

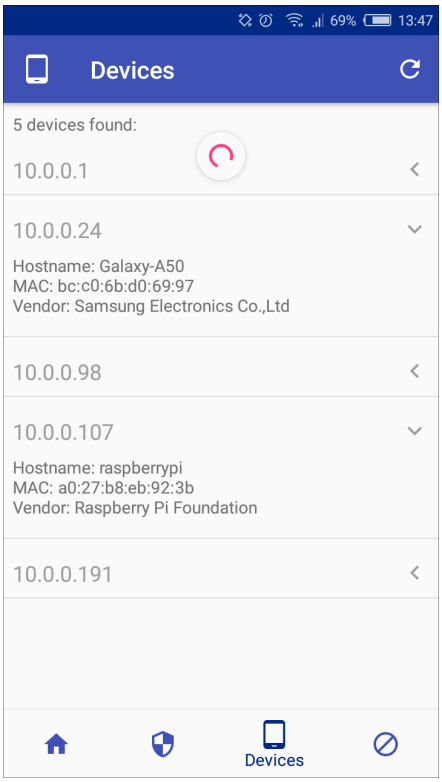
5.2.3 Devices

Část *Devices* sestává z jediné obrazovky, která zobrazuje zařízení nalezená při skenování sítě a umožňuje spustit nové skenování.

Seznam zařízení v síti

Obrazovka zobrazuje seznam zařízení v síti, která při skenování odpověděla na ping, seřazených dle jejich IP adresy. Nad seznamem je zobrazen počet nalezených zařízení, které jsou v seznamu zobrazeny.

Příklad obrazovky seznamu zařízení zobrazuje obrázek 5.5. Každá položka seznamu je rozbalovací a využívá knihovny `ExpandableRelativeLayout` [97] a animované šipky, stejně jako v obrazovce detailu bezpečnostního incidentu 5.2.2. V defaultním, zabaleném stavu je zobrazena pouze IP adresa zařízení, při rozbalení jsou zobrazeny další informace o zařízení - jméno zařízení, MAC (Media Access Control) adresa a výrobce.



Obrázek 5.5: Obrazovka seznamu zařízení v síti

Při probíhajícím skenování je zobrazen kruhový progressbar. Nové skenování je možné začít pomocí tlačítka v pravém horním rohu app baru či gestem potažení prstem na obrazovce shora dolů, pokud předchozí skenování již skončilo.

Při probíhajícím skenování je seznam aktualizován vždy při nalezení nového zařízení v síti. Pro zobrazení dat v seznamu aktivita využívá třídu `ListAdapter` [100], která se stará o animaci změn v seznamu, např. při přidání nové položky. Také vypočítává rozdíly mezi zobrazenými položkami, takže při novém skenování jsou v seznamu aktualizovány jen ty položky, které se změnily. Díky tomu při aktualizaci seznamu rozbalené položky zůstanou rozbalené.

5.2.4 Blocking

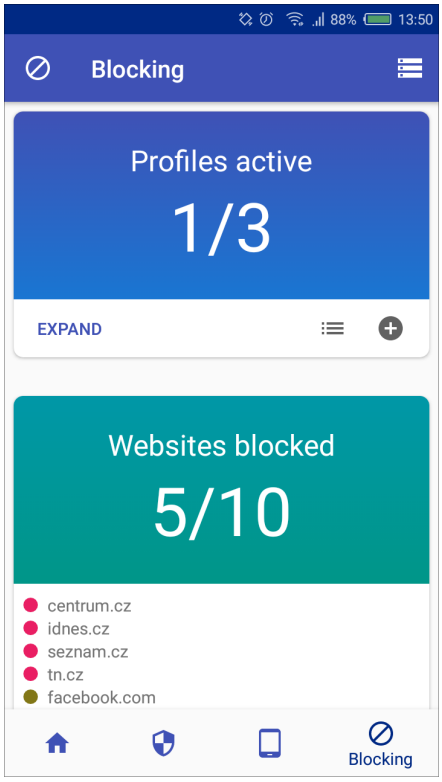
Část *Blocking* uživateli poskytuje rozhraní pro správu blokace webových stránek.

Součástí databáze pro blokaci webových stránek jsou kromě samotných webových stránek také kategorie a profily. Každá webová stránka může být v aplikaci přiřazena do jedné kategorie. Kategorie slouží pro snazší orientaci ve webových stránkách v databázi, ale také pro jednodušší správu blokací. Profily umožňují blokovat více webových stránek najednou tím, že blokují vždy všechny webové stránky ve vybraných kategoriích. Navíc umožňují specifikovat dny v týdnu a časové rozmezí, kdy budou tyto webové stránky pravidelně blokovány.

Při vstupu do části *Blocking* se uživateli zobrazí hlavní obrazovka blokace stránek, která obsahuje základní přehled. Tlačítkem v pravé části app baru je zobrazena obrazovka seznamu kategorií, ze které je možné přejít na obrazovky detailu kategorie a přidání kategorie. Tlačítky na kartách i stisknutím samotné karty je pak z hlavní obrazovky blokace stránek možné přejít na obrazovky seznamu webových stránek a profilů a přidání webových stránek a profilů.

Hlavní obrazovka blokace stránek

Vstupní obrazovka blokace stránek zobrazuje kartu profilů a kartu webových stránek. V těchto kartách je zobrazen seznam a počet aktuálně blokovaných webových stránek, resp. aktivních profilů, počet všech webových stránek, resp. profilů v databázi a tlačítka pro spuštění aktivit seznamu a přidání webových stránek, resp. profilů. Seznam blokovaných webových stránek, resp. aktivních profilů je rozbalovací pomocí tlačítka v levém dolním rohu karty a při spuštění aktivity jsou oba seznamy zabalené. Seznam aktivních profilů kromě názvu profilů také zobrazuje čas, do kdy je profil aktivní, pokud se jedná o profil se specifikovaným časovým oknem. Seznam blokovaných stránek kromě URL (Uniform Resource Locator) adresy webové stránky zobrazuje i barevné kolo reprezentující kategorii, do které patří. Příklad obrazovky je zobrazen na obrázku 5.6.



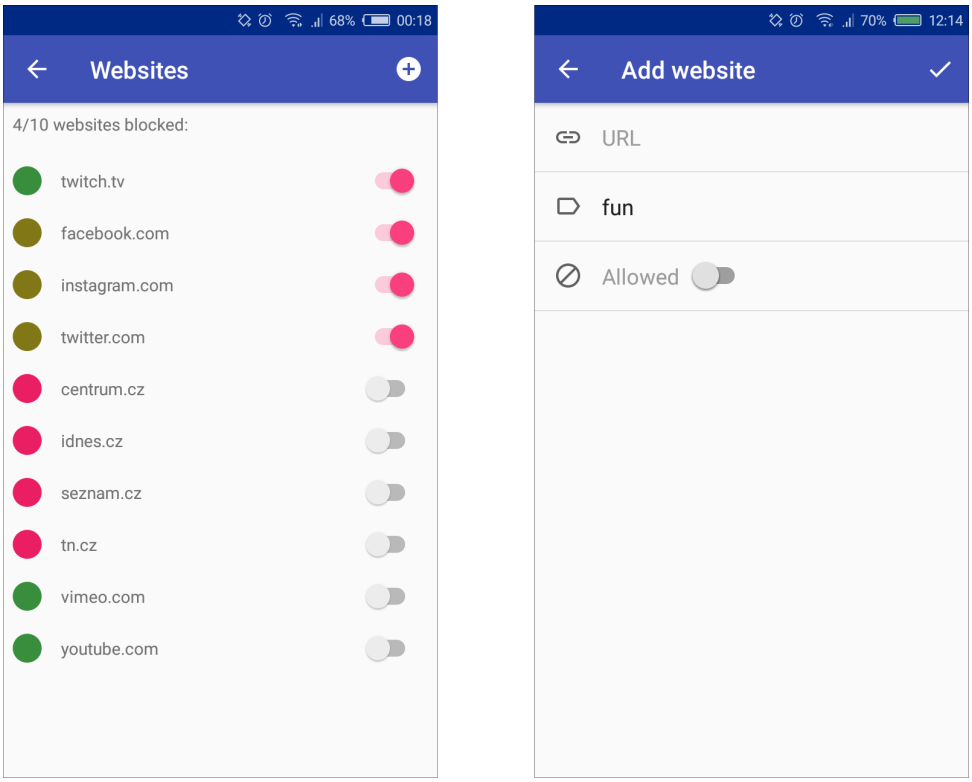
Obrázek 5.6: Hlavní obrazovka blokace stránek

Seznam webových stránek

Obrazovka, jejíž příklad je zobrazen na obrázku 5.7 vlevo, zobrazuje seznam všech webových stránek v databázi. Každá položka seznamu obsahuje barevné kolo, reprezentující kategorii, do které stránka spadá, URL adresu stránky a tlačítkový přepínač, který ukazuje, zda je aktuálně stránka blokována, a umožňuje tento stav změnit, tedy stránku povolit či zakázat. Pokud webová stránka nespadá do žádné kategorie, kolo má bílou barvu. Nad seznamem všech webových stránek je pak zobrazen počet blokovaných a počet všech webových stránek.

Webové stránky jsou v seznamu řazeny nejprve dle blokovaného stavu, kdy blokované stránky jsou zobrazeny první, dále dle kategorií, do kterých spadají, a nakonec dle URL adresy abecedně.

Tlačítkem na pravé stráně app baru je spuštěna aktivita přidání webové stránky. V režimu pro editování webové stránky je pak tato aktivita spuštěna stisknutím některé položky v seznamu webových stránek.



Obrázek 5.7: Obrazovka seznamu webových stránek (vlevo) a obrazovka pro přidání nové webové stránky (vpravo)

Přidání webové stránky

Aktivita může být spuštěna v režimu pro přidání nové webové stránky, nebo v režimu pro editování existující. První režim obrazovky je zobrazen na obrázku 5.7 vpravo. V případě editování je pozměněn titulek aktivity v app

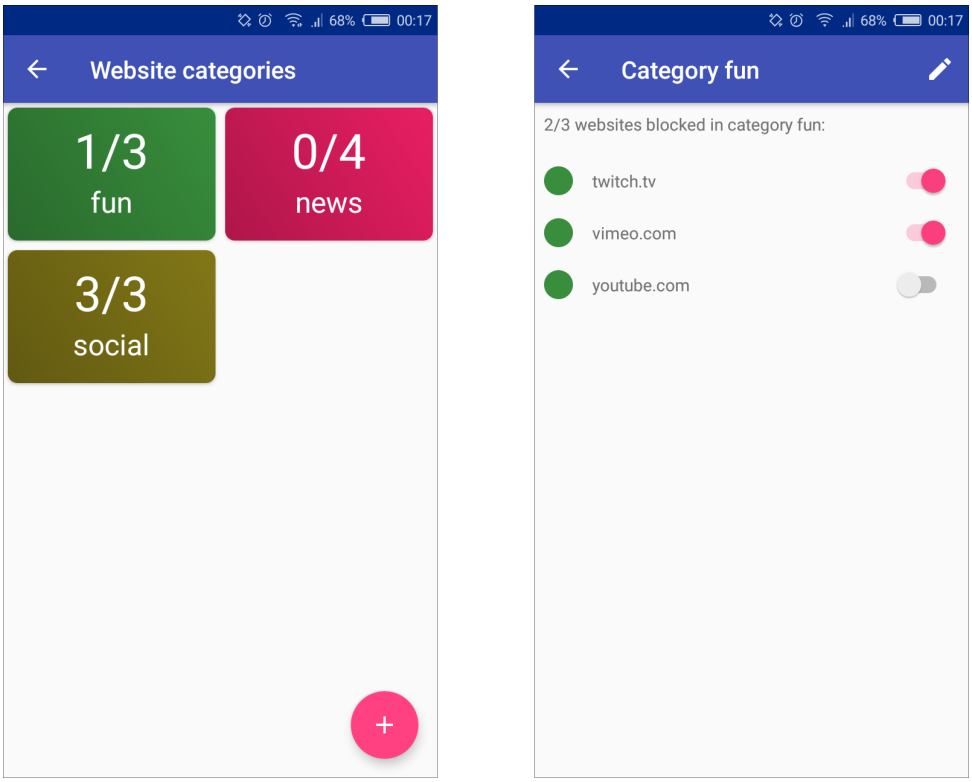
baru, je do něj přidáno tlačítko umožňující smazání webové stránky a pole pro zadávání parametrů jsou předvyplněna.

Uživatel vyplňuje 3 pole - URL adresu webové stránky, kategorii, do které má spadat, a zda má být po uložení v databázi ihned blokována. Pro výběr kategorie je uživateli zobrazen dialog se seznamem všech kategorií v databázi, ze kterého vybere jednu. Výběr kategorie pro webovou stránku nicméně není povinný a pole může zůstat prázdné. Pokud uživatel nezadá URL adresu či pokud zadá adresu, kterou již databáze obsahuje, při snaze o uložení do databáze je uživateli zobrazen chybový dialog.

Seznam kategorií

Obrazovka, jejíž příklad je zobrazen na obrázku 5.8 vlevo, zobrazuje seznam kategorií webových stránek ve mřížce, v jejíž jednom řádku jsou 2 kategorie. Každá položka v seznamu je zobrazena uvnitř karty, jejíž barva pozadí je shodná s vybranou barvou pro kategorii. V kartě je zobrazen počet blokovaných a všech webových stránek v kategorii a název kategorie. Kategorie jsou v mřížce řazeny dle názvu kategorie abecedně.

Stisknutím karty je spuštěna aktivita detailu příslušné kategorie a novou kategorii je možné přidat pomocí plovoucího tlačítka v pravém dolním rohu obrazovky.



Obrázek 5.8: Obrazovka seznamu kategorií (vlevo) a obrazovka detailu kategorie (vpravo)

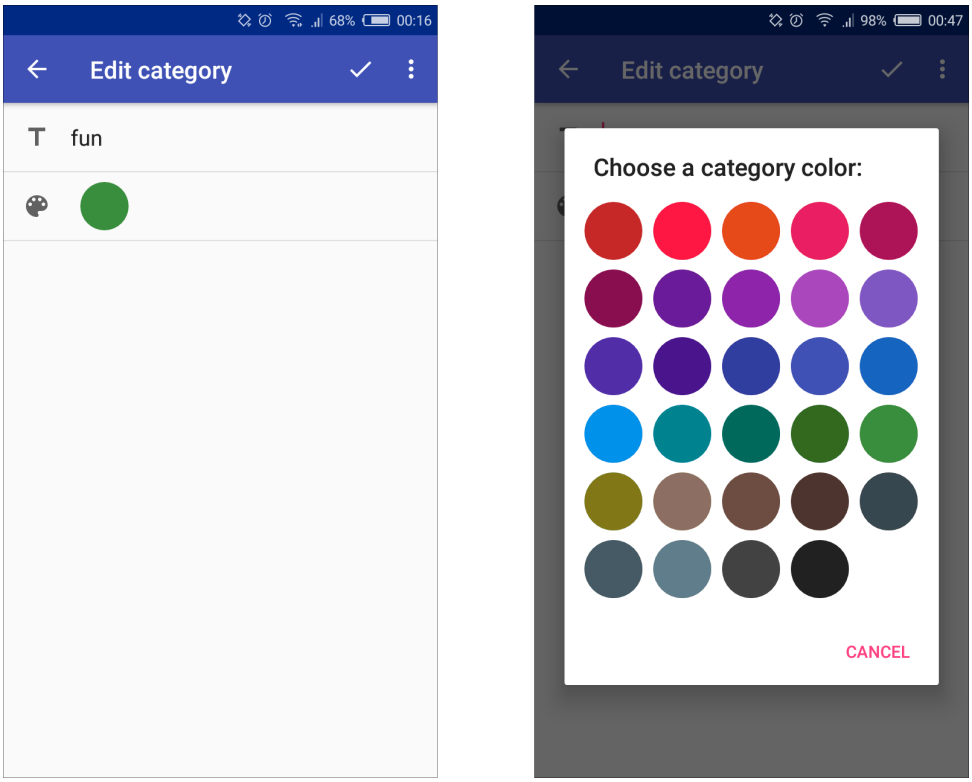
Detail kategorie

Obrazovka detailu kategorie je stejná jako obrazovka seznamu všech webových stránek s tím rozdílem, že jsou v seznamu webových stránek zobrazeny pouze ty stránky, které do příslušné kategorie patří. V app baru je pak tlačítko pro editování kategorie, které spustí aktivitu přidání kategorie v režimu editování. Příklad obrazovky zobrazuje obrázek 5.8 vpravo.

Přidání kategorie

Aktivita může být spuštěna v režimu pro přidání nové kategorie, nebo v režimu pro editování existující. V případě editování je pozměněn titulek aktivity v app baru, je do něj přidáno tlačítko umožňující smazání webové stránky a pole pro zadávání parametrů jsou předvyplněna. Příklad obrazovky v tomto režimu zobrazuje obrázek 5.9 vlevo.

Uživatel vyplňuje 2 pole - název kategorie a její barvu. Obě položky musí být vyplněny, v opačném případě je při pokusu o uložení zobrazen chybový dialog. Pro výběr barvy je uživateli zobrazen dialog s 29 možnými barvami zobrazenými v mřížce, kterou ukazuje obrázek 5.9 vpravo.



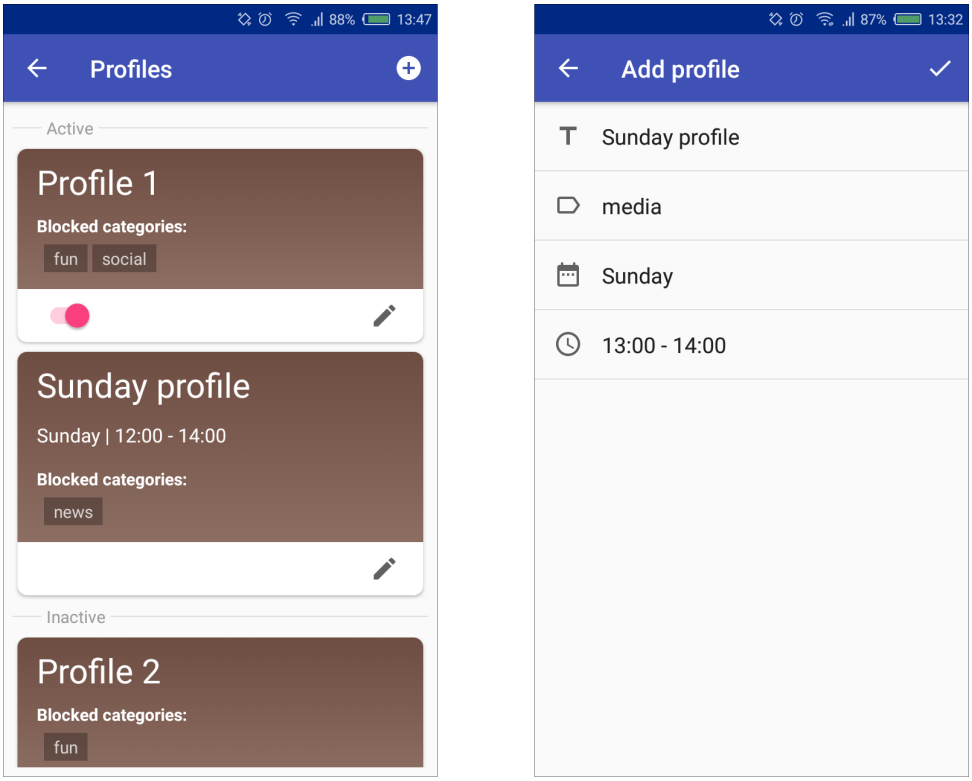
Obrázek 5.9: Obrazovka přidání kategorie v režimu editování (vlevo) a se zobrazeným dialogem pro výběr barvy (vpravo)

Seznam profilů

Obrazovka, jejíž příklad je zobrazen na obrázku 5.10 vlevo, zobrazuje seznam všech profilů uložených v databázi. Jako první jsou v seznamu zobrazeny všechny aktivní profily. Následně jsou v seznamu profily řazeny dle jejich názvu abecedně.

Profily v databázi jsou dvou typů - se specifikovaným časovým oknem a bez něj. Profily se specifikovaným časovým oknem mají specifikované dny a časové okno, kdy jsou pravidelně aktivní. Profily bez specifikovaného časového okna jsou aktivovány a deaktivovány manuálně uživatelem.

Každý profil je zobrazen uvnitř karty, která obsahuje informace o profilu, tlačítko pro jeho editování a v případě, že se jedná o profil bez specifikovaného časového okna, obsahuje také tlačítkový přepínač pro aktivaci či deaktivaci profilu. Informace zobrazované v kartě o profilu jsou jeho název, výčet kategorií, které blokuje, a v případě profilu se specifikovaným časovým oknem také výčet dní s časovým oknem, kdy je profil aktivní. Pro zobrazení výčtu kategorií blokovaných profilem je v aplikaci použita knihovna FlowLayout [101], která se stará o zalamování řádků v případě, že se již název kategorie na něj nevejde celý.



Obrázek 5.10: Obrazovka seznamu profilů (vlevo) a obrazovka přidání profilu (vpravo)

Přidání profilu

Aktivita může být spuštěna v režimu pro přidání nového profilu, nebo v režimu pro editování existujícího. Příklad obrazovky v prvním režimu pro přidání nového profilu se specifikovaným časovým oknem je zobrazen na obrázku 5.10 vpravo. V případě editování je pozměněn titulek aktivity v app baru, je do něj přidáno tlačítko umožňující smazání webové stránky a pole pro zadávání parametrů jsou předvyplněna.

Uživatel vyplňuje až 4 pole - název profilu, výčet kategorií, které blokuje, a v případě profilu se specifikovaným časovým oknem také výčet dní a časy od a do, kdy bude profil aktivní. Pro výběr kategorií se uživateli zobrazí dialog se seznamem všech kategorií v databázi, ze kterých vybírá libovolné množství, minimálně však jednu. Pro výběr dní je zobrazen dialog s výčtem dní v týdnu od pondělí do neděle. Pro výběr časů od a do jsou zobrazeny dialogy `DatePickerDialog` [98] a `TimePickerDialog` [99].

Vyplnění názvu a alespoň jedné kategorie je pro přidání profilu povinné. V případě profilu bez specifikovaného časového okna zůstává pole pro zadání dní nevyplněno. V případě jeho vyplnění je na obrazovce zobrazeno pole pro zadání časového okna, jehož vyplnění je pak povinné. V případě pokusu o uložení profilu s nevyplněnými povinnými parametry je uživateli zobrazen chybové dialogové okno. Stejně tak i v případě uložení profilu se specifikovaným časovým oknem, pokud aplikace není aktuálně připojena k sondě, jelikož by nebylo možné odeslat příkazy pro blokaci na sondu.

5.3 Popis funkcí a použitých knihoven

5.3.1 Předávání událostí mezi třídami

Aplikace využívá knihovnu EventBus [102] pro zjednodušení předávání vytvořených událostí mezi jednotlivými částmi aplikace. V kódu jsou definovány následující události:

- *ConnectionEvent* - připojení se k sondě či odpojení se od sondy, předává daný stav jako `boolean`
- *DevicesFoundEvent* - nalezení zařízení při skenování sítě, předává seznam nalezených zařízení
- *DevicesScanCompletedEvent* - dokončení skenování sítě, nepředává žádný parametr
- *IptablesDirNotFoundEvent* - nenalezení složky pro ukládání `iptables` na sondě, nepředává žádný parametr
- *LogFileNotFoundEvent* - nenalezení souboru s logy událostí Suricaty na sondě, nepředává žádný parametr
- *WrongTimestampEvent* - zjištění různých časů na sondě a mobilním zařízení, nepředává žádný parametr

Nejrpve je vytvořena příslušná událost a odeslána přes EventBus.

```
EventBus.getDefault().post(new Event());
```

Aktivita, která má být o události notifikována, je při volání `onStart()` registrována na EventBus a při volání `onStop()` je její registrace zrušena.

```
@Override
protected void onStart() {
   super.onStart();
   EventBus.getDefault().register(this);
}

@Override
protected void onStop() {
   super.onStop();
   EventBus.getDefault().unregister(this);
}
```

Aktivita pak pro každou událost, o které má být notifikována, implementuje veřejnou metodu anotovanou `@Subscribe`, která jako argument přijímá instanci dané události. Je možné nastavit, v jakém vlákně se akce aktivity na událost bude vykonávat.

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void onEvent(Event event) {
   // akce pri spusteni udalosti
}
```

5.3.2 Připojení a komunikace se sondou

Komunikace mobilní aplikace se sondou probíhá přes SSH protokol. Aplikace k tomu využívá knihovnu SSHJ [103], která podporuje i autentizaci pomocí certifikátu.

V aplikaci je SSH spojení reprezentováno třídou `SSHConnection`, která využívá návrhový vzor Singleton a drží v sobě instanci třídy `SSHClient` knihovny SSHJ, ve které je dále SSH spojení spravováno. Třída `SSHConnection` ostatním částem aplikace poskytuje metody pro připojení a autentizaci k sondě, odpojení od sondy, pro vykonávání vzdálených příkazů na sondě a pro stahování souborů ze sondy.

Připojení a odpojení

Při připojení se k sondě je vytvořena instance třídy `SSHClient` knihovny SSHJ. Před samotným připojením je nastaven časový limit pro připojení

a komunikaci 60 sekund a keep-alive interval, dlouhý 5 sekund, pro zjištění stavu SSH spojení (zda je zařízení k sondě stále připojeno) a jeho udržení.

```
final int TIMEOUT_MS = 10000;
DefaultConfig config = new DefaultConfig();
config.setKeepAliveProvider(KeepAliveProvider.KEEP_ALIVE);
SSHClient sshClient = new SSHClient(config);
sshClient.setConnectTimeout(TIMEOUT_MS);
sshClient.setTimeout(TIMEOUT_MS);
sshClient.getConnection().setTimeoutMs(TIMEOUT_MS);
sshClient.getTransport().setTimeoutMs(TIMEOUT_MS);
sshClient.getConnection().getKeepAlive().setKeepAliveInterval(5);
```

Pro navázání spojení je volána metoda `connect()` objektu `SSHClient`, které se jako parametr předávají hostname/IP adresa sondy a SSH port (standardně port 22), na který se připojuje. Následuje autentizace pro připojení se k uživaltskému účtu na sondě. Různé způsoby autentizace jsou blíže popsány v podkapitole 5.3.3.

```
sshClient.connect(hostname, port);
authentication.authenticate(sshClient, username);
```

Pro ukončení SSH spojení je volána metoda `disconnect()` objektu `SSHClient`. Zároveň je na EventBus poslána událost pro notifikaci jiných částí aplikace, že byla sonda odpojena. Konkrétně je notifikována hlavní aktivita pro zobrazení informace, zda je sonda připojena, či nikoliv.

```
sshClient.disconnect();
EventBus.getDefault().post(new ConnectionEvent(false));
```

Komunikace

Pro posílání příkazů na sondu je vytvořen objekt `Session` knihovny SSHJ, který otevře kanál pro vzdálené vykonání příkazu. Příkaz je na sondě vykonán voláním metody `exec()`, které je jako argument předán příslušný příkaz. Jelikož třída `Session` implementuje rozhraní `Closeable`, blok `try` se postará o uzavření kanálu po vykonání všech operací.

```
try (Session session = sshClient.startSession()) {
   session.exec(command);
}
```

Pokud aplikaci zajímá odpověď vzdáleně vykonaného příkazu, knihovna SSHJ umožňuje jeho přečtení metodou `readFully()` třídy `IOUtils`. Pro

získání odpovědi ze sondy je nastaven timeout 5 sekund, aby se zabránilo blokaci vlákna.

```
String response;
try (Session session = sshClient.startSession()) {
   Session.Command cmd = session.exec(command);
   response = IOUtils.readFully(cmd.getInputStream()).toString();
   cmd.join(5, TimeUnit.SECONDS);
}
```

Při aktualizaci bezpečnostních incidentů je ze sondy stahován soubor, který obsahuje nové bezpečnostní incidenty. Pro stažení tohoto souboru na mobilní zařízení využívá aplikace protokol SFTP, jehož funkcionalitu v aplikaci zajišťuje rovněž knihovna SSHJ.

```
try (Client sftp = sshClient.newSFTPClient()) {
   sftp.get(fileToDownload, outputFileDestination);
}
```

5.3.3 Autentizace

Při prvním připojení se k sondě je vždy použita autentizace pomocí hesla (v kódu reprezentována třídou `PasswordAuthentication`), které je jako argument předáváno heslo pro autentizaci. V první verzi aplikace následující připojení využívala autentizaci pomocí veřejného klíče (třída `PublicKeyAuthentication`). Druhá verze aplikace tento způsob nahradila autentizací pomocí certifikátu (třída `CertificateAuthentication`) pro vyšší úroveň zabezpečení. I když druhá verze aplikace autentizaci pomocí veřejného klíče stále podporuje, nevyužívá ji. Zároveň po prvotní autentizaci pomocí hesla je ve druhé verzi aplikace sonda odpojena a připojena znova, tentokrát s využitím autentizace pomocí certifikátu. V případě, že je autentizace pomocí certifikátu, popř. veřejného klíče, neúspěšná, je uživateli zobrazen dialog, který mu umožňuje využít znova autentizace pomocí hesla a vytvořit nový pár klíčů pro autentizaci.

Všechny tyto způsoby autentizace v aplikaci dědí ze třídy `Authentication`, která jim předepisuje metodu `authenticate()` pro vykonání autentizace k uživatelem definovaného účtu na sondě.

Uživatelem zadané hostname, uživatelský účet a číslo portu, použité pro připojení a autentizaci, jsou v aplikaci uchovány uvnitř `SharedPreferences` pro využití při dalším připojení. Heslo si aplikace neukládá.

Po úspěšné autentizaci pomocí certifikátu, popř. veřejného klíče, je na EventBus poslána událost, která jiné části aplikace notifikuje o úspěšném připojení k sondě.

Autentizace pomocí hesla

Autentizace pomocí hesla probíhá pomocí metody `authPassword` objektu `SSHClient` knihovny SSHJ, které je jako argument předán název účtu a příslušné heslo.

```
sshClient.authPassword(username, password);
```

Generování klíčového páru pro autentizaci

Při autentizaci pomocí hesla je vždy generován pár klíčů pro autentizaci pomocí certifikátu, popř. veřejného klíče. Pro počáteční správu těchto klíčů slouží v aplikaci abstraktní třída `ProbeKeyManager`, ze které dědí třídy `PublicKeyKeyManager` pro správu klíčů pro autentizaci pomocí veřejného klíče a `CertificateKeyManager` pro správu klíčů a certifikátu pro autentizaci pomocí certifikátu. Třída `ProbeKeyManager` předepisuje veřejnou metodu `manage()`, jejíž volání zajistí přípravu klíčů pro následnou autentizaci. Dále také poskytuje metodu `generateKeyPair()`, která za pomoci knihovny JSch [104] vygeneruje RSA klíčový pár, který je dále použit při autentizaci. Pro generování klíčů je použita knihovna JSch, jelikož generuje klíče přímo ve formátu používaného SSH. Soukromý klíč je poté zašifrován a oba klíče jsou následně uloženy ve `SharedPreferences`. Soukromý klíč je uložen zašifrovaný.

```
String generateKeyPair() throws JSchException {
   JSch jsch = new JSch();
   com.jcraft.jsch.KeyPair keyPair =
       com.jcraft.jsch.KeyPair.genKeyPair(jsch,
       com.jcraft.jsch.KeyPair.RSA);
   ByteArrayOutputStream priv = new ByteArrayOutputStream();
   keyPair.writePrivateKey(priv);
   ByteArrayOutputStream pub = new ByteArrayOutputStream();
   keyPair.writePublicKey(pub, null);
   SharedPreferencesHelper sharedPrefs =
       new SharedPreferencesHelper(context);
   String publicKey = pub.toString();
   sharedPrefs.save(KEYPAIR_FILE, publicKey, PUBLIC_KEY_LABEL);
   CryptoProcessor encryptor = new Encryptor(context);
   sharedPrefs.save(KEYPAIR_FILE,
       encryptor.process(priv.toString()),  PRIVATE_KEY_LABEL);
   return publicKey;
}
```

Šiforvání a dešifrování soukromého klíče

Pro šifrování a dešifrování soukromého RSA klíče je v aplikaci generován AES klíč za pomoci třídy `KeyGenerator`, kterou Java poskytuje. Klíč je generován a uložen uvnitř Android úložiště klíčů Android KeyStore.

```
KeyGenerator keyGenerator =
KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES,
    "AndroidKeyStore");
keyGenerator.init(
   new KeyGenParameterSpec.Builder(KEY_ALIAS,
   KeyProperties.PURPOSE_ENCRYPT | KeyProperties.PURPOSE_DECRYPT)
   .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
   .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_NONE)
   .setRandomizedEncryptionRequired(false)
   .build()
);
Key secretKey = keyGenerator.generateKey();
```

Pokud vygenerovaný klíč už v Android KeyStore je, pouze z něj klíč získá.

```
KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
Key secretKey = keyStore.getKey(KEY_ALIAS, null);
```

Při šifrování a dešifrování je soukromý klíč rozdělen na části o definované délce a tyto jednotlivé části jsou zašifrovány/dešifrovány pomocí získaného AES klíče a inicializační hodnoty, která je pro každou část dat generována pomocí Java třídy `SecureRandom` a uložena do `SharedPreferences`.

```
byte[] iv = new byte[12];
new SecureRandom().nextBytes(iv);
new SharedPreferencesHelper(context).save(KEYPAIR_FILE,
    Base64.encodeToString(iv,Base64.DEFAULT), "iv"+chunkNumber);
```

Před každým šifrováním/dešifrováním je šifra pro daný režim (šifrování/dešifrování) inicializována s šifrovacím AES klíčem a inicializační hodnotou ze SharedPreferences pro danou část dat.

```
String transformation = "AES/GCM/NoPadding";
Cipher cipher = Cipher.getInstance(transformation);
byte[] iv = Base64.decode(
    new SharedPreferencesHelper(context).getData(KEYPAIR_FILE,
        "iv"+chunkNumber), Base64.DEFAULT);
cipher.init(CIPHER_MODE, getEncryptionKey(), new
    GCMParameterSpec(128, iv));
```

V případě šifrování (třída `Encryptor`) je pak daná část dat zašifrována a byty převedeny na textový řetězec s použitím kódování Base64.

```
byte[] encoded = cipher.doFinal(chunk.getBytes());
String encrypted = Base64.encodeToString(encoded, Base64.DEFAULT)
```

V případě dešifrování (třída `Decryptor`) je textový řetězec dekódován a byty dešifrovány.

```
byte[] decoded = Base64.decode(chunk, Base64.DEFAULT);
String decrypted = cipher.doFinal(decoded);
```

Obě třídy dědí z abstraktní třídy `CryptoProcessor`. Společná je pro ně veřejná metoda `process()`, která je jinými částmi aplikace volána pro zašifrování či dešifrování soukromého klíče. Metoda rozdělí textový řetězec, který jí je předán jako argument, na části o definované délce, jednotlivé části zašifruje/dešifruje a opět složí do textového řetězce, který obsahuje zašifrovaná či dešifrovaná data.

Příklad dešifrování soukromého klíče, uloženého uvnitř `SharedPreferences`:

```
CryptoProcessor decryptor = new Decryptor(context);
SharedPreferencesHelper sharedPrefs = new
    SharedPreferencesHelper(context);
String publicKey =
    decryptor.process(sharedPrefs.getData(KEYPAIR_FILE,
    PRIVATE_KEY_LABEL));
```

Veřejný klíč

Autentizace pomocí veřejného klíče je použita pouze v první verzi aplikace. Ve druhé verzi aplikace je nahrazena autentizací pomocí certifikátu.

Při první autentizaci je generován RSA klíčový pár, jehož generování bylo popsáno dříve v této kapitole 5.3.3. Následně je na sondu poslán příkaz pro zjištění, zda na sondě existuje soubor pro ukládání autorizovaných klíčů, který vrací 1 pokud soubor existuje, a 0 pokud neexistuje.

```
String response = sshConnection.sendAndGet("test -e
    \".ssh/authorized_keys\" && echo 1 || echo 0");
```

V případě, že neexistuje, pošle sadu příkazů pro jeho vytvoření.

```
sshConnection.send("mkdir .ssh; cd .ssh; touch authorized_keys;
    chmod 700 ~/.ssh/; chmod 600 ~/.ssh/authorized_keys; cd - >
    /dev/null 2>&1");
```

Následně je do souboru přidán vygenerovaný veřejný klíč, čímž je zajištěno, že po prokázání se, že zařízení zná příslušný soukromý klíč, je autentizace pomocí veřejného klíče úspěšná.

```
sshConnection.send("echo \"" + publicKey + "\" >>
    \".ssh/authorized_keys\""));
```

Samotná autentizace pak probíhá pomocí metody `authPublickey()` objektu `SSHClient` knihovny SSHJ, které je jako argument předán název účtu, na který se aplikace přihlašuje, a RSA klíčový pár pro autentizaci.

```
sshClient.authPublickey(username,
    sshClient.loadKeys(decryptedPrivateKey, publicKey, null));
```

Certifikát

Při první autentizaci je generován RSA klíčový pár, jehož generování bylo popsáno dříve v této kapitole 5.3.3. Následně je na sondu poslán příkaz pro podepsání generovaného veřejného klíče certifikační autoritou na sondě, který využívá skriptu na sondě, který jako vstupní argumenty bere veřejný klíč a název uživatelského účtu, na který se aplikace připojuje, a vrací podepsaný certifikát. Ten si pak aplikace uloží do `SharedPreferences`. Cesta ke skriptu na sondě je */home/soniot_app/get-certificate.sh*.

```
 SharedPreferencesHelper sharedPrefs = new
     SharedPreferencesHelper(context);
String publicKey = generateKeyPair();
String certificate =
    sshConnection.sendAndGet("CERTGEN_SCRIPT_LOCATION
    \""+publicKey+"\" "+username);
sharedPrefs.save(KEYPAIR_FILE, certificate, CERTIFICATE_LABEL);
```

Samotná autentizace pak probíhá pomocí metody `authPublickey()` objektu `SSHClient` knihovny SSHJ, které je jako argument předán název účtu, na který se aplikace přihlašuje, a klíčový pár pro autentizaci, kde místo veřejného klíče je předán certifikát uložený ve `SharedPreferences`.

```
sshClient.authPublickey(username,
    sshClient.loadKeys(decryptedPrivateKey, certificate, null));
```

5.3.4 Databáze bezpečnostních incidentů

Databáze bezpečnostních incidentů je vytvořena pomocí knihovny Room a má jedinou entitu `SecurityIncident` s atributy závažnost, zdrojová a cílová IP adresa, signatura, časová značka a dalšími atributy reprezentující detailní informace o incidentu. Jelikož časové značky jsou uchovány ve třídě `AlertTimestamp`, která pracuje s formátem času používaným sondou, je pro konverzi datového typu na typ `Long` použita pomocná třída `TimestampConverter`.

Jelikož obrazovka seznamu bezpečnostních incidentů pracuje pouze s některými atributy bezpečnostních incidentů, je v aplikaci vytvořena pomocná třída `SecurityIncidentOverview`, která neobsahuje detailní informace o incidentu. Dotazy na databázi, které vyžadují jen část atributů tak vrací tento objekt. Tím je sníženo využití paměti a je urychleno dotazování na databázi.

SQL příkazy pro databázi jsou definovány uvnitř `SecurityIncidentDao` rozhraní. Mezi tyto příkazy patří příkaz pro smazání z databáze všech incidentů, které jsou starší než 30 dní, který je volán vždy při spuštění hlavní obrazovky aplikace:

```
@Query("DELETE FROM security_incident WHERE timestamp <
    (strftime('%s', datetime('now', '-30 days'))*1000)")
void deleteOlderThan30Days();
```

Příkaz pro získání počtu všech bezpečnostních incidentů v databázi:

```
@Query("SELECT COUNT(*) FROM security_incident")
LiveData<Integer> getNumberOfSecurityIncidents();
```

Příkaz pro získání seznamu základních informací všech bezpečnostních incidentů v databázi, seřazených dle nejnovějšího:

```
@Query("SELECT id ,severity, source_IP, destination_IP, signature,
    timestamp FROM security_incident ORDER BY timestamp desc")
DataSource.Factory<Integer, SecurityIncidentOverview>
    getSecurityIncidentsDesc();
```

Příkaz pro získání všech různých zdrojových IP adres incidentů v databázi.

```
@Query("SELECT DISTINCT source_IP FROM security_incident")
List<String> getDistinctSourceIPs();
```

Příkaz pro získání základních informací všech bezpečnostních incidentů

v databázi, které vyhovují zadanému filtru, seřazených dle nejnovějšího

```
@Query("SELECT id ,severity, source_IP, destination_IP, signature,
    timestamp FROM security_incident WHERE signature IN
    (:filteredSignatures) AND severity IN (:filteredSeverities) AND
    source_IP IN (:filteredSourceIPs) AND destination_IP IN
    (:filteredDestinationIPs) AND timestamp BETWEEN
    :filteredTimeFrom AND :filteredTimeTo ORDER BY timestamp desc")
DataSource.Factory<Integer, SecurityIncidentOverview>
    getFilteredSecurityIncidents(
List<Integer> filteredSeverities, List<String> filteredSignatures,
List<String> filteredSourceIPs, List<String> filteredDestinationIPs,
long filteredTimeFrom, long filteredTimeTo);
```

Příkaz pro získání konkrétního bezpečnostního incidentu v databázi dle jeho unikátního ID:

```
@Query("SELECT * FROM security_incident WHERE id = :id")
SecurityIncident getSecurityIncidentById(int id);
```

Všechny SQL příkazy pro databázi bezpečnostních incidentů jsou namapovány uvnitř `SecurityIncidentsDatabaseRepository` a následně uvnitř třídy `SecurityIncidentsViewModel`, přes kterou jsou volány třídami aktivit, které příslušná data zobrazují.

5.3.5 Aktualizace bezpečnostních incidentů

Aktualizace databáze bezpečnostních incidentů ze sondy probíhá uvnitř metod třídy `SecurityIncidentsUpdate`. Aktualizace probíhá mimo hlavní vlákno aplikace, aby nedocházelo k jeho blokování, a je opakována každých 5 sekund po skončení poslední aktualizace.

Podmínkami pro to, aby aktualizace mohla probíhat, jsou, že aplikace musí být k sondě připojena a autentizována a že byla zkontrolována existence souboru se záznamy incidentů na sondě a že čas na mobilním zařízení i na sondě je stejný.

Při každé aktualizaci je nejprve ze `SharedPreferences` získána časová značka poslední aktualizace bezpečnostních incidentů:

```
SharedPreferences sharedPref = context.getSharedPreferences(
context.getString(LAST_TIMESTAMP_FILE, Context.MODE_PRIVATE);
String lastTimestamp =
    sharedPref.getString(context.getString(TIMESTAMP_LABEL), null);
```

V případě, že se jedná o první aktualizaci a ve `SharedPreferences` tedy

nic uloženo není, je použita časová značka odpovídající času přesně 30 dní od daného okamžiku (v aplikaci jsou uchovávány pouze bezpečnostní incidenty za posledních 30 dní, aplikaci tedy stačí získat ze sondy incidenty za tuto dobu):

```
if (lastTimestamp == null) {
Calendar cal = Calendar.getInstance();
cal.add(Calendar.DATE, -30); // odecte 30 dni
lastTimestamp = AlertTimestamp(cal.getTime()).getAsString();
}
```

Následně je na sondu zaslán příkaz pro zjištění počtu nových bezpečnostních incidentů od časové značky poslední aktualizace. Záznamy bezpečnostních incidentů na sondě jsou uloženy v souboru */var/log/suricata/eve.json*.

```
String response = sshConnection.sendAndGet("sudo jq \'. |
    select((.event_type==\"alert\") and (.timestamp >
    \"LAST_TIMESTAMP\"))\' EVENT_LOG_FILE | jq -s length");
response = response.replace("\n", "");
int newAlertsNumber = Integer.parseInt(response);
```

Pokud ne sondě žádné nové bezpečnostní incidenty nejsou, je jako časová značka poslední aktualizace do `SharedPreferences` uložen čas odpovídající začátku dané aktualizace a tím také daná aktualizace končí. Pokud v souboru na sondě nové incidenty jsou, pokračuje v jejich získávání.

Při získávání velkého množství (tisíce) nových bezpečnostních incidentů ze sondy aktualizace trvá o něco déle. Může to nastat při první aktualizaci, případně pokud aktualizace delší dobu neprobíhala. Z tohoto důvodu aplikace rozděluje získání a uložení bezpečnostních incidentů do databáze na menší části o velikosti 500 záznamů, aby se uživateli zobrazila alespoň část incidentů. V případě, že počet nových bezpečnostních incidentů je ale vyšší než 2000, aktualizace není na části rozdělena a uživatel je upozorněn hláškou na obrazovce o tom, že aktualizace může trvat delší dobu. V takovém případě by totiž rozdělení na části celkovou aktualizaci zbytečně prodloužilo. Kód pro rozdělení aktualizace na části po 500 záznamech je následující:

```
final int windowSize = 500;
int iterations = (int) Math.ceil((double) newAlertsNumber /
    windowSize); // pocet iteraci pro ziskani vsech incidentu po
    castech o 500 zaznamech
for (int it = 0; it < iterations; it++) {
   int arrayStart = windowSize*it; // zacatek daneho okna - prvni
       zaznam
   int arrayEnd = (it != iterations - 1) ? (arrayStart + windowSize)
       : newAlertsNumber; // konec daneho okna - posledni zaznam

   // ziskani zasti bezpecnostnich incidentu ze sondy a jejich
```

```
        ulozeni do databaze v aplikaci
}
```

Pro získání nových bezpečnostních incidentů ze sondy je na ni poslán příkaz, který ze souboru se záznamy bezpečnostních incidentů vybere příslušnou část incidentů odpovídající danému oknu s 500 záznamy, kterou následně zkopíruje do jiného, nově vytvořeného dočasného souboru na sondě. Pro uložení dočasného souboru je v aplikaci použita cesta */home/soniot_app/new_alerts.json*.

```
sshConnection.send("sudo bash -c \"jq \'. |
    select((.event_type==\\\"alert\\\") and (.timestamp >
    \\\"LAST_TIMESTAMP\\\"))\' EVENT_LOG_FILE | jq -s
    .[arrayStart:arrayEnd] > PROBE_TEMPORARY_FILE\" 2>/dev/null");
```

Dočasný soubor na sondě, do kterého byly bezpečnostní incidenty zkopírovány, následně aplikace stáhne pomocí SFTP a na sondě jej smaže:

```
sshConnection.downloadSFTP(PROBE_TEMPORARY_FILE,
new File(context.getFilesDir() + "/" +
    APP_ALERT_FILENAME).getAbsolutePath()); // stazeni souboru
sshConnection.send("sudo rm PROBE_TEMPORARY_FILE"); // smazani
    souboru
```

Stažený soubor je pak přečten:

```
try (InputStream in = new FileInputStream(context.getFilesDir() +
    "/" + APP_ALERT_FILENAME)) {
   int size = in.available();
   byte[] buffer = new byte[size];
   int i = in.read(buffer);
   String json = new String(buffer, "UTF-8");
}
```

Z přečtených dat ve formátu JSON jsou následně získány jednotlivé bezpečnostní incidenty jako `SecurityIncident` objekty za pomocí knihovny Gson [105], která převádí Java objekty do jejich JSON reprezentace a naopak. Pomocná třída `SecurityIncidentDeserializer` mapuje jednotlivé JSON elementy na atributy třídy `SecurityIncident`.

```
GsonBuilder builder = new GsonBuilder();
builder.registerTypeAdapter(SecurityIncident.class, new
    SecurityIncidentDeserializer());
Gson gson = builder.create();
```

```
List<SecurityIncident> records = gson.fromJson(json, new
    TypeToken<ArrayList<SecurityIncident>>() {}.getType());
```

Získaný seznam bezpečnostních incidentů `SecurityIncident` je uložen do databáze bezpečnostních incidentů v aplikaci.

```
SecurityIncidentsViewModel vm = new
    SecurityIncidentsViewModel(context);
vm.addSecurityIncident(records);
```

Nakonec je aktualizována časová značka poslední aktualizace bezpečnostních incidentů ve `SharedPreferences`.

5.3.6 Filtrování bezpečnostních incidentů

Filtry i řazení jsou uloženy uvnitř `SharedPreferences`. Ukládání a získávání dat z nich spravuje třída `SecurityIncidentsFilterPreferences`, která jiným třídám aplikace poskytuje metody pro získání filtrovaných hodnot, pro získání počtu aplikovaných filtrů, pro uložení nových filtrů a pro zjištění, zda je aplikováno filtrování a limit. Do `SharedPreferences` je ukládán seznam hodnot parametru, které v seznamu má být zobrazen, popřípadě hodnota *null*, pokud parametr být filtrován nemá (důvod je ten, aby v případě přidání bezpečnostního incidentu s novou hodnotou nefiltrovaného parametru do databáze se i tento záznam v seznamu zobrazil). V případě ukládání časů a limitu jsou hodnoty pro filtry uloženy jako číselná hodnota. Časové značky jsou převedeny na unixový čas v milisekundách. Pokud tyto parametry filtrovány nejsou, je uložena defaultní hodnota, která je pro čas od rovna 0, pro čas do roven maximální hodnotě datového typu `Long` a pro limit je -1. Způsob řazení je uložen jako hodnota datového typu `Boolean`, která nabývá *true* v případě řazení dle času.

Při zobrazení seznamu v aktivitě seznamu bezpečnostních incidentů je vybrán SQL dotaz na databázi na základě zvoleného typu řazení a toho, zda jsou aplikovány filtry a limit. To je zjištěno pomocí metod třídy `SecurityIncidentsFilterPreferences`. Dotaz je volán prostřednictvím třídy `SecurityIncidentsViewModel`. Jako parametr je předáván seznam hodnot, které se v seznamu mají zobrazit, pro všechny filtry v případě aplikovaných filtrů a hodnota limitu v případě aplikovaného omezení. Navrácený seznam bezpečnostních incidentů je obalen objektem `PagedList`, který je obalen objektem `LiveData`. Objekt `LiveData` zajišťuje notifikaci v případě, že je do databáze uložen nový bezpečnostní incident, který odpovídá aplikovaným filtrům, na základě které je pak zobrazený seznam aktualizován. Objekt `PagedList` zajišťuje, že je v jednu chvíli načtena pouze část položek v seznamu. Díky objektu `PagedList` je v aktivitě v jednu chvíli načteno pouze 50 bezpečnostních incidentů, díky čemuž je seznam načten

rychle i ve chvíli, kdy obsahuje velké množství položek. Další položky se načítají při rolování seznamu.

5.3.7 Zjišťování zařízení v síti

Nové skenování je provedeno vždy při volání metody `onResume()` hlavní aktivity a aktivity seznamu zařízení v síti, popřípadě při spuštění nového skenu uživatelem v aktivitě seznamu zařízení v síti.

Pro zjištění zařízení v síti je použita knihovna `AndroidNetworkTools` [106], která umožňuje najít zařízení, která jsou ve stejné podsíti jako mobilní zařízení a která reagují na ping. Zařízení jsou nalezena pomocí metody `findDevices()` třídy `SubnetDevices` této knihovny.

Metoda `findeDevices()` jako parametr přijímá rozhraní `OnSubnetDeviceFound`, které v aplikaci implementuje třída `DeviceFinder`. Rozhraní předepisuje metodu `onDeviceFound()`, která je volána při nalezení nového zařízení v síti, a metodu `onFinished()`, která je volána po dokončení skenování.

Jelikož knihovna nezjišťuje MAC adresu mobilního zařízení, na kterém je aplikace spuštěna, třída `DeviceFinder` při vytvoření své instance jeho IP adresu a MAC adresu zjistí a při nalezení nového zařízení v síti kontroluje, zda se IP adresa nalezeného zařízení rovná IP adrese mobilního zařízení, a pokud ano, použije pro něj dříve zjištěnou MAC adresu.

Informace o nalezených zařízeních v síti jsou drženy v objektech `Device`, které mají jako atributy IP adresu a detail o zařízení, což je textový řetězec obsahující hostname, MAC adresu a výrobce zařízení, popřípadě jen část těchto dat, pokud se některá nepodařilo zjistit. Knihovna `AndroidNetworkTools` o nalezeném zařízení poskytuje jeho hostname, IP adresu a MAC adresu. Třída `DeviceFinder` navíc zjišťuje výrobce zařízení na základě jeho MAC adresy. Ten je zjištěn pomocí online MAC Address Lookup API [107], které je poskytováno zdarma a poskytuje nízkou latenci a podporu pro časté dotazování. Pro zaslání HTTP žádosti a získání odpovědi je v aplikaci použita knihovna OkHttp [108]:

```
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder()
   .url("https://api.maclookup.app/v1/macs/" + MAC_ADRESA +
      "/company_name")
   .build();
try (Response response = client.newCall(request).execute()) {
   String vendor = response.body().string();
} catch (IOException e) {
   e.printStackTrace();
}
```

Při každém nalezení zařízení v síti a zjištění výrobce je zařízení přidáno do seznamu zařízení probíhajícího skenování a je přes `Eventbus` poslána událost

`DevicesFoundEvent`, které je jako parametr předán seznam nalezených zařízení. Po dokončení skenování je přes `EventBus` poslána událost `DevicesScanCompletedEvent` a je aktualizován seznam nalezených zařízení ve třídě `DeviceScanner`. V aktivitách tak je sledováním těchto událostí aktualizováno UI vždy při nalezení nového zařízení.

Skenování v aplikaci spravuje třída `DeviceScanner`, která si uchovává i seznam zařízení nalezených v síti při posledním skenování. Pokud je zařízení připojeno k síti a skenování v dané chvíli již neprobíhá, spustí třída nové skenování v novém vlákně na pozadí při volání její metody `startScan()`, která jako návratovou hodnotu používá výčtový typ `DeviceScanState`, který rozeznává stavy, kdy zařízení není připojeno k síti, kdy skenování aktuálně probíhá a kdy skenování právě započalo:

```
public static synchronized DeviceScanState startScan() {
   if (!ProbeConnectionManager.checkConnection()) return
       DeviceScanState.NOT_CONNECTED;
   if (scanning) return DeviceScanState.SCAN_IN_PROGESS;
   scanning = true;
   Executors.newSingleThreadExecutor().execute(() ->
   SubnetDevices.fromLocalAddress().findDevices(new DeviceFinder()));
   return DeviceScanState.SCAN_STARTED;
}
```

Pro přehlednější zobrazení seznamu nalezených zařízení je seznam seřazen dle jejich IP adres. Jejich seřazení v aplikaci spravuje třída `DeviceListSort`.

Seřazení probíhá převedením IP adresy na číselnou hodnotu, které jsou pak mezi sebou porovnány. Adresa je rozdělena po bytech a první byte je zahozen, jelikož číslo sítě je pro všechny zařízení stejné. Číselná reprezentace zbylé části IP adresy je vytvořena posunem o 8 bitů a přičtením následujícího bytu operací *OR*:

```
String[] parts = ipAddress.split("\\.");
int ip = Integer.parseInt(parts[1]);
ip = (ip << 8) | Integer.parseInt(parts[2]);
ip = (ip << 8) | Integer.parseInt(parts[3]);
```

5.3.8 Databáze pro blokaci stránek

Databáze pro blokaci stránek je vytvořena pomocí knihovny Room a tvoří ji 4 entity - `Website` pro uchování webových stránek, `Category` pro uchování kategorií, `Profile` pro uchování profilů a `ProfileCategory`, která slouží jako vazební tabulka mezi entitami `Profile` a `Category` pro vytvoření M:N vazby a slouží pro přiřazení kategorií k profilům. Všechny entity mají jako svůj primární klíč id, které je automaticky generované při přidání nového záznamu. Entita `ProfileCategory` pak obsahuje 2 cizí klíče, které odkazují

na primární klíče entit `Profile` a `Category`. Entita `Website` obsahuje cizí klíč, který odkazuje na primární klíč entity `Category` pro vytvoření 1:N vazby, která přiřadí webovou stránku k určité kategorii.

5.3.9 Blokace stránek

Blokace stránek na sondě probíhá pomocí nástroje `iptables` [87] odesláním příkazu na sondu. Aplikace k tomu využívá následujících příkazů pro blokace a povolení stránek, včetně příkazů pro nastavení časového okna a dní, kdy stránka bude blokována, které používají profily se specifikovaným časovým oknem:

```
sudo iptables -I FORWARD 1 -p tcp -m string --string "WEBSITE_URL"
    --algo kmp --to 65535 -j REJECT --reject-with
    icmp-port-unreachable // blokace webove stranky
sudo iptables -D FORWARD -p tcp -m string --string "WEBSITE_URL"
    --algo kmp --to 65535 -j REJECT --reject-with
    icmp-port-unreachable // povoleni webove stranky
sudo iptables -I FORWARD 1 -p tcp -m string --string "WEBSITE_URL"
    --algo kmp --to 65535 -m time --timestart CAS_OD --timestop
    CAS_DO --weekdays VYCET_DNI -j REJECT --reject-with
    icmp-port-unreachable // blokace webove stranky s nastavenym
    casovym oknem a dny
sudo iptables -D FORWARD -p tcp -m string --string "WEBSITE_URL"
    --algo kmp --to 65535 -m time --timestart CAS_OD --timestop
    CAS_DO --weekdays VYCET_DNI -j REJECT --reject-with
    icmp-port-unreachable // povoleni webove stranky s nastavenym
    casovym oknem a dny (zruseni konkretni blokace)
```

Po každé změně `iptables` je poslán příkaz na jejich uložení, aby změny zůstaly aplikované i po restartování sondy. Pravidla pro `iptables` jsou na sondě ukládány ve */home/test/firewall/iptables.up.rules*.

```
sudo bash -c "iptables-save > IPTABLES_SAVE_LOCATION"
```

O odeslání správného příkazu pro blokaci či povolení webové stránky se v aplikaci starají metody třídy `WebsitesBlockManager`.

Časové okno je před odesláním příkazu na sondu v aplikaci převedeno na koordinovaný světový čas UTC (Coordinated Universal Time) uvnitř pomocné třídy `UniversalTimeConverter`. To s sebou ale přináší problém přetečení nastaveného časového okna uživatelem v aplikaci do předchozího či následujícího dne. Z tohoto důvodu je v aplikaci před odesláním příkazu na sondu zkontrolováno, zda některý z časů či oba časy nepřetekly do jiného dne, a případně pošle příkaz navíc. Pokud například blokace měla probíhat od 1:00 do 3:00 pro UTC+2:00, jsou poslány příkazy pro blokaci daných stránek

od 23:00 do 23:59:59 předchozího dne a od 00:00 do 1:00 daného dne. Posun dní v týdnu, ve kterých je profil aktivní, pak zajišťuje třída `ProfileDays`, která v aplikaci usnadňuje práci s výčtem dní v týdnu.

5.3.10 Zjišťování stavů profilů

Profily mohou být v daném čase buď aktivní, nebo neaktivní. Tyto stavy v aplikaci zjišťují metody třídy `StatesOfProfiles`. U profilů, které nemají specifikované časové okno, je jejich stav uchován v databázi. U profilů se specifikovaným časovým oknem je zjišťování jejich stavu složitější, neboť se mění v průběhu dne, a zajišťuje ho metoda `isTimeProfileActive()`, která zjišťuje aktuální čas a den a pokud je profil aktivní v daný den, porovnává aktuální čas s časovým oknem převedením na čas v milisekundách:

```
private static boolean isTimeProfileActive(Profile profile) {
   assert profile.getDays() != null;
   Calendar calendar = Calendar.getInstance();
   int dayNumber = ProfileDays.getDayNumber(calendar);
   if (profile.getDays().get(dayNumber)) {
      long now = calendar.getTimeInMillis();
      long from = getTimeInMillis(calendar, profile.getHourFrom(),
          profile.getMinuteFrom());
      long to = getTimeInMillis(calendar, profile.getHourTo(),
          profile.getMinuteTo());
      return from <= now && now < to;
   }
   return false;
}
```

Aby UI aplikace, které zobrazuje profily, nebo stránky blokované profily se specifikovaným časovým oknem, mohlo být aktualizováno v případě, že v čase strávené na obrazovce se stav profilu s časovým oknem změní, jsou při spuštění takových aktivit vytvořeny objekty `ProfileDateTimeAction`, které udávají čas změny a stav, na který daný profil přejde. Seznam těchto objektů je aktivitám předán voláním metody `getAllProfileDateTimeActions()` třídy `StatesOfProfiles`. Metoda zjišťuje všechny změny profilů s časovým oknem, které nastanou v průběhu následujících hodin. V aktivitách je pak pro první položku v seznamu vytvořen `Timer`, který v čase změny stavu daného profilu aktualizuje UI aktivity a pokud seznam není dále prázdný, nastaví `Timer` pro další položku seznamu.

5.3.11 Zjišťování stránek blokovaných sondou

Při spuštění aktivit, které zobrazují webové stránky a jejich stavy, zda jsou na sondě blokovány, či povoleny, jsou porovnána pravidla blokací na sondě s daty v databázi. Důvod je takový, že k sondě je v budoucnosti

v plánu vytvořit webové rozhraní, které bude taktéž umožňovat stránky blokovat. Pokud tedy stránka bude blokována či povolena mimo aplikaci, aplikace bude schopna toto detekovat a upravit stav dané stránky v databázi, případně stránku do aplikace přidat. Porovnávány jsou pouze blokace bez specifikovaného časového okna.

Zjištění stavu stránek na sondě a případné změny v databázi provádí v aplikaci metody třídy `WebsiteStates`. Pravidla `iptables` aplikovaná na sondě jsou zjištěna pomocí následujícího příkazu:

```
sudo iptables -S
```

Následně jsou vybrány jen pravidla začínající `"-A FORWARD -p tcp -m string --string"`, která nemají specifikované časové okno. Z vybraných pravidel je pak ve třídě `WebsiteStates` získán seznam sondou blokovaných stránek a je spočítáno, kolikrát je každá stránka blokována (pokud je stránka blokována profilem, může být na sondě blokována vícekrát). Stránky a počty blokací jsou porovnány se záznamy v databázi aplikace. Pokud je stránka na sondě blokována vícekrát než v databázi, je změněn její stav v databázi na blokovanou, a naopak je změněn stav na povolenou, pokud je vícekrát blokována v databázi. Pokud se webová stránka blokovaná na sondě v databázi aplikace nenachází, je do ní přidána v blokovaném stavu.

Kapitola 6

Testování

Aplikace byla manuálně otestována společně s oběma verzemi sondy SonIoT, Home a Industry. Pro testování byly použity mobilní telefony Nubia N2 a Samsung Galaxy A50, jejichž parametry jsou uvedeny v tabulce 6.1. Zprávu z tetování je možné nalézt v příloze A tohoto dokumentu. Ve zprávě jsou zdokumentovány provedené testy a nalezené problémy včetně jejich řešení.

Tabulka 6.1: Parametry testovaných mobilních zařízení [109],[110]

Mobilní telefon	Nubia N2	Samsung Galaxy A50
Verze OS Android	6.0 (Marshmallow)	9.0 (Pie)
Rozlišení displeje	1280 x 720	2340 x 1080
RAM	4 GB	4 GB
Počet jader procesoru	8	8
Frekvence procesoru	1,5 GHz	2,3 GHz

6.1 Seznam nalezených problémů dle závažnosti

V tabulce 6.2 je zobrazen seznam všech problémů, které byly při testování nalezeny. Seznam je řazen dle závažnosti problému a následně dle jeho značky. Ke každému problému je v tabulce uvedeno, při testování které verze aplikace byl nalezen. Sloupec *v1* značí první verzi aplikace a sloupec *v2* druhou verzi aplikace.

Při testování bylo nalezeno celkem 27 problémů. Z toho 4 problémy byly nalezeny ve druhé verzi aplikace, kde byl 1 problém nalezen při testování aktualizace bezpečnostních incidentů, který bude řešen přímo na sondě, další 2 problémy byly nalezeny na obrazovkách blokace webových stránek a poslední problém se týkal detekce přerušení spojení se sondou. Tyto problémy byly následně vyřešeny. Řešení jsou popsána v tabulkách jednotlivých problémů ve zprávě z testování v příloze A.

Tabulka 6.2: Seznam nalezených problémů dle závažnosti

Značka	Problém	Závažnost	v1	v2
S1	Aktualizace bezpečnostních incidentů neběží na pozadí.	1	•	
S3	Přerušení aktualizace bezpečnostních incidentů v některých případech aplikaci znemožní získání záznamu.	1	•	
S9	Časově náročná aktualizace bezpečnostních incidentů v případě velmi velkého souboru na sondě znemožní aktualizaci.	1		•
D1	Spuštění operace zjišťování zařízení v síti vícekrát zároveň způsobí na obrazovce seznamu zařízení v síti pád aplikace.	1	•	
C3	Smazání kategorie, která je ovlivněna profilem, který aktuálně není blokován, způsobí pád aplikace. Stránky uvnitř kategorie jsou z kategorie smazány, ale v případě profilu s nastaveným časovým omezením u nich blokace stránek profilem není odstraněna.	1	•	
W2	Uložení upravené stránky se změněnou adresou způsobí v případě blokované stránky a nepřipojené sondy pád aplikace.	1	•	
W3	Při uložení upravené stránky se změněnou kategorií, kdy nová kategorie je blokována profilem, není při připojené sondě upravená stránka zablokována profilem, který blokuje novou kategorii.	1	•	
W4	Je umožněno uložení upravené stránky se změněnou kategorií, kdy nová kategorie je blokována profilem, i když je sonda odpojena.	1	•	
W6	Smazání blokované stránky při odpojené sondě způsobí pád aplikace.	1	•	
H2	Zjišťování zařízení v síti blokuje jiné operace na pozadí.	2	•	
S2	Stažení souboru s novými bezpečnostními incidenty před ukončením zápisu sondou do něj.	2	•	

S5	Zobrazení filtrovaného seznamu bezpečnostních incidentů, i když nemá být filtrován.	2	•	
S6	Zobrazení nefiltrovaného seznamu bezpečnostních incidentů, ačkoliv má být filtrován.	2	•	
S7	Zobrazení filtrovaného seznamu bezpečnostních incidentů, i když nemá být filtrován v případě aktualizace v jiné části aplikace.	2	•	
W5	Při změně adresy blokované webové stránky není původní adresa na sondě povolena.	2		•
P2	Při změně času na zařízení neodpovídají časy blokací profily s časovým oknem zobrazovaným časům.	2		•
H1	Při vypnutí sondy není detekováno přerušení spojení.	3	•	•
S4	Přehozená priorita závažností 1-3 bezpečnostních incidentů.	3	•	
S8	Časově náročné zpracování aktualizace velkého počtu bezpečnostních incidentů.	3	•	
S10	Prodleva při zobrazení velkého množství záznamů v seznamu bezpečnostních incidentů.	3	•	
F1	V případě aktualizace bezpečnostních incidentů na obrazovce filtrování se v některých případech nezobrazují aktuální dostupné hodnoty pro filtry.	3	•	
B1	V případě změny stavu profilu s časovým omezením na aktivní/neaktivní v průběhu zobrazování hlavní obrazovky blokace internetových stránek není aktualizován počet blokovaných profilů a stránek.	3	•	
B2	Zacyklení tlačítka zpět v části blokování webových stránek.	3	•	
C1	Není aktualizován počet blokovaných stránek na obrazovce detailu kategorie v případě změny blokací stránek přímo na obrazovce.	3	•	

C2	Není aktualizován počet blokovaných stránek na obrazovce seznamu kategorií při návratu z kategorie, u níž byl počet změněn.	3	•	
W1	V případě blokace povolené stránky několika profily najednou je stránka v seznamu zobrazena vícekrát.	3	•	
P1	Pokud je uživatel na obrazovce seznamu profilů a je v daném čase aktivován/deaktivován profil s nastaveným časovým omezením, není aktualizováno rozdělení seznamu na aktivní a neaktivní profily.	3	•	

Kapitola 7

Závěr

V rámci této diplomové práce byla rozšířena a otestována aplikace pro platformu Android, která komunikuje s bezpečnostní sondou SonIoT, vytváří databázi bezpečnostních incidentů detekovaných sondou, poskytuje rozhraní pro správu blokace webových stránek v síti a umožňuje provést skenování sítě pro nalezení připojených zařízení. Minimální verze OS Android, na které lze aplikaci spustit je verze 6.0 Marshmallow. Je tak podporováno téměř 85% celosvětově používaných zařízení s OS Android.

V první kapitole jsou popsány základní principy návrhu aplikace pro platformu Android včetně přínosu jejích novějších verzí a popisu architektury a zabezpečení zařízení s OS Android a jejich dat.

Ve druhé kapitole je popsána sonda SonIoT, která pro detekci hrozeb v síti využívá open-source nástroj Suricata. Sonda může být v síti nakonfigurována jako IDS a IPS. Kapitola obsahuje i obecný popis těchto systémů.

Zabezpečená komunikace mezi aplikací a sondou probíhá pomocí protokolu SSH, který je popsán ve třetí kapitole. Kapitola také krátce popisuje RSA klíče, které jsou v aplikaci použity pro autentizaci sondě, a AES klíče, který je v aplikace použit pro šifrování soukromého RSA klíče na mobilním zařízení.

V páté kapitole je popsána samotná aplikace SonIoT, jejíž první verze byla v rámci diplomové práce rozšířena a vylepšena, především byla přidána autentizace pomocí certifikátu a některá vylepšení uživatelského rozhraní. Popis jejího uživatelského rozhraní je rozdělen na 4 části - Home, Security, Devices a Blocking, ve kterých jsou popsány jednotlivé obrazovky aplikace. Kapitola dále obsahuje popis implementace jednotlivých funkcí aplikace, včetně použitých knihoven.

Poslední, šestá kapitola se zabývá manuálním testováním obou verzí aplikace. V kapitole jsou popsána testovaná zařízení i problémy, které byly při testování nalezeny. Ve druhé verzi aplikace byly při testování nalezeny 4 problémy, které byly v aplikaci vyřešeny.

Vyvinutá aplikace uživateli poskytuje nástroj pro kontrolu bezpečnostních incidentů detekovaných sondou a pro snadnou blokaci webových stránek s možností nastavení časových intervalů blokací a poskytuje přehled

o zařízeních připojených v síti. Použitím autentizace pomocí certifikátu je zajištěno vyšší zabezpečení komunikace mezi zařízeními a uživatelsky přívětivé navázání spojení se sondou stisknutím jediného tlačítka.

Literatura

[1] Platform Architecture. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/guide/platform`

[2] Operating System Market Share Worldwide: Apr 2019 - Apr 2020. *StatCounter Global Stats* [online]. [cit. 2020-05-03]. Dostupné z: `https://gs.statcounter.com/os-market-share`

[3] Mobile Operating System Market Share Worldwide: Apr 2019 - Apr 2020. *https://gs.statcounter.com/os-market-share/mobile/worldwide*

[4] Mobile Operating System Market Share Czech Republic: 2009 - 2020. *StatCounter Global Stats* [online]. [cit. 2020-05-03]. Dostupné z: `https://gs.statcounter.com/os-market-share/mobile/czech-republic/#yearly-2009-2020`

[5] JURKOVÁ, Michaela. *Mobilní aplikace pro ovládání laboratorního napájecího zdroje* [online]. Praha, 2017 [cit. 2020-05-02]. Dostupné z: `https://dspace.cvut.cz/bitstream/handle/10467/68566/F3-BP-2017-Jurkova-Michaela-Bakalarska%20prace.pdf`. Bakalářská práce. FEL ČVUT v Praze.

[6] Documentation. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/docs`

[7] Android Lollipop. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/about/versions/lollipop`

[8] Android 6.0 APIs. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/about/versions/marshmallow/android-6.0`

[9] Android 6.0 Changes. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/about/versions/marshmallow/android-6.0-changes`

[10] Android 7.0 for Developers. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/about/versions/nougat/android-7.0`

[11] Android 8.0 Features and APIs. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/about/versions/oreo/android-8.0`

[12] Android 8.1 Features and APIs. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/about/versions/oreo/android-8.0-changes`

[13] Android 9 features and APIs. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/about/versions/pie/android-9.0`

[14] Behavior changes: all apps. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/about/versions/10/behavior-changes-all`

[15] Android 11 Developer Preview. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/preview/overview`

[16] Features and APIs Overview. *Android Developers* [online]. [cit. 2020-05-03]. Dostupné z: `https://developer.android.com/preview/features`

[17] Mobile & Tablet Android Version Market Share Worldwide: Apr 2019 - Apr 2020. *StatCounter Global Stats* [online]. [cit. 2020-05-03]. Dostupné z: `https://gs.statcounter.com/os-version-market-share/android/mobile-tablet/worldwide`

[18] Mobile & Tablet Android Version Market Share Czech Republic: Apr 2019 - Apr 2020. *StatCounter Global Stats* [online]. [cit. 2020-05-03]. Dostupné z: `https://gs.statcounter.com/android-version-market-share/mobile-tablet/czech-republic`

[19] TAMMA, Rohit, Oleg SKULKIN, Heather MAHALIK a Satish BOMMISETTY. *Practical Mobile Forensics*. 3rd ed. Birmingham: Packt Publishing, 2018. ISBN 978-1-78883-919-8.

[20] Modular Kernel Requirements. *Android Open Source Project* [online]. [cit. 2020-05-04]. Dostupné z: `https://source.android.com/devices/architecture/kernel/modular-kernels`

[21] Legacy HALs. *Android Open Source Project* [online]. [cit. 2020-05-04]. Dostupné z: `https://source.android.com/devices/architecture/hal`

[22] HAL Types. *Android Open Source Project* [online]. [cit. 2020-05-04]. Dostupné z: `https://source.android.com/devices/architecture/hal-types`

[23] Android Architecture. *Android Open Source Project* [online]. [cit. 2020-05-04]. Dostupné z: `https://source.android.com/devices/architecture/`

[24] LACKO, Luboslav. *Vývoj aplikací pro Android.* Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[25] WEI, Jason. *Android Database Programming.* Birmingham: Packt Publishing, 2012. ISBN 978-1-84951-812-3.

[26] Save key-value data. *Android Developers* [online]. [cit. 2020-05-01]. Dostupné z: `https://developer.android.com/training/data-storage/shared-preferences`

[27] Android.database.sqlite. *Android Developers* [online]. [cit. 2020-05-01]. Dostupné z: `https://developer.android.com/reference/android/database/sqlite/package-summary.html`

[28] VOGEL, Lars. Using the Room framework as SQL object mapping library. *Vogella* [online]. 7.9.2017 [cit. 2020-05-01]. Dostupné z: `https://www.vogella.com/tutorials/AndroidSQLite/article.html`

[29] ITL Education Solutions Limited. *Introduction To Database System.* Pearson, 2014. ISBN 978-81-317-3192-5.

[30] HAVLÍČEK, Zdeněk. *Databázové systémy: Paradox.* Praha: Grada, 1992. Educa '99. ISBN 80-85424-97-5.

[31] Relational Databases: Foreign Keys. *MariaDB* [online]. [cit. 2020-05-01]. Dostupné z: `https://mariadb.com/kb/en/library/relational-databases-foreign-keys/`

[32] Types of Table Relationships (Visual Database Tools). *Microsoft Docs* [online]. [cit. 2020-05-01]. Dostupné z: `https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms190651(v=sql.105)`

[33] POKORNÝ, Jaroslav. *Databázové systémy a jejich použití v informačních systémech.* Praha: Academia, 1992. ISBN 80-200-0177-8.

[34] SQLite As An Application File Format. *SQLite* [online]. [cit. 2020-05-01]. Dostupné z: `https://www.sqlite.org/appfileformat.html`

[35] SQLite Copyright. *SQLite* [online]. [cit. 2020-05-01]. Dostupné z: `https://www.sqlite.org/copyright.html`

[36] Appropriate Uses For SQLite. *SQLite* [online]. [cit. 2020-05-01]. Dostupné z: `https://www.sqlite.org/whentouse.html`

[37] SQL Features That SQLite Does Not Implement. *SQLite* [online]. [cit. 2020-05-01]. Dostupné z: `http://www.sqlite.org/omitted.html`

[38] SQLite Index, Trigger & View with Example. *Guru99* [online]. [cit. 2020-05-01]. Dostupné z: https://www.guru99.com/sqlite-view-index-trigger.html

[39] Database File Format. *SQLite* [online]. [cit. 2020-05-01]. Dostupné z: https://www.sqlite.org/fileformat2.html

[40] Frequently Asked Questions. *SQLite* [online]. [cit. 2020-05-01]. Dostupné z: https://www.sqlite.org/faq.html

[41] Write-Ahead Logging. *SQLite* [online]. [cit. 2020-05-01]. Dostupné z: https://www.sqlite.org/wal.html

[42] SQL (Structured Query Language): Definition. *SQL Server: Covering today's SQL Server topics* [online]. [cit. 2020-05-01]. Dostupné z: https://searchsqlserver.techtarget.com/definition/SQL

[43] SQL Commands. *Beginner SQL Tutorial* [online]. [cit. 2020-05-01]. Dostupné z: http://beginner-sql-tutorial.com/sql-commands.htm

[44] SQL As Understood By SQLite: SELECT. *SQLite* [online]. [cit. 2020-05-01]. Dostupné z: http://www.sqlite.org/lang_select.html

[45] Android Room with a View: Java. *Google Codelabs* [online]. [cit. 2020-05-01]. Dostupné z: https://codelabs.developers.google.com/codelabs/android-room-with-a-view

[46] Save data in a local database using Room. *Android Developers* [online]. [cit. 2020-05-01]. Dostupné z: https://developer.android.com/training/data-storage/room

[47] Defining data using Room entities. *Android Developers* [online]. [cit. 2020-05-01]. Dostupné z: https://developer.android.com/training/data-storage/room/defining-data.html

[48] Define relationships between objects. *Android Developers* [online]. [cit. 2020-05-01]. Dostupné z: https://developer.android.com/training/data-storage/room/relationships

[49] Referencing complex data using Room. *Android Developers* [online]. [cit. 2020-05-01]. Dostupné z: https://developer.android.com/training/data-storage/room/referencing-data.html

[50] Accessing data using Room DAOs. *Android Developers* [online]. [cit. 2020-05-01]. Dostupné z: https://developer.android.com/training/data-storage/room/accessing-data.html

[51] Migrating Room databases. *Android Developers* [online]. [cit. 2020-05-01]. Dostupné z: https://developer.android.com/training/data-storage/room/migrating-db-versions.html

[52] Android Architecture Components. *Android Developers* [online]. [cit. 2020-05-02]. Dostupné z: `https://developer.android.com/topic/libraries/architecture`

[53] Handling Lifecycles with Lifecycle-Aware Components. *Android Developers* [online]. [cit. 2020-05-02]. Dostupné z: `https://developer.android.com/topic/libraries/architecture/lifecycle`

[54] LiveData Overview. *Android Developers* [online]. [cit. 2020-05-02]. Dostupné z: `https://developer.android.com/topic/libraries/architecture/livedata`

[55] ViewModel. *Android Developers* [online]. [cit. 2020-05-02]. Dostupné z: `https://developer.android.com/topic/libraries/architecture/viewmodel`

[56] Paging library overview. *Android Developers* [online]. [cit. 2020-05-02]. Dostupné z: `https://developer.android.com/topic/libraries/architecture/paging`

[57] Secure an Android Device. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security#google-security-services`

[58] System and kernel security. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security/overview/kernel-security`

[59] https://source.android.com/security/verifiedboot. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security/verifiedboot`

[60] Security Updates and Resources. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security/overview/updates-resources`

[61] Application security. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security/overview/app-security`

[62] Security Enhancements in Android 6.0. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security/enhancements/enhancements60`

[63] https://source.android.com/security/app-sandbox. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security/app-sandbox`

[64] Security-Enhanced Linux in Android. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security/selinux`

[65] Encryption. *Android Open Source Project* [online]. [cit. 2020-05-20]. Dostupné z: `https://source.android.com/security/encryption`

[66] Security tips. *Android Developers* [online]. [cit. 2020-05-20]. Dostupné z: `https://developer.android.com/training/articles/security-tips`

[67] Android keystore system. *Android Developers* [online]. [cit. 2020-05-20]. Dostupné z: `https://developer.android.com/training/articles/keystore`

[68] App security improvement program. *Android Developers* [online]. [cit. 2020-05-20]. Dostupné z: `https://developer.android.com/google/play/asi`

[69] MEJZROVÁ, Lenka a kolektiv. *Projekt: Vývoj sondy pro preventivní ochranu IoT zařízení před pokusy o jejich převzetí* [online]. 2019 [cit. 2020-05-01]. Interní projektová dokumentace. FEL ČVUT v Praze.

[70] *SonIoT: Bezpečnostní sonda* [online]. [cit. 2020-05-12]. Dostupné z: `https://soniot.fel.cvut.cz/`

[71] LINDSTROM, Pete. *Intrusion Prevention Systems (IPS): Next Generation Firewalls* [online]. 2004 [cit. 2020-04-30]. Dostupné z: `http://www.forum-intrusion.com/Spire_IPS_Whitepaper.pdf`. Spire Security, LLC.

[72] SCARFONE, Karen a Peter MELL. *NIST Special Publication 800-94: Guide to Intrusion Detection and Prevention Systems (IDPS)* [online]. 2007 [cit. 2020-04-30]. Dostupné z: `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf`. National Institute of Standards and Technology.

[73] Suricata. *Suricata: Open Source IDS / IPS / NSM engine* [online]. [cit. 2020-04-28]. Dostupné z: `https://suricata-ids.org/`

[74] About. *Suricata: Open Source IDS / IPS / NSM engine* [online]. [cit. 2020-04-28]. Dostupné z: `https://suricata-ids.org/about/`

[75] POLÁČEK, Radek. Sonda Suricata: IDS/IPS na systému Ubuntu 16.04 LTS (článek pro Sdělovací techniku). *Linux Services* [online]. [cit. 2020-04-28]. Dostupné z: `https://www.linuxservices.cz/sonda_suricata`

[76] All features: Complete list of Suricata Features. *Suricata: Open Source IDS / IPS / NSM engine* [online]. [cit. 2020-04-28]. Dostupné z: `https://suricata-ids.org/features/all-features/`

[77] What is Suricata? Intro to a Best of Breed Open Source IDS and IPS. *Bricata: Network Security Solutions* [online]. 4. 6. 2019 [cit. 2020-04-28]. Dostupné z: `https://bricata.com/blog/what-is-suricata-ids/`

[78] Suricata Rules: Rules Format. *Suricata User Guide: Suricata 5.0.2 documentation* [online]. [cit. 2020-04-28]. Dostupné z: https://suricata.readthedocs.io/en/suricata-5.0.2/rules/intro.html

[79] Quickstart guide: Eve JSON Output. *Suricata User Guide: Suricata 5.0.2 documentation* [online]. [cit. 2020-04-28]. Dostupné z: https://suricata.readthedocs.io/en/suricata-5.0.2/quickstart.html

[80] Configuration: Suricata.yaml. *Suricata User Guide: Suricata 5.0.2 documentation* [online]. [cit. 2020-04-28]. Dostupné z: https://suricata.readthedocs.io/en/suricata-5.0.2/configuration/suricata-yaml.html

[81] Output: Suricata.yaml. *Suricata User Guide: Suricata 5.0.2 documentation* [online]. [cit. 2020-04-28]. Dostupné z: https://suricata.readthedocs.io/en/suricata-5.0.2/output/eve/eve-json-output.html

[82] Snorby / snorby: Ruby On Rails Application For Network Security Monitoring. *GitHub* [online]. [cit. 2020-04-28]. Dostupné z: https://github.com/Snorby/snorby/

[83] BASE. *SourceForge: Download, Develop and Publish Free Open Source* Software [online]. [cit. 2020-04-28]. Dostupné z: https://sourceforge.net/p/secureideas/wiki/Home/

[84] Squert. *Security Onion Documentation: Security Onion 16.04.6.5 documentation* [online]. [cit. 2020-04-28]. Dostupné z: https://securityonion.readthedocs.io/en/latest/squert.html

[85] Jasonish / evebox: Web Based Event Viewer (GUI) for Suricata EVE Events in Elastic Search. *GitHub* [online]. [cit. 2020-04-28]. Dostupné z: https://github.com/jasonish/evebox/

[86] Snorby. *Aldeid* [online]. 23. 11. 2013 [cit. 2020-04-30]. Dostupné z: https://www.aldeid.com/wiki/Snorby

[87] Iptables: Linux manual page. *Linux man pages online* [online]. [cit. 2020-04-28]. Dostupné z: http://man7.org/linux/man-pages/man8/iptables.8.html

[88] FORSHAW, James. *Attacking networking protocols: A Hacker's Guide to Capture, Analysis and Exploitation*. San Francisco: no starch press, 2018, s. 145-179. ISBN 978-1-59327-750-5.

[89] SCHMIDT, Markus. Elements of an SSH session. *EmTec ZOC: SSH Client and Terminal Emulator* [online]. EmTec Innovative Software [cit. 2020-05-21]. Dostupné z: https://www.emtec.com/zoc/whitepapers/SSH_Session_Structure.pdf

[90] GIORDANI, Leonardo. Public key cryptography: RSA keys. *The Digital Cat* [online]. 2018 [cit. 2020-05-21]. Dostupné z: `https://www.thedigitalcatonline.com/blog/2018/04/25/rsa-keys/`

[91] DWIVEDI, Himanshu. *Implementing SSH: Strategies for Optimizing the Secure Shell*. Indianapolis, Indiana: Wiley Publishing, 2004. ISBN 0-471-45880-5.

[92] GORALSKI, Walter. *The Illustrated Network: How TCP/IP Works in a Modern Network*. Burlington: Elsevier, 2009, s. 633-659. ISBN 978-0-12-374541-5.

[93] MALONE,
wrong. *Smallstep* [online]. 2019 [cit. 2020-05-21]. Dostupné z: `https://smallstep.com/blog/use-ssh-certificates/`

[94] Android Version Distribution statistics will now only be available in Android Studio. *XDA-Developers Android Forums* [online]. [cit. 2020-05-18]. Dostupné z: `https://www.xda-developers.com/android-version-distribution-statistics-android-studio/`

[95] PagedListAdapter. *Android Developers* [online]. [cit. 2020-05-14]. Dostupné z: `https://developer.android.com/reference/androidx/paging/PagedListAdapter`

[96] RotateAnimation. *Android Developers* [online]. [cit. 2020-05-14]. Dostupné z: `https://developer.android.com/reference/android/view/animation/RotateAnimation`

[97] AAkira / ExpandableLayout: An android library that brings the expandable layout with various animation. In: *GitHub* [online]. [cit. 2020-05-12]. Dostupné z: `https://github.com/AAkira/ExpandableLayout`

[98] DatePickerDialog. *Android Developers* [online]. [cit. 2020-05-14]. Dostupné z: `https://developer.android.com/reference/android/app/DatePickerDialog`

[99] TimePickerDialog. *Android Developers* [online]. [cit. 2020-05-14]. Dostupné z: `https://developer.android.com/reference/android/app/TimePickerDialog`

[100] ListAdapter. *Android Developers* [online]. [cit. 2020-05-14]. Dostupné z: `https://developer.android.com/reference/androidx/recyclerview/widget/ListAdapter`

[101] ApmeM / android-flowlayout: Linear layout, that wrap its content to the next line if there is no space in the current line. In: *GitHub* [online]. [cit. 2020-05-16]. Dostupné z: `https://github.com/ApmeM/android-flowlayout`

[102] Greenrobot / EventBus: Event bus for Android and Java that simplifies communication between Activities, Fragments, Threads, Services, etc. Less code, better quality. In: *GitHub* [online]. [cit. 2020-05-07]. Dostupné z: `https://github.com/greenrobot/EventBus`

[103] Hierynomus / sshj: ssh, scp and sftp for java. In: *GitHub* [online]. [cit. 2020-05-07]. Dostupné z: `https://github.com/hierynomus/sshj`

[104] JSch: Java Secure Channel. *JCraft* [online]. [cit. 2020-05-22]. Dostupné z: `http://www.jcraft.com/jsch/`

[105] Google / Gson: A Java serialization/deserialization library to convert Java Objects into JSON and back. In: *GitHub* [online]. [cit. 2020-05-11]. Dostupné z: `https://github.com/google/gson`

[106] Stealthcopter / AndroidNetworkTools: Set of useful android network tools. In: *GitHub* [online]. [cit. 2020-05-14]. Dostupné z: `https://github.com/stealthcopter/AndroidNetworkTools`

[107] *MAC Address Lookup: Vendor/Manufacturer Search* [online]. 2018 [cit. 2020-05-14]. Dostupné z: `https://maclookup.app/`

[108] OkHttp. *Square Open Source* [online]. 2019 [cit. 2020-05-15]. Dostupné z: `https://square.github.io/okhttp/`

[109] Nubia N2: 4 GB / 64 GB, Dual SIM, zlatá. *Mall.cz* [online]. [cit. 2020-05-19]. Dostupné z: `https://www.mall.cz/mobilni-telefony/nubia-n2-4-gb-64-gb-dual-sim-zlata`

[110] Samsung Galaxy A50 Dual SIM černá. *Alza.cz* [online]. [cit. 2020-05-19]. Dostupné z: `https://www.alza.cz/samsung-galaxy-a50-dual-sim`

7.1 Seznam zkratek

AES ..... Advanced Encryption Standard

ANSSI ..... Agence Nationale de la Securité des Systémes d'Information

AOT ..... Ahead-Of-Time

API ..... Application Programming Interface

ART ..... Android Runtime

ASN ..... Abstract Syntax Notation

BASE ..... Basic Analysis and Security Engine

CBC ..... Cipher Block Chaining

DAO ..... Data Access Object

DES ..... Data Encryption Standard

DVM ..... Dalvik Virtual Machine

ECB ..... Electronic Code Book

FTP ..... File Transfer Protocol

GCM ..... Galois Counter Mode

GPS ..... Global Positioning System

HAL ..... Hardware Abstraction Layer

HIDL ..... HAL Interface Definition Language

HTTP ..... Hypertext Transfer Protocol

IDS ..... Intrusion Detection System

IP ..... Internet Protocol

IPS ..... Intrusion Prevention System

JIT ..... Just-In-Time

JSON ..... JavaScript Object Notation

JVM ..... Java Virtual Machine

MAC ..... Media Access Control

OISF ..... Open Information Security Foundation

OS ..... Operating System

PEM ..... Privacy-enhanced Electronic Mail

PKCS ..... Public Key Cryptographic Standards

POJO ..... Plain Old Java Object

RAM ..... Random Access Memory

RCP ..... Remote Copy Protocol

RSA ..... Rivest-Shamir-Adleman

RSH ..... Remote Shell

SCP ..... Secure Copy Protocol

SE ..... Secure Element

SFTP ..... SSH File Transfer Protocol

SIM ..... Subscriber Identity Module

SQL ..... Structured Query Language

SSH ..... Secure Shell

TCP ..... Transmission Control Protocol

TEE ..... Trusted Execution Environment

UDP ..... User Datagram Protocol

UI ..... User Interface

URL ..... Uniform Resource Locator

USB ..... Universal Serial Bus

UTC ..... Coordinated Universal Time

WAL ..... Write-Ahead Log

XML ..... Extensible Markup Language

Přílohy

Příloha A

Zpráva z testování

A.1 Rozdělení testování:

Otestovány byly obě verze aplikace. První testovaná verze aplikace je verze, která byla výstupem projektu, který předcházel této diplomové práci. Druhá verze aplikace byla otestována po následných úpravách kódu a doplnění funkcí otestované první verze aplikace.

Testování je rozděleno podle jednotlivých obrazovek aplikace a jejich funkcí. U každé testované části je popsáno, co bylo otestováno a jak se aplikace chovala. Testy i chování bylo pro mnoho testů společné pro obě verze aplikace, proto jsou v zápisu z testování uvedeny společně. Části, které jsou označeny jednou hvězdičkou (*), přísluší pouze testování první verze aplikace a části označené dvěma hvězdičkami (**) se týkají pouze druhé verze aplikace. V textu je tím odlišeno, že některý test byl proveden pouze u jedné verze, a také je tak rozlišeno rozdílné chování u provedeného testu. Text, který není označen hvězdičkami, přísluší testování oběma verzí aplikace.

A.2 Legenda závažnosti problémů:

Každému nalezenému problému byla při testování přiřazena závažnost, která je klasifikována následujícím způsobem:

- 1 - na problému závisí funkčnost aplikace
- 2 - problém omezuje funkčnost aplikace
- 3 - problém zásadně neomezuje funkčnost aplikace

A.3 Obrazovka nastavení komunikace

Bylo otestováno:

- Vyplňování textových polí

- Funkčnost tlačítka připojení:
  - Bez připojení k síti Wi-Fi** - *vypíše chybovou hlášku*
  - Při zadání neplatných dat - *vypíše chybovou hlášku a v případě autentizace pomocí certifikátu** či veřejného klíče* zobrazí dialog pro nové připojení*
  - Při zadání platných dat, pokud nebyl vytvořen klíčový pár a certifikát pro autentizaci - *připojí se pomocí hesla, vytvoří klíčový pár a veřejný klíč předá sondě*, ve druhé verzi aplikace ze sondy předáním veřejného klíče získá podepsaný certifikát, který si aplikace uloží***
  - Při zadání platných dat, pokud neexistuje soubor pro uložení veřejného klíče na sondě - *po připojení pomocí hesla vytvoří na sondě tento soubor vytvoří*, pro autentizaci certifikátem se neuvažuje***
  - Při zadání platných dat, pokud existuje klíčový pár* a certifikát na mobilním zařízení, který ale není podepsaný certifikační autoritou na sondě** - *autentizace pomocí certifikátu je neúspěšná, zobrazí dialog s upozorněním a možností vytvořit nový klíčový pár, který pokud je vybrán, zobrazí pole pro zadání hesla, následně připojení pomocí hesla, vytvoření nového klíčového páru, získání certifikátu a jeho uložení*
  - Při zadání platných dat, pokud existuje klíčový pár a certifikát** v mobilním zařízení a na sondě je uložen příslušný veřejný klíč* či je certifikát podepsán certifikační autoritou na sondě - *připojí se*
- Funkčnost tlačítka odpojení
- Funkčnost tlačítka pro restartování sondy
- Funkčnost tlačítka zpět
- Dokončení připojení se k sondě při odejití z aktivity v průběhu procesu připojování (přechod na hlavní obrazovku) - *na hlavní obrazovce se zobrazí stav připojování a po dokončení připojování zobrazí stav, zda je sonda připojena či nikoliv*
- Zjištění aktuálního času na sondě a jeho porovnáním s časem na mobilním zařízení:
  - Případ odlišného času - *detekuje odlišný čas a vypíše chybovou hlášku*
  - Případ stejného času a jiné časové zóny - *porovná časy v jiných časových zónách převodem na UTC*
  - Případ stejného času i časové zóny - *detekuje stejný čas na obou zařízeních*
- Zjištění přítomnosti souboru s logy na sondě:
  - Soubor neexistuje - *vypíše chybovou hlášku*
  - Soubor neexistuje - *detekuje přítomnost souboru*

- Zjištění přítomnosti souboru pro uložení iptables pravidel na sondě:
  - Soubor neexistuje - *vypíše chybovou hlášku*
  - Soubor neexistuje - *detekuje přítomnost souboru*

Nalezené problémy:

Nebyly nalezeny žádné problémy.

A.4 Hlavní obrazovka

Bylo otestováno:

- Zobrazení upozornění na rozdílný čas na sondě a chybějící soubor s logy či soubor pro ukládání iptables pravidel:
  - Detekce rozdílného času - *zobrazí upozornění, znemožní aktualizaci bezpečnostních incidentů*
  - Detekce chybějícího souboru s logy - *zobrazí upozornění, znemožní aktualizaci bezpečnostních incidentů*
  - Detekce chybějícího souboru pro uložení iptables pravidel - *zobrazí upozornění, na aktualizaci bezpečnostních incidentů nemá vliv*
  - Nedetekován žádný z problémů - *není zobrazeno žádné upozornění*
- Zobrazení stavu připojení sondy* a tlačítka připojení/odpojení**:
  - Nepřipojený stav, sonda ještě nebyla připojena - *zobrazí stav* a tlačítko pro připojení, které uživatele přesměruje na obrazovku nastavení komunikace***
  - Nepřipojený stav, sonda už v minulosti byla připojena - *zobrazí stav*, hostname použité při posledním připojení a tlačítko pro připojení***
  - Nepřipojený stav, zařízení není připojeno k síti Wi-Fi - *zobrazí stav* a tlačítko pro připojení, po jehož stisknutí je vypsána chybová hláška***
  - Připojení, autentizace úspěšná** - *připojí se, zobrazí změnu stavu a tlačítko pro odpojení*
  - Připojení, autentizace neúspěšná** - *vypíše chybovou hlášku*
  - Připojení na hlavní obrazovce a přechod na obrazovku nastavení komunikace** - *zobrazí stav připojování na obrazovce nastavení komunikace a po dokončení změní svůj stav podle toho, zda se připojení zdařilo*
  - Připojení na hlavní obrazovce a přechod do jiné části aplikace (na jinou aktivitu)** - *po dokončení připojování vypíše hlášku o připojení, případně chybovou hlášku, pokud se připojení nezdařilo*
  - Stisknutí tlačítka odpojení** - *sonda je odpojena, stav je aktualizován*

  - Zrušení spojení se sondou - *není detekováno (problém H1 - tabulka A.1)*

- Zobrazení a aktualizace času poslední aktualizace bezpečnostních incidentů:
  - Stav před první aktualizací bezpečnostních incidentů - *zobrazuje "no data"*
  - Zobrazení času poslední aktualizace bezpečnostních incidentů, pokud právě neprobíhá aktualizace (sonda není připojena) - *zobrazí poslední čas*
  - Aktualizace času při aktualizaci bezpečnostních incidentů - *po aktualizaci (uložení záznamů do databáze) zobrazí aktuální počet*
- Zobrazení a aktualizace počtu bezpečnostních incidentů:
  - Stav před první aktualizací bezpečnostních incidentů - *zobrazuje číslo 0*
  - Zobrazení počtu, pokud neprobíhá aktualizace bezpečnostních incidentů (sonda není připojena) - *zobrazí počet bezpečnostních incidentů v databázi*
  - Aktualizace počtu při aktualizaci bezpečnostních incidentů - *po aktualizaci zobrazí aktuální počet*
  - Aktualizace počtu při získání velkého počtu (5000) bezpečnostních incidentů v rámci jedné aktualizace bezpečnostních incidentů - *po aktualizaci zobrazí aktuální počet*
- Zobrazení a aktualizace počtu zařízení v síti:
  - Stav před prvním zjištěním počtu zařízení v síti - *zobrazuje "-"*
  - Zobrazení hodnoty zjištěné při posledním skenování přu spuštění aktivity - *zobrazí hodnotu*
  - Aktualizace počtu při nalezení zařízení v síti při skenování** - *zobrazí hodnotu, pokud je vyšší, než počet všech nalezených zařízení při posledním skenování (než aktuálně zobrazená hodnota)*
  - Zobrazení hodnoty po dokončení skenování - *zobrazí hodnotu*
- Aktualizace počtu bezpečnostních incidentů vs. aktualizace počtu zařízení v síti - *dlouhotrvající operace zjišťování zařízení v síti běžící ve vlákně na pozadí blokuje spuštění jiné operace (připojení se k sondě, aktualizace bezpečnostních incidentů), jelikož jsou spuštěny ve stejném vlákně (problém H2 - tabulka A.2)*, ve druhé verzi bez problému (aktualizace probíhají ve zvláštních vláknech a navzájem se neovlivňují)***
- Funkčnost tlačítka zpět

Nalezené problémy:

Tabulka A.1: Problém H1

Značka	H1
Problém	Při vypnutí sondy není detekováno přerušení spojení.
Závažnost	3
Verze aplikace	1, 2
Řešení	U první verze byl nastaven před připojením k sondě ServerAliveInterval na 1 sekundu. U druhé verze je při připojení se k sondě nastaven Timer, který každých 5 sekund kontroluje, zda je sonda připojena a při detekci výpadku vypíše chybovou hlášku v jakékoliv časti aplikace.

Tabulka A.2: Problém H2

Značka	H2
Problém	Zjišťování zařízení v síti blokuje jiné operace na pozadí.
Závažnost	2
Verze aplikace	1
Řešení	Spuštění operace zjišťování zařízení v síti v jiném vlákně na pozadí pomocí funkce executeOnExecutor().

A.5 Obrazovka seznamu bezpečnostních incidentů

Bylo otestováno:

- Funkčnost operace aktualizace bezpečnostních incidentů:
  - Běh na pozadí - *část kódu blokuje hlavní vlákno aplikace (problém S1 - tabulka A.3)*, u druhé verze bez problému***
  - Získání počtu nových bezpečnostních incidentů při první aktualizaci - *získá počet všech bezpečnostních incidentů*
  - Získání počtu nových bezpečnostních incidentů při dalších aktualizacích - *získá počet bezpečnostních incidentů od uloženého data a času poslední aktualizace*
  - Aktualizace, při které na sondě nejsou žádné nové bezpečnostní incidenty - *uloží nový čas poslední aktualizace signatur*
  - Aktualizace, při které jsou na sondě nalezeny nové bezpečnostní incidenty - *pokus o stažení souboru s novými bezpečnostními incidenty při některých aktualizacích probíhá v době, kdy sonda nedokončila zápis do tohoto souboru, a nedojde k aktualizaci (problém S2 - tabulka A.4)*, ve druhé verzi aplikace proběhne bez problému***
  - Přerušení aktualizace (ukončení aplikace) - *v některých případech je uložena nová časová značka poslední aktualizace signatur, ale*

záznam není vložen do databáze, tedy jej aplikace nezíská (problém S3 - tabulka A.5), ve druhé verzi aplikace proběhne bez problému***

- Zobrazení a aktualizace počtu bezpečnostních incidentů v zobrazeném seznamu:
  - Zobrazení počtu při zobrazení seznamu všech bezpečnostních incidentů - *zobrazí hodnotu*
  - Zobrazení počtu při zobrazení seznamu filtrovaných bezpečnostních incidentů - *zobrazí hodnotu*
  - Aktualizace počtu při aktualizaci seznamu (přidání nového bezpečnostního incidentu do databáze) - *zobrazí aktualizovanou hodnotu*
- Zobrazení a aktualizace seznamu bezpečnostních incidentů:
  - Zobrazení bezpečnostního incidentu uvnitř seznamu - *priorita závažnosti bezpečnostních incidentů je přehozená (problém S4 - tabulka A.6)*, ve druhé verzi aplikace je správně***
  - Zobrazení bezpečnostních incidentů pouze za posledních 30 dní při první aktualizaci - *zkopíruje do mobilního zařízení pouze záznamy, které nejsou starší než 30 dní*
  - Zobrazení bezpečnostních incidentů pouze za posledních 30 dní při dalších aktualizacích - *při spuštění aplikace jsou starší záznamy smazány, zobrazuje tedy jen záznamy za posledních 30 dní*
  - Zobrazení nefiltrovaného seznamu - *při přidání nového záznamu s novými IP adresami či signaturou, která v databázi není, dochází k filtrování (problém S5 - tabulka A.7)*, ve druhé verzi aplikace bez problému***
  - Zobrazení filtrovaného seznamu - *při návratu na seznam bezpečnostních incidentů z obrazovky filtrování je při aktualizaci bezpečnostních incidentů zobrazen nefiltrovaný seznam z důvodu aktivního sledování objektu LiveData pro získání nefiltrovaných hodnot (problém S6 - tabulka A.8)*, ve druhé verzi aplikace bez problému***
  - Případ dokončení aktualizace bezpečnostních incidentů v jiné části aplikace a návratu zpět na obrazovku seznamu bezpečnostních incidentů při zobrazení nefiltrovaného seznamu - *v případě nového záznamu s novými IP adresami či signaturou, která v databázi není, dochází k filtrování, jelikož nedošlo k aktualizaci hodnot pro filtrování (problém S7 - tabulka A.9)*, ve druhé verzi aplikace bez problému***
  - Případ dokončení aktualizace bezpečnostních incidentů v jiné části aplikace a návratu zpět na obrazovku seznamu bezpečnostních incidentů při zobrazení filtrovaného seznamu - *zobrazí filtrovaný seznam včetně případných nových záznamů*
  - Aktualizace seznamu při získání malého počtu nových bezpečnostních incidentů - *zobrazí aktualizovaný seznamu*

  - Aktualizace seznamu při získání velkého počtu (5000) nových bezpečnostních incidentů - *zobrazí aktualizovaný seznam, ale s prodlevou úměrnou počtu nových záznamů (problém S8 - tabulka A.10)*
  - Rolování na první pozici v seznamu při přidání nového bezpečnostního incidentu do seznamu** - *roluje na první pozici včetně animace*
  - Aktualizace seznamu při získání bezpečnostních incidentů z velmi velkého souboru (stovky tisíc záznamů)** - *načtení souboru a výběr nových záznamů z něj trvá příliš dlouho, v případě velmi velkého souboru několik minut, což z důvodu nastaveného timeoutu pro zpracování příkazu znemožní aktualizaci (problém S9 - tabulka A.11)*
  - Zobrazení seznamu při velkém počtu (50000) záznamů v databázi - *zobrazí seznam s prodlevou, do té doby je seznam prázdný (problém S10 - tabulka A.12)*, ve druhé verzi aplikace bez problému***

- Funkčnost tlačítka zpět

Nalezené problémy:

Tabulka A.3: Problém S1

Značka	S1
Problém	Aktualizace bezpečnostních incidentů neběží na pozadí.
Závažnost	1
Verze aplikace	1
Řešení	Aktualizace mimo hlavní vlákno aplikace pomocí třídy ScheduledThreadPoolExecutor s nastaveným zpožděním 5 sekund (další aktualizace proběhne vždy 5 sekund po dokončení minulé aktualizace).

Tabulka A.4: Problém S2

Značka	S2
Problém	Stažení souboru s novými bezpečnostními incidenty před ukončením zápisu sondou do něj.
Závažnost	2
Verze aplikace	1
Řešení	V případě neúspěšného stažení souboru je vykonán nový pokus o stažení souboru, maximálně je však vykonáno 10 pokusů o stažení souboru. Po ukončení stahování souboru je poslán příkaz na smazání souboru na sondě a před stažením je poslán příkaz pro zjištění existence souboru.

Tabulka A.5: Problém S3

Značka	S3
Problém	Přerušení aktualizace bezpečnostních incidentů v některých případech aplikaci znemožní získání záznamu.
Závažnost	1
Verze aplikace	1
Řešení	Čas poslední aktualizace bezpečnostních incidentů je aktualizován až po uložení záznamů, takže při spuštění nové aktualizace budou staženy i bezpečnostní incidenty, u kterých byla aktualizace přerušena.

Tabulka A.6: Problém S4

Značka	S4
Problém	Přehozená priorita závažností 1-3 bezpečnostních incidentů.
Závažnost	3
Verze aplikace	1
Řešení	Změněno seřazení v případě řazení dle závažnosti a vyměněny barvy pro závažnost 1 a 3 tak, aby zobrazení odpovídalo závažnosti 1 jako nejvyšší a závažnosti 3 jako nejnižší.

Tabulka A.7: Problém S5

Značka	S5
Problém	Zobrazení filtrovaného seznamu bezpečnostních incidentů, i když nemá být filtrován.
Závažnost	2
Verze aplikace	1
Řešení	Aktualizace hodnot pro filtrování při aktualizaci seznamu (přidání nového záznamu do databáze).

Tabulka A.8: Problém S6

Značka	S6
Problém	Zobrazení nefiltrovaného seznamu bezpečnostních incidentů, ačkoliv má být filtrován.
Závažnost	2
Verze aplikace	1
Řešení	Odebrání sledování LiveData objektu při přechodu aktivity na pozadí. Sledování objektu tedy nepřetrvává do opětovného přechodu aktivity do popředí.

Tabulka A.9: Problém S7

Značka	S7
Problém	Zobrazení filtrovaného seznamu bezpečnostních incidentů, i když nemá být filtrován v případě aktualizace v jiné části aplikace.
Závažnost	2
Verze aplikace	1
Řešení	K uloženým hodnotám pro filtry jsou v aplikaci uloženy i boolean hodnoty, které udávají, zda je seznam bezpečnostních incidentů filtrován jednotlivými filtry. Hodnota je aktualizována při každém uložení filtrů na obrazovce filtrování bezpečnostních incidentů a nabývá hodnoty true, pokud je nastaveno filtrování dle určitého filtru (dle závažnosti, dle cílové IP adresy apod.). Pokud je některý z filtrů aplikován, hodnoty pro ostatní (neaplikované) filtry jsou před načtením databáze aktualizovány. Pokud není aplikován žádný filtr, je zobrazen nefiltrovaný seznam bezpečnostních incidentů.

Tabulka A.10: Problém S8

Značka	S8
Problém	Časově náročné zpracování aktualizace velkého počtu bezpečnostních incidentů.
Závažnost	3
Verze aplikace	1
Řešení	Příkaz jq je upraven tak, aby byla zpracována jen určitá část nových záznamů v případě jejich velkého počtu. Je nastavena hodnota 100 záznamů. Aktualizace probíhá ve for cyklu, tedy následujících 100 záznamů je zpracováno ihned po zpracování předchozích 100 záznamů. Pozn.: Ve druhé verzi aplikace je zvolen počet 500 záznamů pro jednu část aktualizace a pouze pokud je záznamů méně než 2000. Případ nastává pouze pokud bezpečnostní incidenty delší dobu nebyly aktualizovány či při první aktualizaci.

Tabulka A.11: Problém S9

Značka	S9
Problém	Časově náročná aktualizace bezpečnostních incidentů v případě velmi velkého souboru na sondě znemožní aktualizaci.
Závažnost	1
Verze aplikace	2
Řešení	Jelikož v souboru eve.json na sondě je uchováváno velké množství záznamů událostí na sondě, může za delší dobu používání jeho velikost výrazně narůst. Problém bude řešen přímo na sondě vytvořením nového souboru, kde se budou uchovávat pouze události ze Suricaty typu alert a tento soubor bude na sondě pravidelně promazáván, aby byla udržena malá velikost souboru. V aplikaci bude změněna cesta k souboru pro aktualizaci bezpečnostních incidentů.

Tabulka A.12: Problém S10

Značka	S10
Problém	Prodleva při zobrazení velkého množství záznamů v seznamu bezpečnostních incidentů.
Závažnost	3
Verze aplikace	1
Řešení	Zobrazení kruhového progressbaru dokud není načten seznam bezpečnostních incidentů.

A.6 Obrazovka filtrování bezpečnostních incidentů

Bylo otestováno:

- Zobrazení počtu aktivních filtrů
- Funkčnost tlačítka pro resetování filtrů
- Zobrazení počtu vybraných hodnot pro filtry a počtu dostupných hodnot - *pokud je aktualizace bezpečnostních incidentů dokončena v obrazovce filtrování, nejsou zobrazeny aktuální hodnoty dostupné pro filtry, pokud v databázi přibyl záznam s novými IP adresami či signaturou, která v databázi není (problém F1 - tabulka A.13)*, ve druhé verzi aplikace bez problému***
- Funkčnost výběru hodnot pro filtry pomocí checkboxů
- Aktualizace seznamu výběru hodnot parametru pro filtr, pokud je do databáze při zobrazení seznamu výběru přidán nový záznam s novou hodnotou daného parametru** - *seznam výběru hodnot pro filtr se aktualizuje včetně zaškrtnutí nové hodnoty*
- Funkčnost zadání data a času pro nastavení času od a do:
  - Zadání jen času od - *filtrování od určitého data, nové bezpečnostní incidenty se zobrazují*
  - Zadání jen času do - *filtrování jen do určitého data*
  - Čas od je dříve než čas do - *filtrování v daném rozmezí*
  - Čas do dříve než od - *při potvrzení aplikování filtrů je uživateli zobrazen chybový dialog, je znemožněna aplikace takového filtru*
- Funkčnost řazení dle času a závažnosti:
  - Pro nefiltrovaná data - *záznamy jsou řazeny dle požadavku uživatele*
  - Pro filtrovaná data - *záznamy jsou řazeny dle požadavku uživatele*
- Funkčnost tlačítka zpět
- Funkčnost tlačítka pro potvrzení aplikování filtrů

Nalezené problémy:

Tabulka A.13: Problém F1

Značka	F1
Problém	V případě aktualizace bezpečnostních incidentů na obrazovce filtrování se v některých případech nezobrazují aktuální dostupné hodnoty pro filtry.
Závažnost	3
Verze aplikace	1
Řešení	Pomocí LiveData objektu jsou sledovány změny seznamu zdrojových IP adres, cílových IP adres a signatur na obrazovce filtrování bezpečnostních incidentů. Pokud je detekována změna, je aktualizován daný seznam, zobrazený příslušný počet vybraných/všech prvků v seznamu a v případě, že je zobrazen příslušný dialog pro výběr prvků, je aktualizován i příslušný dialog, přičemž je zachován stav zaškrtnutí jednotlivých prvků a prvek, který je do seznamu přidán, je označen jako vybraný (je zaškrtnutý).

A.7 Obrazovka detailu bezpečnostního incidentu

Bylo otestováno:

- Zobrazení informací příslušného bezpečnostního incidentu
- Funkčnost možnosti rozbalení/zabalení jednotlivých částí
- Funkčnost tlačítka zpět

Nalezené problémy:

Nebyly nalezeny žádné problémy.

A.8 Obrazovka seznamu zařízení v síti

Bylo otestováno:

- Zobrazení zařízení v síti nalezených při skenování započatém na hlavní obrazovce:
  - Případ dokončeného zjišťování na hlavní obrazovce - *je zobrazen seznam dle předchozího zjištění a spuštěna nová aktualizace pro zjištění aktuálního seznamu zařízení v síti*
  - Případ stále běžícího zjišťování na hlavní obrazovce - *způsobí pád aplikace (problém D1 - tabulka A.14)*, ve druhé verzi aplikace pokračuje ve skenování***
- Zobrazení počtu nalezených zařízení
- Zobrazení seznamu nalezených zařízení:
  - Před dokončením prvního skenování - *je zobrazen prázdný seznam*

  - Před dokončením následujících skenování sítě - *zobrazí seznam zařízení nalezených při předchozím skenování a po dokončení skenování aktualizuje seznam*, nebo aktualizuje seznam při nalezení prvního zařízení při novém skenování v případě druhé verze apliakce***
  - Po dokončení skenování sítě - *zobrazí aktualizovaný seznam*
  - Přidání nově nalezeného zařízení do seznamu v průběhu skenování** - *nalezené zařízení je zobrazeno v seznamu, při aktualizaci rozbalené položky zůstávají rozbalené*

- Funkčnost manuálního spuštění nového skenování zařízení v síti**:
  - Tlačítkem v app baru - *skenování je spuštěno*
  - Gestem potažení shora dolů - *skenování je spuštěno*
  - Pokud mobilní zařízení není připojeno k Wi-Fi síti - *na obrazovce je zobrazen text informující uživatele, že zařízení není připojeno k síti, zařízení nalezená při posledním skenování na obrazovce zůstanou zobrazena*
- Zobrazení informace o nepřipojení k síti, pokud mobilní zařízení není připojeno k Wi-Fi síti**
- Funkčnost možnosti rozbalení/zabalení jednotlivých nalezených zařízení
- Zobrazení zjištěných údajů o zařízení po rozbalení
- Funkčnost tlačítka zpět

Nalezené problémy:

Tabulka A.14: Problém D1

Značka	D1
Problém	Spuštění operace zjišťování zařízení v síti vícekrát zároveň způsobí na obrazovce seznamu zařízení v síti pád aplikace.
Závažnost	1
Verze aplikace	1
Řešení	Použití zámku ReentrantLock pro uzamknutí přístupu k operaci zjišťování zařízení v síti. Pokud je operace již spuštěna z hlavní obrazovky, pro přístup z obrazovky zařízení v síti je uzamknuta. Po ukončení operace zjišťování zařízení proces na obrazovce zařízení v síti jen získá seznam zařízení získaný operací, která byla spuštěna na hlavní obrazovce. Pozn.: Ve druhé verzi aplikace je spouštění skenování řízeno v metodě startScan() třídy DeviceScanner.

A.9 Hlavní obrazovka blokace webových stránek

Bylo otestováno:

- Zobrazení karet profilů a stránek
- Zobrazení aktuálního počtu blokovaných stránek a aktivních profilů a celkového počtu:
  - Stav při otevření obrazovky - *zobrazí aktuální hodnoty*
  - Stav, kdy došlo k povolení/zablokování stránky mimo aplikaci před otevřením obrazovky - *zobrazí aktuální hodnoty*
  - Stav, kdy došlo k aktivování/deaktivování profilu s nastaveným časovým oknem v době, kdy je uživatel na obrazovce - *hodnota blokovaných profilů a stránek se neaktualizuje (problém B1 - tabulka A.15)*, ve druhé verzi aplikace aktualizuje hodnoty***
- Funkčnost tlačítka rozbalení/zabalení karty
- Zobrazení seznamu aktivních profilů a stránek při rozbalení karty
- Funkčnost tlačítka pro zobrazení obrazovky seznamu kategorií
- Funkčnost tlačítek pro přechod na obrazovky seznamu profilů a stránek a na obrazovky přidání nových profilů a stránek
- Funkčnost tlačítka zpět - *tlačítkem zpět je uživatel navrácen na poslední obrazovku v části blokace internetových stránek, kterou navštívil, a dochází tak k zacyklení v části profilů, místo navrácení se na hlavní obrazovku (problém B2 - tabulka A.16)*, ve druhé verzi aplikace návrat na hlavní obrazovku aplikace***

Nalezené problémy:

Tabulka A.15: Problém B1

Značka	B1
Problém	V případě změny stavu profilu s časovým oknem na aktivní/neaktivní v průběhu zobrazování hlavní obrazovky blokace internetových stránek není aktualizován počet blokovaných profilů a stránek.
Závažnost	3
Verze aplikace	1
Řešení	Při spuštění aktivity je vytvořen seznam všech změn stavů profilů s časovým oknem v následujících 24 hodinách, seřazený dle času. Pro první z času v seznamu je při volání metody vytvořen jednorázový Timer, který nastaví akci aktualizace na daný čas a zároveň nastaví nový Timer na čas, který v seznamu následuje. Tímto je při změně stavu profilu s časovým oknem aktualizován i seznam aktivních profilů a blokovaných webových stránek na obrazovce.

Tabulka A.16: Problém B2

Značka	B2
Problém	Zacyklení tlačítka zpět v části blokování webových stránek.
Závažnost	3
Verze aplikace	1
Řešení	Na obrazovkách seznamu kategorií, seznamu profilů a seznamu internetových stránek není spuštěna aktivita hlavní obrazovky blokace internetových stránek, ale je volána metoda finish() dané aktivity.

A.10 Obrazovka seznamu kategorií

Bylo otestováno:

- Zobrazení kategorií dle abecedy
- Zobrazení aktuálních počtů blokovaných a všech stránek u každé kategorie:
  - Případ blokace stránky uživatelem - *počet je zobrazen*
  - Případ blokace kategorie profilem - *počet je zobrazen (všechny stránky kategorie jsou blokované)*
- Zobrazení obrazovky detailu kategorie po stisknutí kategorie
- Funkčnost tlačítka přidání kategorie
- Funkčnost tlačítka zpět

Nalezené problémy:

Nebyly nalezeny žádné problémy.

A.11 Obrazovka detailu kategorie

Bylo otestováno:

- Zobrazení názvu kategorie
- Zobrazení aktuálního počtu blokovaných a všech stránek v kategorii:
  - Kategorie neobsahuje žádné webové stránky - *zobrazí 0/0*
  - Kategorie obsahuje webové stránky, které byly blokovány mimo obrazovku detailu kategorie - *zobrazí hodnoty*
  - Stránka byla blokována mimo obrazovku detailu kategorie - *zobrazí hodnoty*
  - Kategorie obsahuje webové stránky, které jsou zablokovány na obrazovce detailu kategorie - *po zablokování (popř. povolení) se neaktualizuje počet blokovaných stránek (problém C1 - tabulka A.17)*, ve druhé verzi aplikaci se počet aktualizuje***

  - Kategorie je součástí aktivního profilu - *všechny stránky jsou blokovány*
  - Kategorie je součástí neaktivního profilu - *zobrazí hodnoty stejně jako by nebyla součástí profilu*

- Zobrazení seznamu stránek v kategorii dle abecedy
- Funkčnost zablokování/povolení stránky (přidání/odebrání stránky do/z iptables zkontrolováno pomocí příkazu „sudo iptables -S“ na sondě):
  - Kategorie není blokovaná profilem - *povolí/zablokuje stránky*
  - Kategorie je blokovaná profilem - *povolení stránky z blokovaného stavu profilem není aplikací povoleno, na obrazovce je vypsána chybová hláška*
  - Sonda není připojena - *blokace/povolení stránky není umožněno, na obrazovce je vypsána chybová hláška*
- Zobrazení stavu stránky povolena/blokována v případě změny pravidel mimo aplikaci před zobrazením seznamu:
  - Povolení blokované stránky mimo aplikaci - *stránka je v seznamu zobrazena jako povolená*
  - Zablokování povolené stránky mimo aplikaci - *stránka je v seznamu zobrazena jako blokovaná*
- Funkčnost tlačítka úpravy kategorie
- Funkčnost tlačítka zpět - *zobrazí obrazovku seznamu kategorií, ale pokud byl počet stránek v kategorii změněn, nezobrazuje aktuální stav (problém C2 - tabulka A.18)*, ve druhé verzi aplikace zobrazí aktualizovaný počet***

Nalezené problémy:

Tabulka A.17: Problém C1

Značka	C1
Problém	Není aktualizován počet blokovaných stránek na obrazovce detailu kategorie v případě změny blokací stránek přímo na obrazovce.
Závažnost	3
Verze aplikace	1
Řešení	Přidán Observer pro sledování počtu stránek v kategorii.

Tabulka A.18: Problém C2

Značka	C2
Problém	Není aktualizován počet blokovaných stránek na obrazovce seznamu kategorií při návratu z kategorie, u níž byl počet změněn.
Závažnost	3
Verze aplikace	1
Řešení	Observer pro sledování kategorií je volán v metodě onResume() místo onCreate() a dojde tedy k aktualizaci seznamu kategorií i při návratu z detailu kategorie.

A.12 Obrazovka přidání kategorie

Bylo otestováno:

- Zadání barvy:
  - První výběr barvy - *barva je nastavena*
  - Změna výběru barvy - *je nastavena nová barva*
  - Zrušení výběru barvy - *zůstane vybrána předchozí barva, popř. žádná barva pokud ještě barva nebyla vybrána*
- Funkčnost tlačítka uložení kategorie:
  - Bez zadaného názvu - *zobrazí chybový dialog*
  - Bez vybrané barvy - *zobrazí chybový dialog*
  - Oba parametry jsou zadány - *kategorie je uložena, přesun na obrazovku seznamu kategorií*
- Editování existující kategorie:
  - Automatické vyplnění polí při editování existující kategorie
  - Změna názvu
  - Změna barvy
  - Funkčnost uložení změny - *pokud není zadán název, zobrazí chybový dialog, jinak uloží změnu*
- Smazání kategorie:
  - Všechny stránky kategorie jsou povolené - *smaže z kategorie stránky (nejsou pak přiřazené do žádné kategorie) a smaže kategorii*
  - Některé stránky kategorie jsou blokované - *smaže z kategorie stránky a smaže kategorii*
  - Kategorie není součástí žádného profilu - *smaže z kategorie stránky a smaže kategorii*

  - Kategorie je součástí profilu, který není aktivní - *pokus o smazání kategorie vyvolá pád aplikace, v případě profilu s časovým oknem není odstraněna blokace stránek profilem (problém C3 - tabulka A.19)*, v případě druhé verze aplikace nepovolí smazání, vypíše chybovou hlášku***
  - Kategorie je součástí profilu, který je aktivní - *nepovolí smazání, vypíše chybovou hlášku*

- Funkčnost tlačítka zpět

Nalezené problémy:

Tabulka A.19: Problém C3

Značka	C3
Problém	Smazání kategorie, která je ovlivněna profilem, který aktuálně není blokován, způsobí pád aplikace. Stránky uvnitř kategorie jsou z kategorie smazány, ale v případě profilu s časovým oknem u nich blokace stránek profilem není odstraněna.
Závažnost	1
Verze aplikace	1
Řešení	Pokud je kategorie spojena s nějakým profilem, nepovolí její smazání a vypíše chybovou hlášku s výpisem všech profilů, se kterými je spojena.

A.13 Obrazovka seznamu webových stránek

Bylo otestováno:

- Zobrazení počtu blokovaných a všech stránek
- Zobrazení seznamu stránek dle kategorie a názvu, blokované nahoře
- Zobrazení seznamu stránek při aktivních profilech:
  - Blokování kategorie profilem bez časového okna - *ovlivněné stránky jsou zobrazeny jako blokované*
  - Blokování kategorie profilem s časovým oknem - *ovlivněné stránky jsou zobrazeny jako blokované*
  - Blokování kategorie několika profily zároveň, stránka je blokována - *ovlivněné stránky jsou zobrazeny jako blokované*
  - Blokování kategorie několika profily zároveň, stránka je povolena - *v seznamu se stránka zobrazí vícekrát (problém W1 - tabulka A.20)*, ve druhé verzi aplikace se zobrazí jednou jako blokovaná***
- Zobrazení stavu stránky povolena/blokována v případě změny pravidel mimo aplikaci před zobrazením seznamu:
  - Povolení blokované stránky mimo aplikaci - *stránka je v seznamu zobrazena jako povolená*

  - Povolení blokované stránky, která je zároveň blokovaná profilem, mimo aplikaci - *stránka je v seznamu zobrazena jako blokovaná, v databázi je povolena*
  - Povolení povolené stránky, která je zároveň blokovaná profilem bez časového okna, mimo aplikaci - *stránka je v seznamu zobrazena jako povolená a je smazána ze své kategorie, která je stále profilem blokovaná*
  - Povolení povolené stránky, která je zároveň blokovaná profilem s časovým oknem, mimo aplikaci - *stránka je v seznamu zobrazena jako blokovaná*
  - Zablokování povolené stránky mimo aplikaci- *stránka je v seznamu zobrazena jako blokovaná, v databázi je zablokována*
  - Zablokování povolené stránky, která je zároveň blokovaná profilem, mimo aplikaci - *stránka je v seznamu zobrazena jako blokovaná, v databázi je zablokována*
  - Zablokování stránky, která není v databázi, mimo aplikaci - *stránka je přidána do databáze bez specifikované kategorie a je v seznamu zobrazena jako blokovaná*

- Funkčnost zablokování/povolení stránky:
  - Případ nepřipojené sondy - *blokace/povolání stránky není umožněna, uživateli je na obrazovce zobrazeno upozornění v podobě krátké zprávy*
  - Stránka je povolena a není blokována profilem - *stránka je zablokována*
  - Stránka je blokována a není blokována profilem - *stránka je povolena*
  - Stránka je povolena a je blokována profilem - *blokace stránky není umožněna, uživateli je na obrazovce zobrazeno upozornění v podobě krátké zprávy*
  - Stránka je blokována a je blokována profilem - *povolení stránky není umožněno, uživateli je na obrazovce zobrazeno upozornění v podobě krátké zprávy*
- Funkčnost tlačítka pro přístup k obrazovce přidání nové stránky
- Zobrazení obrazovky úpravy stránky po stisknutí stránky v seznamu
- Funkčnost tlačítka zpět

Nalezené problémy:

Tabulka A.20: Problém W1

Značka	W1
Problém	V případě blokace povolené stránky několika profily najednou je stránka v seznamu zobrazena vícekrát.
Závažnost	3
Verze aplikace	1
Řešení	Při zjišťování profilů, které blokují kategorie stránek, je při přidání kategorie do seznamu kategorií blokovaných profily zkontrolováno, zda již kategorie v tomto seznamu není.

A.14 Obrazovka přidání webové stránky

Bylo otestováno:

- Výběr kategorie
- Výběr počátečního stavu povolena/blokována
- Funkčnost tlačítka uložení nové stránky:
  - Nezadaná adresa stránky - *není umožněno uložení stránky, je zobrazena chybová hláška*
  - Adresa stránky totožná se stránkou již existující v databázi - *není umožněno uložení stránky, je zobrazen chybový dialog*
  - Zadaná adresa, nevybraná kategorie - *stránka je uložena a nepatří do žádné kategorie*
  - Stránka povolena, sonda připojena - *stránka je uložena*
  - Stránka blokována, sonda připojena - *stránka je uložena a je poslán příkaz pro blokaci*
  - Kategorie stránky blokována profilem, stránka povolena, sonda připojena - *stránka je uložena a je poslán příkaz pro blokaci profilem*
  - Kategorie stránky je součástí neaktivního profilu bez časového okna, sonda připojena - *stránka je uložena*
  - Kategorie stránky je součástí neaktivního profilu s časovým oknem, sonda připojena - *stránka je uložena a je poslán příkaz pro blokaci profilem*
  - Kategorie stránky blokována profilem, stránka blokována, sonda připojena - *stránka je uložena jako blokovaná a jsou poslány příkazy pro blokaci*
  - Stránka povolena, sonda odpojena - *stránka je uložena*
  - Stránka blokována, sonda odpojena - *není umožněno uložení stránky, je zobrazen chybový dialog*

  - Kategorie stránky blokována profilem, sonda odpojena - *není umožněno uložení stránky, je zobrazen chybový dialog*
  - Kategorie stránky je součástí neaktivního profilu bez časového okna, stránka je povolena, sonda odpojena - *stránka je uložena*
  - Kategorie stránky je součástí neaktivního profilu s časovým oknem, stránka je povolena, sonda odpojena - *není umožněno uložení stránky, je zobrazen chybový dialog*

- Předvyplnění polí při editování existující stránky
- Zobrazení stavu povolena/blokována při editování existující stránky
- Funkčnost tlačítka uložení editované stránky:
  - Změna adresy - *není umožněno uložení stránky, je zobrazena chybová hláška*
  - Změna kategorie - *stránka je odebrána z původní kategorie a uložena do nové kategorie*
  - Změna adresy, stránka blokována, sonda odpojena - *způsobí pád aplikace (problém W2 - tabulka A.21)*, ve druhé verzi aplikace nepovolí uložení změny a zobrazí chybovou hlášku***
  - Změna kategorie, stránka blokována, sonda odpojena - *nepovolí uložení změny a zobrazí chybovou hlášku*
  - Změna adresy, stránka povolena, sonda odpojena - *stránka je uložena s novou adresou*
  - Změna kategorie, která není součástí žádného profilu či jen neaktivního profilu bez časového okna, stránka povolena, sonda odpojena - *nepovolí uložení změny a zobrazí chybovou hlášku*, ve druhé verzi aplikace je stránka uložena do nové kategorie a odebrána z původní kategorie***
  - Změna adresy či kategorie, stránka povolena, kategorie blokována profilem či součástí profilu s časovým oknem, sonda odpojena - *stránka je uložena s novou adresou/kategorií, ale profilem (který blokuje původní kategorii stránky) zůstává blokována stará stránka (problém W2 - tabulka A.21)*, ve druhé verzi aplikace nepovolí uložení změny a zobrazí chybovou hlášku***
  - Změna adresy, stránka blokována, kategorie blokována profilem či součástí profilu s časovým oknem, sonda odpojena - *uložení úpravy stránka způsobí pád aplikace (problém W2 - tabulka A.21)*, ve druhé verzi aplikace nepovolí uložení změny a zobrazí chybovou hlášku***
  - Změna kategorie, stránka blokována, stránka blokována profilem či součástí profilu s časovým oknem, sonda odpojena - *stránka je uložena do nové kategorie, ale zůstává blokována profilem, který blokuje původní kategorii (problém W2 - tabulka A.21)*, ve druhé verzi aplikace nepovolí uložení změny a zobrazí chybovou hlášku***

  - Změna kategorie, nová kategorie je blokována profilem či součástí profilu s časovým oknem, sonda je odpojena - *stránka není profilem zablokována (problém W3 - tabulka A.22)*, ve druhé verzi aplikace je stránka uložena s novou kategorií a poslán příkaz pro blokaci profilem***
  - Změna kategorie, nová kategorie je blokována profilem či součástí profilu s časovým oknem, sonda je odpojena - *je umožněno přesunout stránku do kategorie právě blokované profilem, stránka není profilem zablokována (problém W4 - tabulka A.23)*, ve druhé verzi aplikace není povoleno uložení změny a je zobrazen chybový dialog***
  - Povolení blokované stránky, sonda odpojena - *není umožněno uložení upravené stránky, je zobrazena chybová hláška*
  - Povolení blokované stránky při současné změně adresy, sonda připojena - *původní adresa je povolena a je uložena povolená stránka s novou adresou*, ve druhé verzi aplikace není původní stránka povolena na sondě (problém W5 - tabulka A.24)***
  - Povolení blokované stránky, kategorie není součástí profilu či neaktivního profilu bez časového okna, sonda připojena - *stránka je povolena*
  - Povolení blokované stránky, kategorie je součástí aktivního profilu či profilu s časovým oknem, sonda připojena - *stránka je v databázi povolena, ale zároveň zůstává zablokována profilem*
  - Blokování povolené stránky, sonda odpojena - *není umožněno uložení upravené stránky, je zobrazena chybová hláška*
  - Blokování povolené stránky při současné změně adresy, sonda připojena - *původní stránka je povolena, nová adresa je uložena jako povolená*
  - Blokování povolené stránky, kategorie není součástí profilu či neaktivního profilu bez časového okna, sonda připojena - *stránka je zablokována*
  - Blokování povolené stránky, kategorie je součástí aktivního profilu či profilu s časovým oknem, sonda připojena - *stránka je zablokována a zároveň zůstává blokována profilem*

- Funkčnost smazání existující stránky:
  - Stránka povolena, sonda připojena - *stránka je smazána*
  - Stránka blokována, sonda připojena - *je poslán příkaz na povolení stránky, stránka je smazána*
  - Stránka povolena, sonda odpojena - *stránka je smazána*
  - Stránka blokována, sonda odpojena - *způsobí pád aplikace (problém W6 - tabulka A.25)*, ve druhé verzi aplikace nepovolí smazání a zobrazí chybový dialog***

  - Stránka povolena, kategorie blokována profilem či součástí profilu s časovým oknem, sonda připojena - *blokace stránky je zrušena, blokace stránky profilem je zrušena, stránka je smazána*
  - Stránka blokována, kategorie blokována profilem či součástí profilu s časovým oknem, sonda připojena - *blokace stránky je zrušena, blokace stránky profilem je zrušena, stránka je smazána*
  - Kategorie stránky blokována profilem či součástí profilu s časovým oknem, sonda odpojena - *způsobí pád aplikace (problém W6 - tabulka A.25)*, ve druhé verzi aplikace nepovolí smazání a zobrazí chybový dialog***

- Funkčnost tlačítka zpět

Nalezené problémy:

Tabulka A.21: Problém W2

Značka	W2
Problém	Uložení upravené stránky se změněnou adresou způsobí v případě blokované stránky a nepřipojené sondy pád aplikace.
Závažnost	1
Verze aplikace	1
Řešení	Pokud je stránka blokována a sonda je odpojena, není umožněno uložení stránky se změněnou adresou a je zobrazena chybová hláška.

Tabulka A.22: Problém W3

Značka	W3
Problém	Při uložení upravené stránky se změněnou kategorií, kdy nová kategorie je blokována profilem, není při připojené sondě upravená stránka zablokována profilem, který blokuje novou kategorii.
Závažnost	1
Verze aplikace	1
Řešení	Pokud je nová kategorie blokována profilem, je dle něj zablokována i upravená stránka.

Tabulka A.23: Problém W4

Značka	W4
Problém	Je umožněno uložení upravené stránky se změněnou kategorií, kdy nová kategorie je blokována profilem, i když je sonda odpojena.
Závažnost	1
Verze aplikace	1
Řešení	Pokud je nová kategorie blokována profilem a sonda je odpojena, není umožněno uložení upravené stránky a je zobrazena chybová hláška.

Tabulka A.24: Problém W5

Značka	W5
Problém	Při změně adresy blokované webové stránky není původní adresa na sondě povolena.
Závažnost	2
Verze aplikace	2
Řešení	Byl posílán příkaz na blokaci nové adresy. Kód byl upraven pro odeslání příkazu pro zrušení blokace původní adresy.

Tabulka A.25: Problém W6

Značka	W6
Problém	Smazání blokované stránky při odpojené sondě způsobí pád aplikace.
Závažnost	1
Verze aplikace	1
Řešení	Pokud je stránka blokována a sonda je odpojena, není umožněno smazání stránky a je zobrazena chybová hláška.

A.15 Obrazovka seznamu profilů

Bylo otestováno:

- Zobrazení karty profilu:
  - Profil s časovým oknem - *zobrazen název, výčet dní, časové okno, blokované kategorie a tlačítko úpravy*
  - Profil bez časového okna - *zobrazen název, blokované kategorie, tlačítko blokace/povolení a tlačítko úpravy*
- Rozdělení profilů na aktivní a neaktivní, zobrazení aktivních nahoře:
  - Stav při otevření obrazovky - *zobrazí aktivní profily nahoře*
  - Stav při aktivaci/deaktivaci profilu bez časového okna - *při aktivaci profil přesune nahoru do částí aktivovaných profilů, při deaktivaci do části deaktivovaných*
  - Stav při aktivaci/deaktivaci profilu s časovým oknem - *profil není přesunut do patřičné části (problém P1 - tabulka A.26)*, ve druhé verzi aplikace je profil přesunut***
- Funkčnost aktivace/deaktivace profilu bez časového okna:
  - Sonda připojena - *stránky v kategoriích ovlivněných profilem jsou zablokovány/povoleny*
  - Sonda odpojena - *aktivace/deaktivace profilu není povolena, uživateli je na obrazovce zobrazeno upozornění v podobě krátké zprávy*

- Blokace stránek profily s časovým oknem při změně času** - *při změně času se neaktualizuje blokace na sondě, ani časové okno nastavené pro profily (problém P2 - tabulka A.27)*
- Funkčnost tlačítka pro přístup k obrazovce přidání nového profilu
- Funkčnost tlačítka pro přístup k obrazovce úpravy profilu
- Funkčnost tlačítka zpět

Nalezené problémy:

Tabulka A.26: Problém P1

Značka	P1
Problém	Pokud je uživatel na obrazovce seznamu profilů a je v daném čase aktivován/deaktivován profil s časovým oknem, není aktualizováno rozdělení seznamu na aktivní a neaktivní profily.
Závažnost	3
Verze aplikace	1
Řešení	Při volání metody onResume aktivity je vytvořen seznam všech změn stavů profilů s časovým oknem v následujících 24 hodinách, seřazený dle času. Pro první z času v seznamu je při volání metody vytvořen jednorázový Timer, který nastaví akci aktualizace na daný čas a zároveň nastaví nový Timer na čas, který v seznamu následuje. Při volání metody onPause aktivity je Timer zrušen. Tímto je při změně stavu profilu s časovým oknem aktualizován i seznam profilů na obrazovce.

Tabulka A.27: Problém P2

Značka	P2
Problém	Při změně času na zařízení neodpovídají časy blokací profily s časovým oknem zobrazovaným časům.
Závažnost	2
Verze aplikace	2
Řešení	Při každém připojení se k sondě si aplikace uvnitř třídy ProbeChecker zkontroluje aktuální posun času na mobilním zařízení oproti času UTC. Hodnotu posunu si aplikace ukládá do SharedPreferences. Pokud je detekována změna hodnoty posunu oproti času UTC, odešle na sondu příkazy pro povolení blokací webových stránek profily s časovým oknem a následně pro jejich opětovné zablokování, ale tentokrát s novým posunem. Takto zobrazený čas odpovídá časům nastaveným pro blokace na sondě.

A.16 Obrazovka přidání profilu

Bylo otestováno:

- Výběr kategorií pomocí checkboxů
- Výběr dní pomocí checkboxů
- Zobrazení výběru časového okna v případě vybraných dní a jeho schování v případě vyškrtnutí všech dní z výběru
- Výběr časového okna aktivace profilu od a do
- Funkčnost tlačítka uložení nového profilu:
  - Nezadán název - *není umožněno vytvoření nového profilu, je zobrazen chybový dialog*
  - Kategorie nejsou vybrány - *není umožněno vytvoření nového profilu, je zobrazen chybový dialog*
  - Zadaný název i kategorie, nevybrané dny, sonda připojena - *je vytvořen nový profil bez časového okna*
  - Zadaný název i kategorie, nevybrané dny, sonda odpojena - *není umožněno vytvoření nového profilu, je zobrazena chybová hláška*, ve druhé verzi aplikace je vytvořen nový profil bez časového okna*
  - Zadaný název i kategorie, vybrané dny, nezadaný čas - *není umožněno vytvoření nového profilu, je zobrazen chybový dialog*
  - Zadaný název i kategorie, vybrané dny, zadaný pouze čas od - *není umožněno vytvoření nového profilu, je zobrazen chybový dialog*
  - Zadaný název i kategorie, vybrané dny, zadaný čas do je před zadaným časem od - *není umožněno vytvoření nového profilu, je zobrazen chybový dialog*
  - Zadaný název i kategorie, vybrané dny i čas, sonda připojena - *je vytvořen nový profil s časovým oknem*
  - Zadaný název i kategorie, vybrané dny i čas, sonda odpojena - *není umožněno vytvoření nového profilu, je zobrazen chybový dialog*
  - Blokace stránek v kategoriích nově přidaného profilu s časovým oknem** - *stránky v kategoriích jsou po přidání profilu blokovány profilem*
- Předvyplnění polí při editování existujícího profilu
- Funkčnost tlačítka uložení změny existujícího profilu:
  - Změna názvu, profil aktivní či s časovým oknem, sonda připojena - *profil uložen s novým názvem*
  - Změna názvu, neaktivní profil bez časového okna, sonda připojena - *profil uložen s novým názvem*
  - Změna názvu, profil aktivní či s časovým oknem, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*

- Změna názvu, neaktivní profil bez časového okna, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*, ve druhé verzi aplikace je profil uložen s novým názvem***
- Změna kategorií, aktivní profil bez časového okna, sonda připojena - *stránky uvnitř původních blokovaných kategorií jsou povoleny, upravený profil je uložen a jsou zablokovány stránky v nových kategoriích*
- Změna kategorií, neaktivní profil bez časového okna, sonda připojena - *upravený profil je uložen*
- Změna kategorií, aktivní profil bez časového okna, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*
- Změna kategorií, neaktivní profil bez časového okna, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*, ve druhé verzi aplikace uloží upravený profil***
- Změna kategorií, profil s časovým oknem, sonda připojena - *stránky uvnitř původních blokovaných kategorií jsou povoleny, upravený profil je uložen a jsou zablokovány stránky v nových kategoriích*
- Změna kategorií, profil s časovým oknem, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*
- Aktivní profil bez časového okna, přidání časového okna, sonda připojena - *stránky uvnitř původních blokovaných kategorií jsou povoleny, upravený profil je uložen a jsou s časovým oknem zablokovány stránky v nových kategoriích*
- Neaktivní profil bez časového okna, přidání časového okna, sonda připojena - *upravený profil je uložen a jsou s časovým omezením zablokovány stránky v nových kategoriích*
- Profil s časovým oknem, odebrání časového okna, sonda připojena - *stránky uvnitř původních blokovaných kategorií jsou povoleny, upravený profil je uložen*
- Profil s časovým oknem, změna vybraných dní či časového okna, sonda připojena - *stránky uvnitř původních blokovaných kategorií jsou povoleny, upravený profil je uložen a jsou s časovým oknem zablokovány stránky v nových kategoriích*
- Aktivní profil bez časového okna, přidání časového okna, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*
- Neaktivní profil bez časového okna, přidání časového okna, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*
- Profil s časovým oknem, odebrání časového okna, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*
- Profil s časovým oknem, změna vybraných dní či časového okna, sonda odpojena - *uložení upraveného profilu není povoleno, je zobrazen chybový dialog*

- Funkčnost smazání existujícího profilu:
  - Profil bez časového okna, profil aktivní, sonda připojena - *profil je smazán, stránky blokované profilem jsou povoleny*
  - Profil bez časového okna, profil neaktivní, sonda připojena - *profil je smazán*
  - Profil s časovým oknem, sonda připojena - *profil je smazán, stránky blokované profilem jsou povoleny*
  - Profil bez časového okna, profil aktivní, sonda odpojena - *smazání profilu není povoleno, je zobrazena chybová hláška*
  - Profil bez časového okna, profil neaktivní, sonda odpojena - *smazání profilu není povoleno, je zobrazena chybová hláška*, ve druhé verzi aplikace je profil smazán***
  - Profil s časovým oknem, sonda odpojena - *smazání profilu není povoleno, je zobrazena chybová hláška*
- Funkčnost tlačítka zpět

Nalezené problémy:

Nebyly nalezeny žádné problémy.