Výzkum mobilního malware pro Průmysl 4.0

Jan Kincl

Bakalářská práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Kincl**
Osobní číslo:	**A17053**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Prezenční**
Téma práce:	**Výzkum mobilního malwaru pro Průmysl 4.0**
Téma práce anglicky:	**Mobile Malware Research for Industry 4.0**

Zásady pro vypracování

1. Získejte vzorky mobilního malwaru.
2. Proveďte statickou i dynamickou analýzu získaných vzorků mobilního malwaru.
3. Zpracujte výsledky provedených analýz.
4. Identifikujte rysy, kterými se od sebe jednotlivé vzorky liší.
5. Najdete znaky, které jsou společné pro celou množinu zkoumaných vzorků nebo její podmnožiny (za předpokladu, že takové znaky skutečně existují).

Rozsah bakalářské práce:
Rozsah příloh:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

AU, Man Ho. Mobile security and privacy: advances, challenges and future research directions. Cambridge, MA: Elsevier, 2016. ISBN 978-012-8046-296.
DUNHAM, Ken, Shane HARTMAN, Jose Andre MORALES, Manu QUINTANS a Tim STRAZZERE. Android malware and analysis. Boca Raton: Auerbach Publications, [2014]. ISBN 14-822-5219-8.
DUBEY, Abhishek a Anmol MISRA. Android security: attacks and defenses. Boca Raton: CRC Press, c2013. ISBN 978-1-4398-9646-4.
JIANG, Xuxian a Yajin ZHOU. Android malware. New York: Springer, [2013]. SpringerBriefs in computer science. ISBN 978-1-4614-7393-0.
ELENKOV, Nikolay. Android security internals: an in-depth guide to Android's security architecture. San Francisco: No Starch Press, [2015]. ISBN 15-932-7581-1.
RAGGO, Michael T. Mobile data loss: threats and countermeasures. Waltham, MA: Syngress, [2016]. ISBN 01-280-2864-5.
DUNHAM, Ken. Mobile malware attacks and defense. Burlington, MA: Elsevier, c2009. ISBN 15-974-9298-1.

Vedoucí bakalářské práce: **Ing. Milan Oulehla**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **20. prosince 2019**
Termín odevzdání bakalářské práce: **15. května 2020**

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Ve Zlíně dne 20. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor;
- **že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.**

Ve Zlíně, dne 4.8.2020 Jan Kincl, v. r.

ABSTRAKT

Operační systém Android se vzhledem ke stále snadnější dostupnosti mobilních technologií stává velmi rychle od svého uvedení na trh v roce 2008 velice populárním. Zařízení fungující na bázi operačního systému Android pronikají nejen do osobní sféry formou chytrých telefonů, tabletů, domácích asistentů a jiných zařízení. Ale také do sféry firemní, jako pracovní nástroj sloužící zaměstnancům k přístupu do firemní infrastruktury, nebo jako nástroj monitoringu a řízení výrobních procesů. Z těchto důvodů se Android stává hlavním cílem tvůrců mobilního malwaru a je potřeba intenzivní snaha o zmapování bezpečnostních rizik s cílem vytvoření adekvátních bezpečnostních vrstev pro tuto platformu. Práce přináší unikátní data týkající se bezpečnostní situace spojené s neoficiálními úložišti, analyzuje činnost vybraných vzorků malware aplikací a na základě nalezených výsledků poskytuje důležitý základ pro další výzkum v oblasti bezpečnosti platformy Android.

Klíčová slova: Android, malware, statická analýza, dynamická analýza, vizualizace

ABSTRACT

Due to the ever-easier availability of mobile technologies, the Android operating system is quickly becoming very popular since its launch in 2008. Devices operating on the basis of the Android operating system are not only becoming part of personal life in the form of smartphones, tablets, home assistants and other devices, but also a part of companies, as a work tool used by employees to access the company´s infrastructure, or as a tool for monitoring and controlling production processes. Because of these reasons, Android has become the main target of mobile malware creators, and due to that, intensive efforts are needed to map security risks in order to create adequate security protection layers for this platform. This work presents unique data related to the security situation associated with unofficial files sources, analyzes the activity of selected malware applications samples and, based on the discovered results, provides an important basis for further research in the field of security of the Android platform.

Keywords: Android, malware, static analysis, dynamic analysis, visualization

Děkuji Ing. Milanu Oulehlovi, Ph.D. za velmi otevřený přístup, důsledné a motivující vedení při výzkumné činnosti, podnětné rady a konzultace a pomoc při formulaci bakalářské práce. Rovněž děkuji všem členům laboratoře penetračního testování PTLAB UTB za projevenou důvěru při přijetí do laboratoře a poskytnutí zázemí pro vykonávání výzkumné činnosti a prostředků pro spuštění webové vizualizace.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

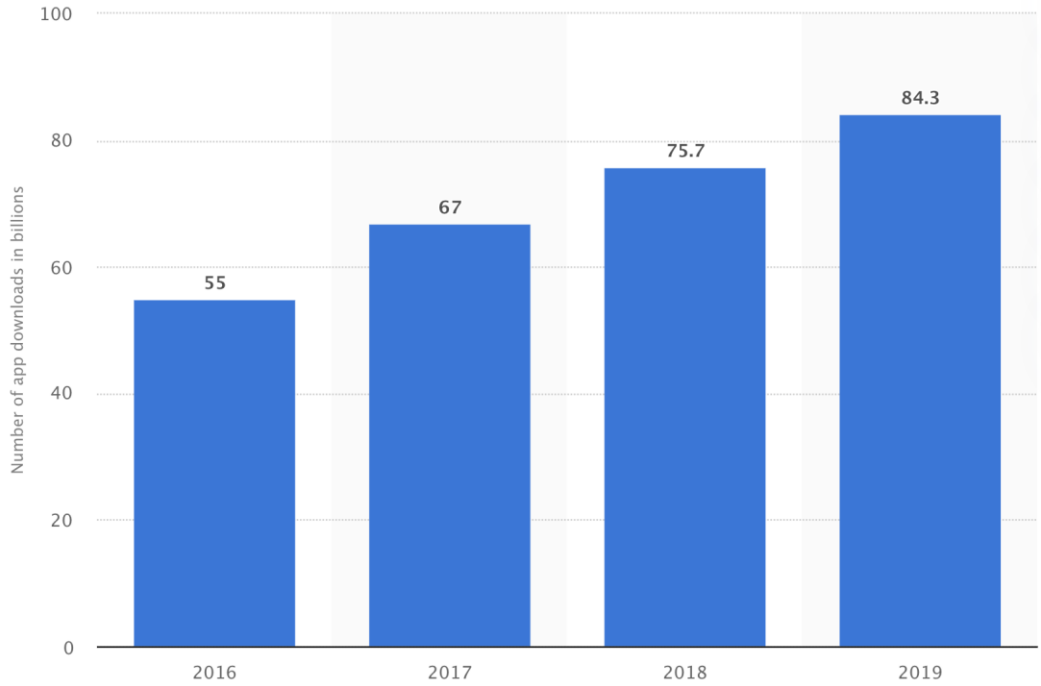
ÚVOD

Otázka bezpečnosti mobilních zařízení je v současné době velmi aktuální.

V globálním měřítku zažívají mobilní technologie obrovský rozmach. Mobilní zařízení, díky neustále se zvyšující dostupnosti, pronikají nejen do soukromé sféry formou osobních telefonů, tabletů nebo například chytrých hodinek, ale také do sféry profesní. Většina firem aktivně implementuje mobilní technologie jako součást své infrastruktury, z čehož pramení velmi citlivá povaha dat, pohybujících se napříč mobilní platformou.

Bohužel, v porovnání například s platformou osobních počítačů, není vývoj bezpečnosti mobilní platformy dostatečně rychlý a flexibilní, a není schopný udržet tempo s neustále se zrychlujícím rozvojem nových mobilních zařízení a technologií. Což často vede k nedostatečné bezpečnostní robustnosti a závažným bezpečnostním hrozbám.

V současné době nejrozšířenějším operačním systémem pro mobilní platformy je Android OS. První verze tohoto systému, označená jako Android 1.0, byla zveřejněna 23. 9. 2008 [1]. Android nejlépe zareagoval na vznikající prostor na světovém trhu. Díky odlišnému pojetí tvorby operačního systému a zařízení, na kterých je tento operační systém provozován, se Android velmi rychle stává populárním systémem, a to nejen z pohledu uživatelů ale i výrobců mobilních zařízení. Pro rozmach každého operačních systému je nutná dostatečně rozsáhlá základna uživatelských aplikací.



Obr. 1 Počty aplikací stažených uživateli z Google Play v letech 2016 až 2019 [2]

Z Obr. 1 je patrné, že operační systém Android tuto podmínku splňuje, neboť jen za rok 2019 si uživatelé stáhli 84,3 miliard aplikací. Všechna výše uvedená fakta mají za následek skutečnost, že podíl operačního systému Android na světovém trhu mobilních zařízení se v současné době pohybuje okolo 86 % [3].

S rostoucí popularitou systému Android přichází na řadu otázka bezpečnosti. Primárním cílem tvůrců mobilního malwaru je vždy platforma s největším tržním podílem. Vývoj škodlivého softwaru (dále jen malware) vyžaduje investice peněz a času, což znamená, že tvůrci nebudou plýtvat zdroji a zabývat se operačním systémem, s malou cílovou skupinou a tudíž s malou šancí návratnosti počáteční investice.

Vzhledem k rychlému rozmachu, se systém Android již po 4 letech, stává nejčastějším cílem tvůrců mobilního malwaru, kdy 79 % všech zaznamenaných útoků cílilo právě na tento systém. Viz výroční zpráva FBI [4].

Android od svého vzniku čelí stále se stupňujícímu počtu útoků. A tedy zhoršující se bezpečnostní situaci, kdy například v roce 2018 došlo k zachycení cca 4 milionů různých malwarů [5]. Firma Kaspersky ve své výroční zprávě [6] z roku 2018 popisuje neustále narůstající počet útoků na platformu Android. Současné predikce předpokládají další zhoršování této situace.

Dalším aktuálním problémem platformy Android je skutečnost, že více něž 40 % uživatelů (což odpovídá více než 1 miliardě uživatelů) používá Android verze 6.0 a starší. Společnost Google LLC koncem roku 2019 přestala dodávat bezpečnostní aktualizace pro výše uvedené verze. Z čehož plyne, že tato zařízení zůstávají nechráněné vůči novým hrozbám [7].

Současné predikce předpokládají, že škoda vzniklá vlivem malwaru napříč všemi platformami, dosáhne v roce 2021 hodnoty 6 bilionů dolarů [8]. Všechny výše uvedené skutečnosti naznačují nutnost intenzivního výzkumu v oblasti zabezpečení mobilní platformy.

Práce se snaží přispět ke zlepšení bezpečnostní situace tím, že představuje unikátní poznatky popisující metodiku tvorby mobilního malwaru. A v rámci práce byl rovněž vytvořen systém pro:

- Automatické získávání škodlivých aplikací z neoficiálních file-share serverů.
- Vyhodnocení bezpečnostní situace týkající se těchto serverů, systém vytváří unikátní statistiku popisující využití takovýchto serverů pro předpokládanou distribuci malwaru.

Na získané množině unikátních škodlivých aplikací byla provedena analýza s cílem odhalit charakteristické znaky indikující škodlivost aplikace a na základě získaných výsledků umožnit navrhnout další výzkum, navazující na světový vůzkum v oblasti detekce mobilního malware a přinést tak potřebné zlepšení bezpečnostní situace.

V rámci práce byla také navržena standardizovaná metodika pro analýzu mobilních aplikací v rámci mobilní sekce PTLAB UTB.

TEORETICKÁ ČÁST

1 MOBILNÍ MALWARE

Označení malware vzniklo spojením dvou anglických slov: malicious + software. V současná terminologii je malware odborným výrazem pro všechen škodlivý software [9].

V případě mobilního malware pak platí, že jeho cílem jsou mobilní zařízení, jako jsou chytré telefony, tablety apod. Malware obsahuje celou řadu různých kategorií škodlivého kódu. Od trójských koní, ransomwaru, virů a červů až po bankovní malware. Nejčastěji s cílem poškodit samotné zařízení nebo získat osobní data uživatele [10].

1.1 Historie mobilního malwaru

S nástupem mobilních zařízení a jejich vzrůstající popularitou dochází i ke změně zaměření útočných skupin, právě na tuto platformu. Dle přehledu historie mobilního malwaru zpracovaného v rámci průzkumu [11] Univerzity Britské Kolumbie, jedním z prvních škodlivých softwarů, které se objevily na mobilní platformě v globálním měřítku je Cabir [12]. Malware cílil na zařízení s operačním systémem Symbian a šířil se prostřednictvím Bluetooth.

Od této chvíle vznikají další, různé typy malwarů tvořených pro mobilní zařízení, často s cílem získávat citlivá data, nebo způsobit finanční újmu. Jedním z prvních škodlivých softwarů, cílících na operační systém Android byl dle reportu [13] firmy Kaspersky – SMS Trojan. K infikaci docházelo prostřednictvím aplikace přehrávače médií, kdy po instalaci Trojan začal odesílat SMS zprávy na čísla se zvýšenou sazbou, bez vědomí uživatele, čímž způsoboval nemalé finanční ztráty. Firma Kaspersky označila tento malware za první systematicky zacílený škodlivý software pro platformu Android. Společnost Kaspersky rovněž potvrzuje, že první izolované útoky na tuto platformu, byly zaznamenány již v předchozím roce 2009. Výše uvedené skutečnosti spolu s predikcí zvyšujícího se počtu útoků na mobilní platformu, vedly společnost Kaspersky k vývoji bezpečnostního software pro ochranu zařízení fungujících pod operačním systémem Android.

1.2 Typy mobilního malwaru

V rámci mobilního malwaru je rozlišováno, podobně jako u PC platformy, několik typů malwaru, lišících se svým cílem, nebo režimem činnosti.

Firma Kaspersky v jednom ze svých bezpečnostních shrnutí [14], odlišila několik populárních typů mobilního malwaru:

- **Bankovní malware**

  V případě bankovního malwaru se jedná o velmi nebezpečné techniky, pomocí kterých se útočníci snaží cílit na přístupové údaje uživatelů do služby internetových bankovnictví. Nejvíce zranitelnou skupinou jsou uživatelé, kteří ze svého mobilního zařízení aktivně vyřizují bankovní operace. Scénář útoku bývá takový, že po infiltraci do zařízení, dochází ke sběru přístupových údajů, které jsou následně odeslány na server útočníků.

  Ze statistik popisující využití uvedeného typu malwaru vyplývá stále vzrůstající míra využití tohoto typu malwaru. Zároveň firma Kaspersky v reportu potvrzuje, že v druhé polovině roku 2015 byla hrozba bankovního malwaru nejrychleji rostoucí v porovnání s ostatními typy.

- **Ransomware**

  Malware typu ransomware se nejdříve objevuje na PC platformě a se stejnou funkcionalitou se později přesunuje na platformu mobilní. Ransomware útoky vždy cílí na důležitá data, jako jsou dokumenty, fotografie apod. U těchto souboru následně dojde k jejich zašifrování. Tvůrci ransomware posléze zkontaktují uživatele a požadují výkupné za opětovné rozšifrování souborů.

  V případě nezaplacení výkupného dochází ke ztrátě dat. Zejména v průmyslovém sektoru, v případě napadení kriticky důležitých dat, se jedná o velice závažný problém. Z průzkumu [15] provedeného společností IDG (International Data Group) vyplynulo, že 74 % všech dotázaných firem zaznamenalo narušení vlastní bezpečnosti, které přímo souviselo s mobilní platformou. Mobilní malware je uváděn jako největší bezpečnostní problém.

- **Spyware**

  Obecně lze spyware označit jako software monitorující činnost uživatele. Na mobilních zařízení se nejčastěji jedná o monitoring aktivit uživatele, sledování pomocí vestavěných kamer zařízení a v nejhorším případě také sbírání přístupových údajů.

  Často dochází k připojení spywaru k legitimním aplikacím, kdy takovýto spyware, bez vědomí uživatele sbírá a odesílá data na pozadí a jeho činnost je nejčastěji odhalitelná pouze pozorováním poklesu výkonnosti telefonu, nebo odhalením prostřednictvím antivirových skenů.

- **MMS malware**

  V případě MMS malwaru se tvůrci pro šíření malwaru snaží zneužít MMS komunikaci mezi zařízeními. V roce 2015 odhalila firma Zimperium zranitelnost v systému Android, kdy na základě článku [16] popisujícího tuto zranitelnost, dochází k šíření malwaru mezi zařízeními pomocí MMS zpráv, a to bez nutnosti, aby uživatel zprávu otevřel. Tvůrcům tedy stačilo znát pouze telefonní čísla a na ta rozeslat malware, čímž byla příslušná zařízení infikována. Vzhledem k architektuře systému Android mohl škodlivý kód smazat zprávu, prostřednictvím které byl šířen. V takovém případě uživatel neobdrží upozornění na příchozí MMS zprávu a malware se šíří zcela nepozorovaně.
  V infikovaném zařízení pak malware mohl získávat vysoká oprávnění a přebírat kontrolu nad zařízením.

- **Adware**

  V začátcích byl adware poměrně neškodný. Jednalo se o malware, který na displeji mobilního zařízení zobrazoval reklamy na různé produkty, případně prováděl sběr dat o návštěvnosti webů apod. Jednalo se tedy spíše o aplikace snižující uživatelský komfort.

  Postupem času dochází ale k tvorbě agresivnějších aplikací (viz. článek [17]), které provedou tzv. root[1] zařízení bez uživatelova vědomí. Čímž dochází k postupnému transformaci adwaru na klasický

[1] Umožnění využívat oprávnění uživatele root v systému i jiným uživatelům a aplikacím

malware typu Trojan. Takto škodlivý software do telefonu následně stahuje legitimně vyhlížející aplikace, které obsahují škodlivou funkcionalitu a postupně dochází k infikování celého zařízení.

- **SMS Trojans**

  Tak jak bylo popsáno v odstavci popisující historii malwaru, software patřící do této skupiny zneužívá možnost odesílat zprávy na čísla se zvýšenou sazbou a přinášet tak zisk útočníkům. V horších případech dochází k napadání zpráv týkajících se přístupů k bankovním službám. Z článku [18] popisující tuto problematiku vyplývá, že malware v infikovaném zařízení prohledával příchozí zprávy a v případě, že se jednalo o SMS spojenou s bankovními službami, odesílal data zachycená v SMS do Číny. Zachycením a zneužitím obsahu SMS docházelo ke sběru potřebných informací pro odcizení a zneužití přístupu k bankovním službám.

2 OPERAČNÍ SYSTÉM ANDROID

Operační systém Android je v současné době nejvíce rozšířeným operačním systémem pro mobilní platformu. Využívají jej nejen chytré telefony, ale také tablety, chytré televize, nositelná elektronika a domácí asistenti.

2.1 Historie systému Android

Společnost Android Inc. byla založena v říjnu roku 2003 čtveřicí zakladatelů: Andy Rubin, Rich Miner, Nick Sears a Chris White [19].

Původním plánem společnosti bylo dle tvrzení [20] Andy Rubina vytvořit operační systém pro digitální kamery. Vzhledem k pomalému růstu odvětví digitálních kamer však společnost rychle přeorientovala svůj záměr. Novým cílem společnosti bylo vytvořit operační systém pro neustále narůstající trh mobilních zařízení, který bude schopný konkurovat tehdejším verzím operačních systémů Windows Mobile a Symbian.

V roce 2005 vstupuje na trh s mobilními zařízeními společnost Google LLC a v rámci tohoto kroku dochází k odkupu společnost Android Inc. O rok později dochází k představení prvního prototypu využívajícím operační systém Android, zatím bez implementace dotykového displeje [21].

V roce 2007 vstoupila na trh mobilních zařízení také společnost Apple, konkrétně uvedením prvního modelu telefonu iPhone. Uvedená skutečnost ovlivnila i vývoj operačního systému Android, který bylo nutné v reakci na vysokou úroveň konkurence přepracovat a také přidat do Android zařízení dotykový displej. První komerčně dostupné zařízení využívající systému Android byl telefon HTC Dream, který byl představen v září 2008 [22].

2.2 Přehled systému Android

Jednou z hlavních vlastností operačního systému Android je „Unix-like“ podstata tohoto systému. Android OS využívá LTS Linuxový kernel, který je modifikován pro účely mobilních zařízení. Další z výhod je, že společnost Google LLC se rozhodla zpřístupnit OS Android v rámci Android Open Source project. Hlavním vývojářem projektu je společnost Google LLC, ale vyvinutý systém a jeho zdrojový kód je volně přístupný zájemcům z celého světa, kterým je udělen přístup. V globálním měřítku to znamená, že existuje vždy základní verze Androidu, kterou si jednotliví výrobci mohou upravit, tak aby plně vyhovovala jejich zařízení. Zároveň mohou výrobci dodat své vlastní proprietární aplikace. V základu obsahuje

systém Android většinou proprietární aplikace společnosti Google LLC, jednotliví výrobci k těmto předinstalovaným mohou připojit vlastní aplikace, které budou k dispozici v rámci jejich modifikace systému.

Z hlediska samotného operačního systému pak platí, že podobně jako v linuxových systémech je datový prostor dělený na jednotlivé adresáře [23]. Kdy na rozdíl od desktopových verzí linuxových systémů uživatelé nemají root přístup k zařízení. Právě přístup k root oprávnění je častým cílem malwaru, za účelem převzít kontrolu nad zařízením.

Celková architektura operačního systému Android je vrstvená. Základní vrstvou je již zmiňované Linuxové jádro a následující vrstvy tvoří C a C++ knihovny využívané systémem, jako například knihovny pro přehrávání videí, databázové knihovny apod.

Nejvýše položenou vrstvou je pak vrstva aplikační, která tvoří rozhraní mezi zařízením a uživatelem. Samotné aplikace pro operační systém jsou vyvíjeny v jazyce Java, popřípadě v jazyce Kotlin. Po kompilaci aplikace vzniká instalační APK balíček, který je instalován do mobilního zařízení.

2.3 Rozšířenost systému Android

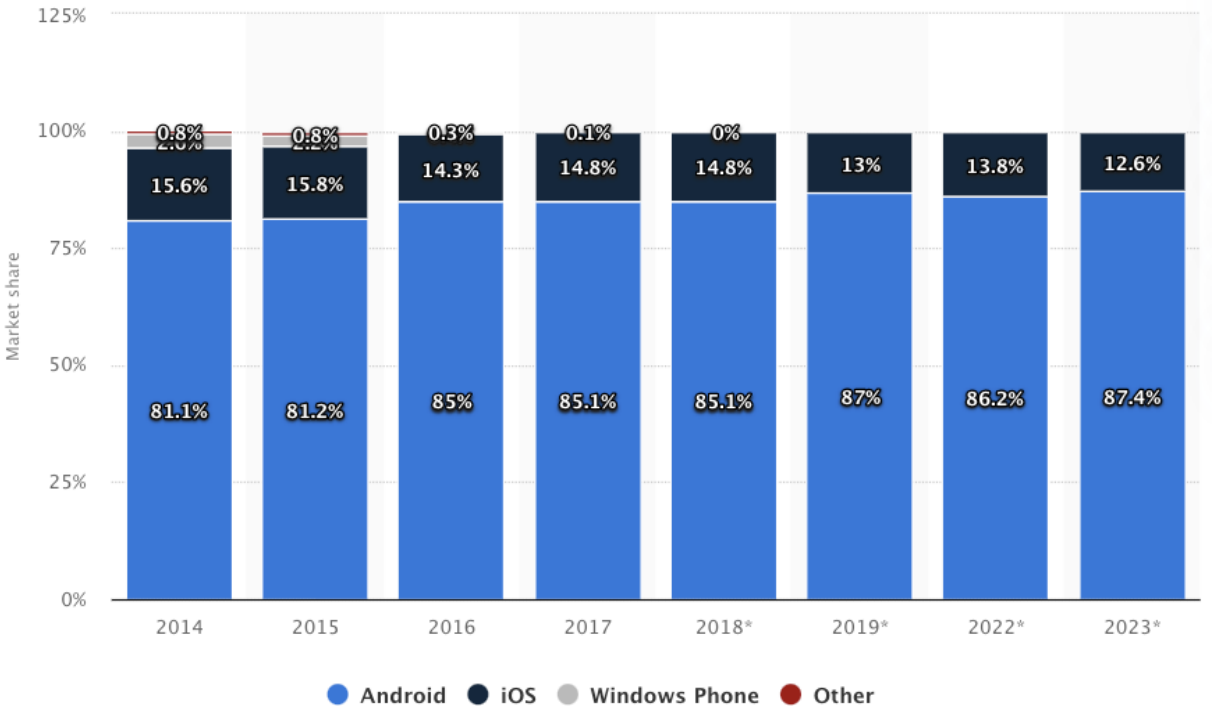
Android od svého uvedení na trh zažívá poměrně rychlý nárůst na globálním trhu. Vývojáři sami uváděli[20], že důvod, proč poskytují Android zdarma, je, aby se rozšířil na co největší počet zařízení po celém světě.

Z výsledků [24] společnosti Canalys, popisující prodeje z druhého čtvrtletí roku 2009 vyplývá, že Android tvoři 2,8 % podíl světového trhu. Což vzhledem ke krátké době od uvedení předpovídalo nadějný nárůst.

Ve druhém čtvrtletí následujícího roku už Android představoval 10 % světového trhu a zároveň předstihl platformu Windows Mobile. Do konce roku 2010 Android zaznamenává nejvýraznější nárůst prodejů a stává se tak světově nejrozšířenějším mobilním operačním systémem a předstihl tak i do té doby hojně rozšířený operační systém Symbian.

Android se během dvou let stává nejrozšířenějším mobilním operačním systémem [25] s předpokladem dalšího nárůstu podílu na světovém trhu. Na konci roku 2011 představuje více než polovinu světového trhu a o další rok později již celé ¾ světového trhu [26].

Graf na Obr. 2 znázorňuje neustále vzrůstající trend v zastoupení OS Android. V současné době je jeho podíl více než 85 % s očekáváním mírného nárůstu v dalších letech.



Obr. 2 Vývoj podílu operačního systému Android na světovém trhu [27]

Android ovšem nevládne pouze trhu s mobilními zařízeními, ale vzhledem k obrovskému počtu zařízení (přes 2,5 mld. [28]) se Android stává celosvětově dominantním operačním systémem.

Výše popsané skutečnosti podporuje statistika [29], která uvádí, že od roku 2018 platí, že více než 50 % všech dotazů na webové stránky, je prováděno z mobilních zařízení.

3 MALWARE PRO OPERAČNÍ SYSTÉM ANDROID

Jak bylo popsáno v předchozích kapitolách, mobilní zařízení zažívají stálý rozmach. Operační systém Android se stává nejvíce rozšířeným na trhu mobilních zařízení. Bohužel mohutný rozmach této platformy s sebou nese i zvyšující se bezpečnostní rizika.

Tvůrci škodlivého softwaru velmi rychle zaměřili svoji pozornost na nově vznikající mobilní zařízení a vcelku logicky pak primárně na nejvíce dominantní platformu, tedy Android.

V následující časové ose je popsán vývoj bezpečnostní situace spojené se systémem Android:

2008	• Září 2008 - představení první komerční verze systému Android a oznámení zařízení HTC Dream - viz. oddíl 2.1 Historie Android OS. • Ještě před příchodem zařízení HTC Dream dochází k vytvoření prvních hrozeb pro systém Android, konkrétně čtyř malware aplikací [30], jejichž hlavním účelem nebylo škodit, ale upozornit na rizika spojená s používáním systému Android.
2009	• Vznikají první falešné bankovní aplikace, které jsou vyvíjeny anonymními uživateli. Účelem těchto škodlivých aplikací bylo získat přístupové údaje uživatelů [31].
2010	• Android se postupně stává nejvíce využívaným systémem na trhu s mobilními zařízeními, čímž k sobě výrazně přitahuje pozornost tvůrců mobilního malwaru z celého světa. • Vniká první organizovaný útok na systém Android ve formě SMS Trojan (viz. 1.2 Typy mobilního malwaru). Aplikace maskovaná

jako přehrávač médií, odesílala SMS zprávy na čísla se zvýšenou sazbou, čímž získává peníze pro útočníky.

- Objevuje se jeden z prvních případů modifikace oficiálních aplikací, byla infikována legitimní aplikace hry Had, aplikace monitorovala údaje o poloze a odesílala tato data na server. Ke sledování ale byla potřebná další Android aplikace na jiném zařízení a aplikace Hada musela být registrována a propojena s touto aplikací. Nejednalo se tedy o závažnou hrozbu, protože útočník potřeboval fyzický přístup k napadenému zařízení [32].
- Nejzávažnější hrozbou tohoto roku, se stává aplikace Geinimi. Aplikace byla prvním opravdu sofistikovaným malwarem, který využíval tzv. anti-analysis metody. Malwary dále využívaly, šifrování síťové komunikace, obfuskaci zdrojového kódu, kdy takto upravený kód, je jen velmi obtížné pochopitelný při čtení. Aplikace také rovněž plně využívala možnost upravit legitimní aplikace, vložit do nich škodlivý obsah a na rozdíl od předchozích útoků, už nevyužívala pro svou distribuci oficiální obchod s aplikacemi Android Market (v dnešní době znám pod jménem Google Play), ale byla šířena prostřednictvím neoficiálních webových úložišť a obchodů s aplikacemi. Takto unikala oficiálním kontrolám a byla šířena mezi uživatele pod názvy oficiálních aplikací, které byly napadeny [30].

2011

- Po vzoru aplikace Geimini, dochází začátkem roku k nárůstu počtu útoků stejného typu. Tedy k napadání oficiálních aplikací a modifikaci jejich obsahu. Dochází k vložení škodlivého obsahu a následnému znovu sestavení aplikace (tzv. repackage).
- Takto upravené aplikace jsou pak šířeny prostřednictvím neoficiálních zdrojů a úložišť.

- V reakci na narůstající hrozby je uživatelům doporučováno instalovat aplikace pouze z oficiálního distribučního kanálu.
- Později téhož roku dochází k jednomu z prvních napadení oficiálního Android Market, kdy ve více než 50 aplikacích je nalezen malware DroidDream [33]. Malware následně sbíral citlivá data uživatelů jako je IMEI a IMSI. Uvedený malware zároveň obsahoval funkcionalitu schopnou způsobit root zařízení a tím získat kontrolu nad zařízením. Odhaduje se, že před odstraněním aplikace došlo k 50 000 – 200 000 tisícům stažení [34]. Google vydává opravný patch, který odstraní veškeré změny provedené malwarem. O pár dní později je detekováno infikování i této opravné patch aplikace [35]. Vzniklý malware byl šířen opět prostřednictvím převážně čínských neoficiálních zdrojů a docházelo ke komunikaci na pozadí s řídícím serverem.
- Dochází k tvorbě dalších SMS Trojanů. Jejich častým cílem byla skrytá komunikace s čínskými telefonními čísly se zvýšenou sazbou.
- V průběhu roku, také dochází k dalšímu útoku cílícímu na oficiální distribuční obchod Android Market a šíření malwaru touto cestou mezi desetitisíce uživatelů. Jedná se převážně o sběr citlivých dat o uživateli a jeho zařízení a instalaci dalších škodlivých aplikací do zařízení. Malware byl vytvořen stejnými vývojáři jako předchozí DroidDream [36].

2012

- Bezpečnostní situace systému Android se s jeho narůstající popularitou stále zhoršuje. Na základě analýzy [37] společnosti

	GDATA dochází v tomto roce ke vzniku více než 200 tisíc nových malware aplikací.
2013	• FBI potvrzuje ve své výroční zprávě [38], že Android zůstává primárním cílem tvůrců mobilního malwaru, kdy 79 % všech zachycených vzorků, míří právě na tuto platformu. Dalších 19 % pak představují malware aplikace pro platformu Symbian, zbylé 2 % pak ostatní systémy včetně iOS. • Zároveň dochází k prudkému nárůstu počtu nově vzniklých škodlivých aplikací jejichž počet dosahuje téměř 1,2 milionu [37].
2014	• Dochází ke vzniku dalších 1,5 milionu malware aplikací [37].
2015	• Z šetření [39] University of Cambrige vyplývá, že v průměru přes 85 % všech Android zařízení jsou zranitelné vůči minimálně 11 kritickým zranitelnostem. • Zároveň bezpečnostní laboratoře Zimperium odhalují hrozbu Stagefright, uvedená hrozba byla schopná v době jejího odhalení aktivně ovlivnit 95 % Android zařízení [40]. • Opět dochází k nárůstu počtu nově vzniklých škodlivých aplikací na 2,3 milionu nově vzniklých malwarů v tomto roce [37].

2017	• Firma Kaspersky eviduje 66,4 milionu útoků způsobených malwarem na mobilní platformu [41].
2018	• Společnost Kaspersky zaznamenala zdvojnásobení počtu malware útoků oproti předchozímu roku, na celkový počet 116,5 milionu [41]. • Dochází k nárůstu počtu útoků typu banking Trojan – je zaznamenáno přes 150 tisíc různých instalačních balíčků obsahujících tento malware [41]. • Tvůrci mobilního malwaru využívají pokročilejší distribuční metody jako je SMS spam, nebo DNS hijacking[2]. Z výše uvedeného vyplývá stále se zvyšující potřeba spolehlivých bezpečnostních prvků pro mobilní platformu. Vznikají doporučení udržovat operační systém mobilního zařízení stále aktualizovaný a neinstalovat aplikace z neoficiálních distribucí [41].
2019	• Android jako operační systém funguje na obrovském množství zařízení, velkého počtu výrobců. Aktivní zařízení se od sebe i přes základ v systému Android liší. Jednak modifikací systému výrobcem, kdy ve zmiňovaném výzkumu [39] University of Cambrige bylo zjištěno, že zranitelnost Android zařízení se liší v závislosti na výrobci. V druhé řadě pak na aktivních zařízení po celém světě funguje několik desítek verzí systému Android v různém stádiu poskytování aktualizací. • Výše zmíněné skutečnosti výrazně ztěžují možnosti návrhu ochranných prvků pro Android zařízení. Vzhledem k obrovské

[2] Útočníci modifikují DNS dotazy a přesměrovávají oběti na své stránky za účelem šíření malwaru

	variaci aktivních zařízení a verzí, nejsou všechna citlivá na stejné zranitelnosti a je obtížné vydávat unifikované aktualizace. • Koncem roku Google oznamuje, že přestává podporovat zařízení pracující na Androidu verze 6.0 a starším. Následná analýza [42] zjišťuje, že se jedná o více než 40 % aktivních zařízení, což odpovídá více než 1 miliardě uživatelů po celém světě. Takto vzniklá skupina už na svých zařízeních neobdrží bezpečnostní aktualizace a zůstává tak nadále nechráněná vůči nově vznikajícím útokům.
2020	• Bezpečnostní situace týkající se mobilní platformy se stále zhoršuje. • Odhaduje se, že je blokováno zhruba 24 tisíc škodlivých aplikací denně. Často se jedná o malware cílící na nechráněná starší zařízení [43]. • 70 % dotázaných společností potvrzuje, že nejsou dostatečně připravené na případné bezpečnostní napadení [43][15]. • Předpokládá se další zhoršování bezpečnostní situace v průběhu následujících let – celková výše finančních škod způsobených malwarem napříč platformami se pro rok 2021 odhaduje na 6 bilionů USD$ [43].

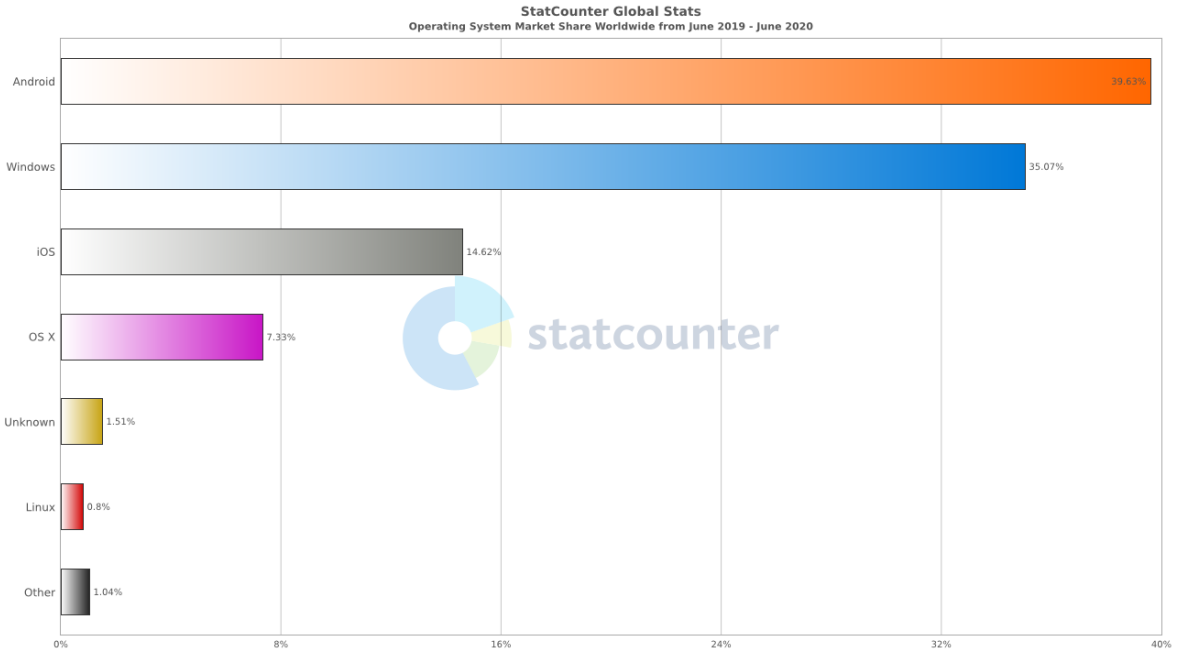
Tabulka 1 Časová osa vývoje malwaru pro Android

Vzhledem k popsaným skutečnostem se práce v jednom se svých bodů zaměřuje právě na analýzu neoficiálních možností distribuce aplikací, z uvedeného kontextu jasně vyplývá, že neoficiální distribuční kanály jsou často využívány pro distribuci škodlivého softwaru. Z tohoto důvodu v rámci práce byla stanovena míra rizika spojená s využitím českých file-share serverových úložišť pro získání aplikací.

Rovněž se práce v reakci na uvedený kontext a absenci ochranných vrstev pro Android zařízení zaměřuje na analýzu zachycených vzorů malwaru s cílem odhalit a navrhnout další výzkumné strategie pro oblast tvorby ochranné vrstvy systému Android.

4 PRŮNIK OS ANDROID DO FIREMNÍHO PROSTŘEDÍ

Predikce dalšího nárůstu tržního podílu OS Android se neprojevuje pouze na poli mobilních zařízení. Vlivem snadné dostupnosti operačního systému Android v rámci AOSP, dochází k rozšíření tohoto systému i mimo mobilní zařízení a Android se stává nejpoužívanějším operačním systémem na světě, viz Obr. 3



Obr. 3 Podíl světového trhu pro jednotlivé operační systémy [59]

Zařízení se systémem Android v dnešní době neslouží pouze pro osobní použití jako zdroj zábavy, fotoaparát nebo nástroj k procházení internetu, ale postupně se z těchto zařízení stávají prostředky pro práci.

Velké množství uživatelů je v dnešní době schopno realizovat část svých pracovních úkolů přímo v telefonu nebo tabletu, čímž dochází k postupnému snižování potřeby osobních počítačů. V reakci na uvedenou skutečnost dochází ze strany řady firem a průmyslových subjektů k podpoře integrace Android zařízení do své infrastruktury. Dochází k vývoji mobilních aplikací, které uživatelům umožní přístup k datům společnosti a vznikají specializovaná průmyslová zařízení se systémem Android.

Obecně můžeme nově vznikající trend rozdělit na dva směry:

- Android zařízení pro administrativní využití
- Specializovaná Android zařízení pro průmysl

4.1 Android Enterprise

Společnost Google LLC zareagovala na vzrůstající zájem o integraci Android zařízení pro pracovní využití a vytvořila iniciativu Android Enterprise. Prostřednictvím této iniciativy se snaží do nově vznikajících verzí operačního systému Android implementovat potřebné nástroje pro bezpečnou integraci do firemního prostředí.

4.1.1 Trend Bring your own device (BYOD)

Trend využívat své vlastní zařízení i pro vykonávání administrativních pracovních úkonů odstartoval průnik mobilních technologií do firemního prostředí [44]. Vzhledem ke zvyšujícímu se počtu mobilních zařízení dochází k postupnému vývoji mobilních aplikací pro systém Android, které dříve byly dostupné pouze prostřednictvím osobních počítačů. Vznikem těchto aplikací mají uživatelé možnost část své práce provádět na mobilních zařízeních, která jsou v jejich osobním vlastnictví.

V rámci integrace do firemního prostředí je umožněn přístup k firemní infrastruktuře z mobilních zařízení. Zaměstnanci mohou přistupovat do firemních databází, pracovat s pracovními maily (např. Microsoft 365) přímo ze svých vlastních zařízení, popřípadě jinak pracovat s daty společnosti.

Protože se nejedná o firemní zařízení, často na tato zařízení nejsou kladeny žádné bezpečnostní požadavky. Mnohdy je možné instalovat aplikace s firemním přístupem na zařízení, která nemají naistalován antivirový program, nejsou dostatečně zabezpečena nebo se jedná o tzv. rooted zařízení (tj. telefony nebo tablety u kterých došlo k odblokování přístupu k root oprávněním systému Android). Z pohledu bezpečnosti se pak pro společnost jedná o obrovské riziko spojené s možným únikem dat, napadením firemní sítě, rozšíření malwaru typu ransomware do firemní infrastruktury apod.

Z výše uvedených důvodů se společnost Google LLC snaží v rámci iniciativy Android Enterprise [45] navrhnout nové standarty a možnosti bezpečného využití Android zařízení ve firemním prostředí.

Společnost Google LLC představuje 3 scénáře [45] integrace mobilních zařízení do firemní oblasti:

4.1.1.1 Zařízení vlastněná uživateli

V rámci tohoto scénáře se jedná o klasický případ trendu BYOD a společnost integruje do své infrastruktury osobní zařízení uživatelů.

Od verze Android 5.1 je do operačního systému implementovaná možnost vytvářet v zařízení osobní a tzv. pracovní profil. Dochází k rozdělení datového prostoru zařízení na dvě nezávislé oblasti. Část zařízení, která je připadá na osobní profil, slouží uživateli jako klasické osobní zařízení, uživatel není ve využití nijak limitován a správu nad profilem a jeho daty provádí sám. Soukromí uživatele a možnosti osobního využití zařízení zůstávají nezměněny.
Naopak, část zařízení odpovídající pracovnímu profilu je izolována od datového prostoru uživatele a plně pod kontrolou společnosti. Takto jsou oddělena a chráněna pracovní data a aplikace, mohou být implementovány prvky ochrany nad pracovním profilem a společnost má možnost omezit přístup k datům v rámci zařízení.

Zařízení vlastněná uživateli nejsou v žádném případě vhodná pro integraci do průmyslového prostředí.

4.1.1.2 Zařízení vlastněná společností

Jak už z názvu vyplývá při implementaci tohoto scénáře jsou zaměstnancům poskytovaná zařízení, která vlastní společnost.

V rámci uvedeného scénáře existují dvě varianty**:**

- zařízení s úplnou správou
- zařízení s úplnou správou a pracovním profilem

V prvním případě společnost zaměstnanci poskytuje kompletně spravované zařízení, které slouží pouze pro vykonávání pracovní činnosti. V zařízení je systém nastaven tak, že neumožňuje jiné využití. Často dochází k využití tzv. Managed Google Play, jedná se upravenou verzi obchodu s aplikacemi. Tento obchod je dostupný pouze v rámci mobilních zařízení společnosti a seznam dostupných aplikací je spravován společností. Do zařízení lze nainstalovat pouze schválené, nebo neveřejné firemní aplikace.

V druhém případě, tedy zařízení s úplnou správou a pracovním profilem je zařízení opět rozděleno na více profilů. Společnost stejně jako v prvním případě má plnou kontrolu nad zařízením, ale rozdělením zařízení do dvou profilů je umožněno také osobní využití zařízení.

V rámci pracovního profilu mohou být vyžadována přísnější bezpečnostní opatření a restrikce, čímž je zajištěna bezpečnost firemních dat. Nad uživatelským prostorem pak mohou být aplikována mírnější pravidla. Nicméně nad celým uživatelským prostorem existuje další vrstva ochrany, která ovlivňuje i uživatelský prostor.

4.1.1.3 Zařízená vlastněná společností pro cílené využití

V tomto případě se jedná o využití specializovaných průmyslových zařízení se systémem Android. Problematika bude popsána v rámci sekce 4.2 Integrace operačního systému Android do průmyslu.

4.2 Integrace operačního systému Android do průmyslu

Popularita systému Android pro využití v průmyslových zařízeních narůstá zejména z důvodu otevřeného přístupu ke zdrojovému kódu v rámci AOSP. Díky tomu lze operační systém modifikovat a upravit pro konkrétní využití. Dochází k implementaci systému Android ve formě embedded zařízení, průmyslových počítačů, prostředků pro monitorování a řízení průmyslových procesů apod.

4.2.1 Android embedded

Stejně jako většina současných operačních systému pro mikropočítače, i Android pracuje na procesorech typu ARM. Díky možnosti libovolně modifikovat operační systém pro konkrétní činnost, proniká Android i do kategorie embedded zařízení [51][52].

Jako hlavní důvody [53] proč využívat systém Android pro tvorbu embedded zařízení je uváděno:

- Snadný vývoj uživatelského rozhraní, zároveň vysoká úroveň podpory dotykového ovládání.
- Rozsáhlá vývojářská komunita.
- Široká podpora výrobců jednotlivých periferií (LCD displajů, GPS antén, WiFi/BT chipset) a tedy přístupnost korektních ovladačů zařízení. Vzhledem k využití linuxového jádra je možné využívat i linuxové ovladače.
- Vzhledem k rozšířenosti Android open-source projektu vzniká velké množství vývojářských nástrojů, které usnadňují vývoj a testování aplikací [54].

Nejpoužívanější jsou v současné době zdarma dostupné nástroje společnosti Google LLC – Android Studio, SDK a emulátory [55].

Díky uvedeným výhodám je možné rychle vyvíjet a navrhovat Android embedded řešení [54].

Mezi nevýhody [54] spojené s využíváním OS Android v embedded zařízeních pak patří:

- Vyšší náročnost systému na výpočetní prostředky – Android jako systém je poměrně obsáhlý a vyžaduje ke svému fungování minimálně 512 MB RAM a procesor o frekvenci 1 GHz nebo vyšší. Oproti jednoúčelovým mini zařízením má Android vyšší spotřebu elektrické energie. Tato nevýhoda je ovšem vyvážena intuitivním dotykovým ovládáním Android zařízení, kdy většinu uživatelů není nutné seznamovat s jeho principy. Zároveň v porovnání s jinými embedded zařízeními, má Android přehlednější možnosti vizualizace. Také je předpokládáno, že trend nositelné elektroniky, zejména pak požadavky na vyšší výdrž těchto zařízení, povedou k další „miniaturizaci“ systému Android.
- Operační systém Android v embedded zařízeních nelze automaticky aktualizovat na nejnovější verze, z důvodu modifikace systému pro pracovní úkon.

4.2.2 Průmyslové počítače

Android proniká také do sféry odolných průmyslových zařízení. Známý výrobce těchto zařízení Honeywell nabízí [47] mobilní průmyslové počítače, fungující na systému Android.

Výhoda těchto zařízení je jejich snadná mobilita a odolnost pro použití v průmyslovém prostředí. Zařízení slouží pro monitorování vozových parků, evidovaní skladovaných položek, jsou využívána v rámci poštovních služeb pro správné doručování. Zařízení lze vybavit RFID čtečkami, optickými kamerami podporující technologie zpracování obrazu (tzv. image processing) a v případě dohody se společností Google mohou být implementovány proprietární systémové aplikace.

4.2.2.1 Pokladny s OS Android

V rámci možných modifikací sytému Android lze v dnešní době pořídit pokladnu [48] pracující na systému Android, která může mít integrovaný platební terminál, tiskárnu účtenek a je schopná evidovat tržby.

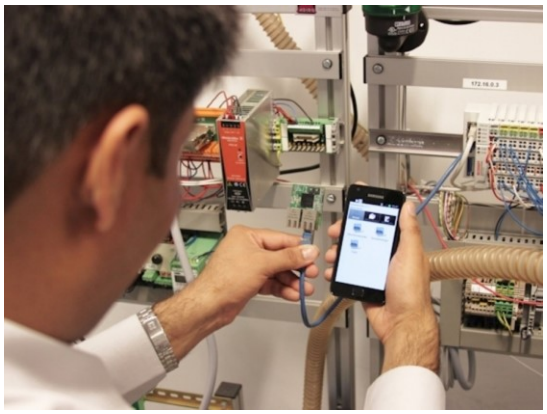
4.3 Android SCADA

Neustálý rozvoj automatizace a pozvolný nástup Průmyslu 4.0 vede k pronikání systému Android přímo do výrobních procesů. Vznikají SCADA systémy podporující systém Android nebo aplikace pro průmyslové využití.

Pojmem SCADA označujeme software, který umožňuje sběr dat z průmyslových zařízení a jejich řízení. Jedním ze systémů, který funguje na Android zařízeních je TeslaSCADA [60]. Systém představuje rozhraní mezi strojem a člověkem (HMI) a umožňuje sledování a řízení průmyslových PLC zařízení přímo z chytrého zařízení fungujícím pod systémem Android. Mezi uváděná využití pak patří:

- Management
  Mobilní zařízení jsou použita pro monitorování výrobního procesu.
- Údržba
  V případě poruchy, nebo chyby vzniklé na PLC zařízení, je možné toto zařízení přenastavit z mobilního zařízení.
- Řízení
  Operátoři výrobních strojů mohou měnit výrobní parametry a činnost stroje bez nutnosti fyzického přístupu.
- Automatizace domácností

  Mobilní zařízení je možné využít i pro řízení automatizovaných prvků domácností.



Obr. 4 Operátor využívající Android zařízení pro diagnostiku [62]

Obr. 5 Operátor využívající zařízení Android pro nastavení stroje [58]

Dalším z příkladů využití systému Android pro monitorování výrobních procesů je aplikace [49] SBC Micro Browser společnosti Saia Burgess Controls [50]. Společnost se zabývá výrobou a implementací průmyslových řídících systémů a zmiňovaná aplikace umožňuje uvedení do provozu, servis a vizualizaci procesů implementovaných zařízení a strojů.

S Průmyslem 4.0 bývá často spojován koncept IoT, neboli Internet of Things. Průmysl 4.0 předpokládá integraci konceptu IoT, kdy dojde k propojení jednotlivých zařízení do inteligentní sítě a sdílená data mezi zařízeními budou sloužit k automatizovanému řízení a regulaci.

V současné době začínají být Android zařízení spojována i s tímto konceptem. V prvním případě v souvislosti s inteligentními domácnostmi, pro které existují zařízení Android (tzv. Domácí asistent), které jsou schopné řídit činnost kompatibilních spotřebičů, vytápění, osvětlení nebo zabezpečení objektu [56].

V druhém případě opět v souvislosti s integrací Android zařízení do konceptu IoT a Průmyslu 4.0. Například cílem platformy ThingOS [57] je vytvoření řešení, v rámci kterého bude možné vzájemně propojit jednotlivá mobilní zařízení, chytré senzory, aktuátory (apod.) do jedné vrstvy a sjednotit tak výrobní proces.

Současné trendy předpokládají zvyšující se míru využití operačního systému Android i v rámci průmyslových zařízení. V souvislosti s aktuální bezpečnostní situací popsanou v rámci kapitoly 3, vnikají důvodné obavy týkající se integrace mobilních zařízení do průmyslu.

Jsou zaznamenány případy, při kterých dochází k útoku na průmyslový objekt s využitím malwaru. Jedním z nejznámějších případů je malware Stuxnet [61] z roku 2010. Jednalo se o první zachycený malware, který záměrně cílil na průmyslová zařízení. Malware infikoval řídící počítače s OS Windows (v roce 2010 měl operační systém Windows zhruba 90 % podíl světového trhu), na kterých byl instalován řídící software pro PLC Siemens. Prostřednictvím tohoto řídícího softwaru malware upravil činnost PLC, čímž došlo ke zničení řízeného stroje.

Se současnou orientací tvůrců malwaru primárně na mobilní platformu Android a rozšířenosti tohoto systému, je potenciální riziko podobných útoků prostřednictvím platformy Android velmi vysoké. Častým cílem průmyslových útoků je zničení výrobního zařízení, nebo průmyslová špionáž.

Průmyslová zařízení a firmy jsou také populárním cílem pro útoky typu ransomware. V případě napadení jednoho z kontrolních mobilních zařízení, by mohlo dojít k rozšíření malwaru po celém výrobním objektu a k následnému ochromení.

V zájmu rozvoje Průmyslu 4.0 a s tím spojené integrace mobilních zařízení, je nutné se intenzivně věnovat problematice mobilního malwaru, konkrétně identifikovat nové hrozby pro platformu Android. Dostatečně zabezpečit platformu zařízení Android, čímž bude zajištěna nutná ochrana pro jednotlivá odvětví, do kterých budou mobilní zařízení integrována a umožnit tak bezpečnou integraci těchto zařízení do průmyslového prostředí a pro samotný Průmysl 4.0.

5 ANALÝZA ANDROID APLIKACÍ

V reakci na bezpečnostní situaci systému Android se práce v e praktické části věnuje získávání a následné analýze vzorků mobilního malwaru pro systém Android. Obecně neexistuje standardizovaná metodika a způsob, jakým analyzovat mobilní aplikace a spolehlivě detekovat mobilní malware. Většina výzkumných subjektů i samotní tvůrci malwaru si chrání podstatu svého know-how souvisejícího s analýzou a tvorbou mobilního malwaru. Pro provádění kvalitní analýzy, je nejprve potřeba porozumět struktuře mobilních aplikací.

5.1 Struktura Android aplikace [63]

Aplikace pro operační systém Android mohou být v současné době napsané v jazycích Java, C++ a nově vyvíjeném jazyku Kotlin. Po napsání zdrojových kódů dochází k tzv. kompilaci zdrojových souborů aplikace a vzniká archiv s příponou .apk (tzv. Android application package). Jedná se o soubor, který je následně využíván v Android zařízení pro instalaci aplikace – princip lze přirovnat k .exe souborům v OS Windows.

V operačním systému Android každá aplikace figuruje jako samostatný uživatel s vlastním privátním datovým prostorem. Každá aplikace má také svůj vlastní virtual machine, sloužící při běhu programu (jednotlivé aplikace jsou odděleny) [64]. Právě kvůli uvedenému rozdělení se malware často snaží cílit na získání přístupu k root oprávnění telefonu, což mu umožní manipulovat s datovým prostorem ostatních aplikací.

Omezení činnosti jednotlivých aplikací v zařízeních s Android OS je realizováno prostřednictvím oprávnění. V systému existují chráněné komponenty (například možnost přistupovat na internet), které jsou chráněny jednotlivými oprávněními. V případě, že aplikace vyžaduje přístup k této komponentě, musí zažádat o příslušné oprávnění v souboru manifestu (viz. Oddíl 5.1.3 Soubor AndroidManifest.xml)

5.1.1 Komponenty aplikace

Aplikace běžící pod operačním systémem Android se skládají ze 4 komponent. Jednotlivé vrstvy představují rozhraní mezi uživatelem a aplikací, mezi různými aplikacemi, nebo mezi aplikací a komponenty systému.

5.1.1.1 Aktivity aplikace

Pod pojmem aktivita aplikace se rozumí rozhraní mezi uživatelem a aplikací. Zjednodušeně se vždy jedná o aplikační logiku jedné obrazovky, kterou uživatel vidí (aktivita, která je na popředí) a se kterou interaguje. Google uvádí příklad [63]: obrazovka se seznamem příchozích emailů představuje aktivitu. Další aktivita je spuštěna v případě kliknutí na tlačítko nový email (zobrazí se nová obrazovka pro vytvoření emailu, tedy spustí se další aktivita).

Aktivity navzájem spolupracují, ale jsou vzájemně nezávislé. V případě, kdy to aplikace umožňuje, mohou být některé její aktivity spouštěny i z jiných aplikací externě. Například u fotografie je možné zvolit sdílení za pomocí emailu, v reakci na tento požadavek se spouští externí aktivita emailové aplikace pro vytváření emailu. Po odeslání emailu se spuštěná aktivita uzavírá a uživatel se vrací na aktivitu fotogalerie. Pro správné fungování je potřeba zajistit zachování posloupnosti spouštěných aktivit a předávání informací mezi aktivitami.

5.1.1.2 Service

Tato komponenta zajišťuje správné fungování procesů na pozadí. Nejjednodušším příkladem služby je například fungování přehrávače hudby, kdy uživatel může pracovat s telefonem, ale zároveň poslouchat hudbu z aplikace přehrávače.

Služby jsou rozdělovány na vázané a nevázané:

- Nevázané služby figurují jako proces na pozadí. Slouží k vykonání nějaké činnosti a nejsou závislé na ostatních komponentách. V operačním systému pak taková služba existuje do momentu vykonání svého cíle. Výjimkou je případ nedostatku operační paměti nebo zdrojů systému. V takovém případě může režie OS tyto služby ukončit a uvolnit paměť pro potřeby aplikace běžící na popředí, systémových komponent apod. Časté využití nevázaných služeb je pro synchronizaci a stahování dat na pozadí, přehrávání hudby apod. Častou snahou malware aplikací, je vytvoření nevázaných služeb, prostřednictvím kterých je na pozadí telefonu realizována škodlivá funkcionalita.
- Vázané služby představují server v modelu client-server. Tento typ služeb ostatním komponentám slouží pro vykonání nějaké činnosti, přijímání a odesílání upozornění a k realizaci komunikace mezi procesy. Vázaná služba pak v operačním systému

existuje do chvíle, dokud je využívána některou z komponent. Poté, co služba již není využívána, je ukončena režii OS.

5.1.1.3 Broadcast receiver

Tato komponenta tvoří vrstvu primárně mezi událostmi v systému a aplikací. Operační systém a aplikace mohou přijímat a vysílat broadcast zprávy. Tyto zprávy v rámci operačního systému vznikají v reakci na nastalou událost. Například při dokončení bootu zařízení, při spuštění nabíjení, dosažení hodnoty nastaveného času pro alarm apod. Prostřednictvím komponenty broadcast receiver může být v aplikacích realizována obsluha takto vzniklých zpráv, bez nutnosti aktivního čekání aplikace na událost.

5.1.1.4 Content provider

V předchozí části bylo objasněno, že každá aplikace v systému Android má svůj privátní datový prostor a funguje odděleně. Komponenta Content provider pak slouží jako prostředek pro sdílení dat a informací mezi jednotlivými aplikacemi.

5.1.2 Zdrojové soubory aplikace

Data nutné pro kompilaci .apk instalačního balíčku, se neskládají pouze z aplikační logiky. V rámci těchto dat jsou rozlišovány také tzv. resources, neboli zdrojové soubory aplikace.

Jednotlivými zdroji rozumíme nejčastěji obrázky, videa, ikony a textové řetězce použité v aplikaci. Díky oddělení zdrojů od aplikační logiky, je snadno možné upravit data, která aplikace využívá, bez nutnosti upravovat zdrojový kód a vytvářet nový APK balíček. Dalším využitím je možnost přepínat jazyk aplikace, kdy operační systém na základě lokálního nastavení může přizpůsobit jazyk aplikace bez nutnosti změny aplikační logiky.

5.1.3 Soubor AndroidManifest.xml

Tento soubor je naprosto kritický pro správné fungování aplikace. Jedná se o konfigurační soubor, ve kterém musí být tvůrcem aplikace specifikovány veškeré komponenty a požadavky aplikace. Přesněji zde musí být nadefinovány výše zmíněné komponenty aplikace: activity, service, broadcast receiver a content provider.

Veškeré komponenty, které jsou vytvořené v rámci zdrojového kódu, ale nejsou deklarovány v souboru manifestu, nejsou pro systém viditelné a v případě pokusu o jejich spuštění dochází k vyvolání bezpečnostní výjimky a pádu aplikace.

Další součástí souboru manifestu je konfigurace požadavků aplikace. Například specifikovat nutnou úroveň softwarového vybavení zařízení, nebo určit, které hardwarové periferie musí v zařízení být dostupné, aby aplikace mohla fungovat (například je pro aplikaci nutná přítomnost fotoaparátu).

V neposlední řadě se v souboru manifestu nachází jedna z nejpodstatnějších informací o aplikaci – seznam všech oprávnění, která aplikace vyžaduje pro své fungování. Jak již bylo zmíněno výše, aplikace pro přístup k chráněným částem operačního systému musí zažádat o příslušná oprávnění. Pokud by došlo k pokusu o vykonání nepovolené činnosti ze strany aplikace, dojde k vyvolání bezpečnostní výjimky a ukončení aplikace.

5.2 Reverzní inženýrství

Základem analýzy mobilních aplikací pro platformu Android je tzv. metoda reverzního inženýrství [65]. Podstata této metody spočívá v procesu dekompilace (= zpětné rozložení aplikace na jednotlivé prvky), získání zdrojů aplikace a jejich analýza. Pro aplikaci metod reverzního inženýrství a následnou analýzu mobilních aplikací, existují specializované nástroje.

5.2.1 Nástroj APKTOOL

Nástroj APKTOOL [66] je velmi populární nástroj mezi analytiky mobilní aplikací, umožňující reverzní inženýrství Android aplikací prostřednictvím CLI příkazů. Nástroj je vytvořený v jazyce Java a je dostupný zdarma.

Proces rozložení aplikace na jednotlivé prvky se nazývá dekompilace, během něj dochází k rozložení .apk souboru na jednotlivé stavební prvky, se kterými je možné dále pracovat.

Nástroj také poskytuje možnost opětovné tvorby .apk souboru z jednotlivých zdrojových souborů. Tento proces je označován pojmem kompilace.

5.2.2 Nástroj Dex2Jar

Dekompilací nástrojem APKTOOL získáváme zdrojový kód ve formátu smali. Formát tohoto jazyka je velmi podobný formátu jazyka assembleru. Provádění analýzy činnosti aplikace, je v tomto jazyce velmi obtížné.

Z tohoto důvodu vniká nástroj dex2jar [67], tento nástroj umožňuje převod .dex souborů obsažených v rámci .apk balíčku.

Samotný převod je opět inicializován prostřednictvím příkazu CLI, kdy ze vstupního .apk balíčku vznikne .jar archiv, ve kterém jsou zdrojové kódy převedeny do formátu .class jazyku Java.

5.2.3 Nástroj JD-GUI

Dalším nástrojem vhodným pro analýzu mobilních aplikací je JD-GUI [68], neboli Java decompiler. Jedná se o volně dostupný nástroj.

Java decompiler umožňuje otevření .jar souboru vytvořeného pomocí nástroje dex2jar a následné zobrazení .class souborů v čitelné formě. Mezi hlavní výhody jednoznačně patří snadná čitelnost kódu (neboť kód je v jazyce Java) a také vytvoření závislostí mezi jednotlivými třídami. Analytik prostřednictvím těchto vazeb může vyhledávat výskyt volání jednotlivých metod a provádět analýzu.

5.2.4 Nástroj JADx

Další populární možností pro analýzu mobilních aplikací je volně přístupný nástroj JADx [69]. Jedná se o pokročilejší nástroj, který kombinuje funkcionalitu předchozích tří nástrojů do jedné aplikace.

Pomocí nástroje JADx je tedy možné pracovat přímo s .apk soubory. Dochází k dekompilaci aplikace a extrakci souborů potřebných pro analýzu (například AndroidManifest.xml). A zároveň proběhne i převod aplikační logiky do jazyka Java.

V rámci nástroje pak jsou opět tvořeny reference na základě volání metod a jsou implementovány i jednoduché možnosti automatické deobfuskace. Pomocí JADx lze vyhledávat místa využití konkrétní metody, popřípadě spouštět vyhledávání klíčových slov v rámci dekompilované aplikační logiky [70].

5.2.5 Nástroj JEB Decompiler

Profesionální, placený nástroj JEB Decompiler [71] je v současné době nejpokročilejší platformou vytvořenou pro testování mobilních aplikací. Vzhledem k nepřetržitému vývoji a podpoře nástroje firmou PNF software, je možné pomocí tohoto nástroje provádět analýzu i nejnovějších aplikací běžících pod operačním systémem Android.

Aplikace opět umožňuje zpracovávat přímo jednotlivé .apk soubory. V rámci dekompilace jsou získány jednotlivé zdroje, soubor manifestu, informace o certifikačních autoritách a klíčích spojených s aplikací a zároveň je převedena aplikační logika do jazyka Java.

V JEB Decompileru jsou implementovány metody automatické deobfuskace získaných zdrojových kódů. Analytik má rovněž možnost libovolně přejmenovávat jednotlivé metody a proměnné, čímž může manuálně deobfuskovat kód. V rámci zdrojového kódu pak lze vytvářet komentáře, vyhledávat prvky na základě klíčových slov a převádět číselné hodnoty mezi jednotlivými formáty zápisu. Software umožňuje využívat skripty a v neposlední řadě lze provádět debugging analyzované aplikace na testovacím zařízení, včetně možnosti krokovat činnost aplikace a grafického znázornění pozice ve zdrojovém kódu.

5.3 Metody analýzy a detekce malwaru v mobilních aplikací

Rizika spojená s tvorbou malwaru pro OS Android jsou stále vyšší, a jak předchozí kapitoly vysvětlují, v dnešní době riziko nehrozí pouze uživatelům, ale také firmám, domácnostem a výrobním zařízením. Na tuto zhoršující se situaci se snaží reagovat výzkumné týmy po celém světě. Byla zmíněna potřeba vytvoření spolehlivého klasifikátoru a systému schopného odhalit škodlivé mobilní aplikace pro Android platformu.

Obecně se v technické praxi a v pracích věnující se analýze mobilních aplikací, právě za účelem detekce malwaru, rozlišuje několik různých přístupů k analýze aplikací.

5.3.1 Statická analýza

Při provádění statické analýzy se zkoumají se jednotlivé zdrojové soubory, ikony aplikace, prochází se soubor AndroidManifest.xml a v neposlední řadě se prochází také zdrojový kód. Důležité je, že při statické analýze nedochází ke spouštění aplikace na testovacím zařízení a analytik pouze na základě výše popsaných souborů určuje chování aplikace a tím odhaluje potenciální malware.

V práci [72] Pekingské univerzity je popsána metoda detekce mobilního malwaru na základě oprávnění, které aplikace požaduje a použitých metod ve zdrojovém kódu. Tato analýza je prováděna na množině škodlivých aplikací získaných pro výzkumné účely ze služby VirusShare [73]. Jedná se o stránku poskytující škodlivé mobilní aplikace pro výzkumné účely na základě oficiální žádosti

Analýzou souboru AndroidManifest.xml a prvků zdrojového kódu, dochází k rozeznání často se opakujících vzorců v rámci množiny škodlivých aplikací. Tato data následně slouží jako vstupní vektor pro Naive Bayesův klasifikátor a dochází tak k probabilistickému rozpoznání malware aplikací.

Podobný přístup pak aplikuje práce [74] pracovníků z University of Electronic Science and Technology of China, práce se rovněž zabývá problematikou detekce mobilního malwaru na základě využitých oprávnění a volaných metod ve zdrojovém kódu, na rozdíl od předchozí práce využívá klasifikátor typu Random forrest a dochází tedy k propojení metod statické analýzy a metod uměle inteligence.

Hlavní výhodou statické analýzy je její nenáročnost na výpočetní výkon, a tedy snadná implementovatelnost jak do mobilních zařízení, u kterých je velmi žádané efektivní nakládání s výpočetním výkonem mobilního zařízení, tak jako součást serverové infrastruktury (například pro kontrolu aplikacích nahrávaných na distribuční platformy).

Další výhodou je možnost propojit statickou analýzu s metodami umělé inteligence s cílem klasifikovat škodlivé aplikace. V rámci praktické části práce je popsán proces statické analýzy s cílem odhalit a popsat funkcionalitu zachycených škodlivých aplikací a nalézt znaky charakteristické pro mobilní malware. Jedná se o znaky, které by bylo možné využít v rámci dalšího výzkumu ve spolupráci s Laboratoří umělé inteligence (AILAB UTB [79]) pro vytvoření detektoru mobilního malwaru.

Na rozdíl od uvedených výzkumů, bakalářská práce nespoléhá na veřejně dostupný malware. Pro účely výzkumu byl vytvořen systém, který na vybraných neoficiálních webových úložištích je schopen vyhledávat aplikace pro operační systém Android a takto získávat množinu aplikací k analýze. Výsledkem je unikátní množina škodlivých aplikací. Pro odlišení benigních a malware aplikací, byla využita služba VirusTotal [75].

5.3.1.1 VirusTotal

Společnost VirusTotal je označována jako vysoce důvěryhodný zdroj identifikace malware aplikací [76]. Prostřednictvím svých webových stránek sjednocuje desítky anti-virových enginů, které provádějí bezpečnostní analýzu.

Soubor, který má být analyzován, je nahrán prostřednictvím webového rozhraní, popřípadě využitím tzv. VirusTotal API (viz. Praktická část práce). Nahraný soubor je následně otestován každým z dostupných anti-virových enginů. Hlavním výsledkem analýzy je tzv. virus-ratio, jedná se o číslo znázorňující poměr mezi počtem testů, které soubor vyhodnotily jako malware a celkovým počtem provedených testů ve tvaru:

počet testů detekujících hrozbu / celkový počet provedených testů

Na základě virus-ratia je možné určit potenciální nebezpečnost aplikace.

5.3.2 Dynamická analýza

Opakem statické analýzy je analýza dynamická, při použití této metody dochází ke spouštění analyzované aplikace na testovacím zařízení, kdy se monitorují jednotlivé procesy v rámci zařízení, změny datového prostoru a síťová komunikace.

Práce [77] popisující využití dynamické analýzy ve spojením se strojovým učením popisuje dva rozdílné přístupy k dynamické analýze.

V prvním případě práce popisuje využití nástrojů, které generují pseudo-náhodný vstup do zařízení, jako jsou kliknutí, potažení nebo například skrolování displeje. Nevýhodou náhodných vstupů je riziko ovlivnění stavu testovaného zařízení, kdy může například dojít k náhodnému vypnutí internetového připojení apod. Z tohoto důvodu, může docházet ke snižování přesnosti detekce a ovlivnění výsledků testů.

Zmiňovaná práce proto implementuje i druhý přístup, kdy nejprve proběhne statická analýza zdrojového kódu a určení, který uživatelský vstup je relevantní ve vztahu k jednotlivým aktivitám aplikace. A následně je vytvořen model konečného automatu aplikace, ve kterém aktivity aplikace představují jednotlivé stavy a vstup pro dynamickou analýzu je generován přesně pro potřeby analyzované aplikace. Při použití výše uvedené metody výzkumný tým zaznamenal nárůst v přesnosti výsledků.

Práce se opět nezaměřuje na tvorbu vlastního datasetu z aktuálně dostupných hrozeb na internetu a spoléhá na předem připravené datasety mobilního malwaru.

Nevýhodou dynamické analýzu a implementovaného přístupu je výpočetní náročnost. Protože musí být neustále aktivní prostředky realizující monitorování změn způsobených aplikací a zároveň musí být aktivní jednotlivé nástroje, které simulují interakci s aplikací. V porovnání s testováním, které se zakládá na poznatcích a metodách statické analýzy je dynamická analýza a její testy časově náročná a vysoce závislá na komplexnosti testované aplikace.

Dynamická analýza je tedy méně vhodná pro implementaci v mobilních zařízení.

5.3.3 Hybridní analýza

Jak bylo naznačeno, v praxi lze pro účely analýzy kombinovat metody statické a dynamické analýzy. Společná práce [78] čínských a amerických výzkumníků popisuje hybridní analytickou metodu, ve které dochází ke kombinaci prvků z prací, které byly popsány v předchozích dvou oddílech. Na testované množině jsou vyhledávány společné znaky

v rámci statické analýzy, opět zejména v souboru manifestu a volání jednotlivých metod ve zdrojovém kódu. Zároveň však dochází k vyhodnocování dynamické analýzy. Nakonec, kombinací výsledků dochází ke klasifikaci malwaru.

Obecně lze říci, že kombinací statické a dynamické analýzy je možné získat přesnější výsledky. Vyšší analytická přesnost je ovšem vyvážena vyšší výpočetních a časovou náročností.

V rámci práce je dynamická analýza implementovaná jako forma ověření výsledků získaných statickou analýzou.

6 CÍLE PRÁCE

V reakci na zhoršující se bezpečnostní situaci mobilní platformy, si práce klade za cíl získat poznatky použitelné pro výzkum v oblasti mobilního malwaru, běžícího pod operačním systémem Android. Práce se rovněž snaží přispět ke zlepšení metodik penetrační laboratoře PTLAB UTB [80]. Uvedené metodiky se používají pro výzkumné i komerční testování mobilních aplikací.

1. **ZÍSKEJTE VZORKY MOBILNÍHO MALWARE**

   Na rozdíl od ostatních prací věnujících se analýze mobilního malwaru, které byly popsány v oddílu 5.3, se práce snaží získat vlastní datovou sadu vzorků mobilního malwaru pro analýzu.

   Na základě předběžného výzkumu byla jako zdroj mobilních aplikací zvolena česká file-share serverová úložiště, u nichž je předpoklad, že jejich prostřednictvím dochází k neoficiálnímu šíření mobilního softwaru.

   Jedním z hlavních cílů je navržení a sestrojení automatizovaného systému schopného systematicky procházet vybrané file-share servery a provádět analýzu nalezených mobilních aplikací (APK balíčků). Automatizovaný systém by měl mít tyto funkce:

   - periodické procházení vybraných úložišť a získávání nových vzorků,
   - analýza získaných vzorků (virus-ratio),
   - identifikace potenciálně nebezpečných vzorků mobilních aplikací,
   - webová vizualizace, zobrazující aktuální bezpečnostní situaci spojenou s využíváním těchto zdrojů.

2. **PROVEĎTE STATICKOU I DYNAMICKOU ANALÝZU ZÍSKANÝCH VZORKŮ MOBILNÍHO MALWARU**

   V rámci získané datové sady bude vybrána množina vzorků, které mají analytický potenciál. Následně bude na vybraných vzorcích provedena statická analýza (metody byly popsány v oddíle 5.3.1).

Vyšetření bude zahrnovat tyto tři oblasti:

- analýza souboru AndroidManifest.xml,
- analýza zdrojových souborů aplikace,
- analýza zdrojového kódu za účelem odhalení škodlivé činnosti aplikace.

Výsledky získané statickými metodami budou následně ověřeny dynamickou analýzou na testovacím zařízení.

3. ZPRACUJTE VÝSLEDKY UVEDENÝCH ANALÝZ, 4. IDENTIFIKUJTE RYSY, KTERÝMI SE OD SEBE JEDNOTLIVÉ APLIKACE LIŠÍ, 5. IDENTIFIKUJTE SPOLEČNÉ ZNAKY

Na základě výsledků získaných statickou analýzou budou identifikovány nebezpečné rysy společné v rámci množiny získaného mobilního malwaru, pomocí těchto rysů pak budou aplikace rozřazeny do jednotlivých podmnožin.

V rámci každé podmnožiny bude pro náležící aplikace popsána jejich činnost v mobilním zařízení.

Z dat získaných analýzou zdrojových kódů budou určeny metody využívané při tvorbě mobilního malwaru a techniky šíření těchto škodlivých aplikací.

V rámci identifikace společných znaků je cílem získané aplikace seskupit do podmnožin o podobné nebezpečnosti. Seskupení bude probíhat na základě výsledků získaných ze služby VirusTotal a pro jednotlivé podmnožiny bude stanovena míra jejich distribuce.

Práce si klade za cíl zvýšení povědomí o problematice malwaru pro platformu Android, proto budou získané výsledky publikovaná formou webové vizualizace. Uvedená vizualizace by měla být sestavena tak, aby ji mohla jako bezpečnostní nástroj využívat i laická veřejnost. Přístupnost výsledků a informací by mohla vést k větší obezřetnosti uživatelů a snížení míry využití neoficiálních aplikací z file-share serverů, čímž by mohlo dojít ke zlepšení bezpečnostní situace na platformě Android.

Na základě výsledků získaných statickou analýzou proběhne stanovení scénářů pro další postup při vyšetřování mobilního malwaru, včetně navržení dalšího výzkumného postupu v rámci pracoviště PTLAB UTB, nejlépe ve spojení s dalšími laboratořemi UTB FAI.

Navržené postupy práce s mobilními aplikacemi by měly sloužit k rozvoji mobilní sekce laboratoře PTLAB UTB pro další výzkumnou a komerční práci na platformě mobilních aplikací.

PRAKTICKÁ ČÁST

7 ZÍSKÁVÁNÍ KOLEKCE MOBILNÍHO MALWARU

Jedním z cílů této práce je popsat bezpečnostní situaci na neoficiálních úložištích a zmapovat tak míru rizika, kterému jsou prostřednictvím neoficiálních zdrojů vystaveni uživatelé OS Android. Z těchto zdrojů získat dostupné mobilní aplikace a vytvořit tak unikátní množinu dat pro následnou analýzu.

V následujících podkapitolách bude krok po kroku představen postup při tvorbě unikátního systému, který je schopen procházet vybrané zdroje a získávat dostupné mobilní aplikace.

7.1 Výběr zdrojového úložiště

Na základě historického vývoje a skutečnosti, že webová úložiště často bývají zneužívána pro šíření škodlivého softwaru, bylo v prvním kroku vytváření automatického systému pro analýzu zvoleno několik českých file-share úložišť, které z důvodu zachování anonymity nemohou být jmenovány.

Na těchto serverech jsou dostupné .apk instalační balíčky Android aplikací. Zároveň přístup k těmto souborům z pravidla nebývá nijak omezen a potenciální škodlivost nahraných souborů není prověřována. Zmíněná skutečnost opět podporuje možnost šířit prostřednictvím těchto zdrojů škodlivý software mezi uživatele.

Dalším zdrojem se kterým systém v současné době systém pracuje jsou torrentová úložiště. V poslední době dochází k opětovnému nárůstu jejich popularity, hlavně z důvodu postupné eliminace světový file-share serverů v důsledku změn legislativy a prohraných soudních procesů.

Oproti klasickým file-share serverovým úložištím se torrentová liší v metodice ukládání souborů. Zatímco na file-share serveru je soubor fyzicky uložen, na torrentových serverech je poskytován pouze odkaz, který umožňuje následné stažení souborů ze sítě peer-to-peer[3] od ostatních uživatelů. Obecně se torrentová úložiště považují za velmi nebezpečný zdroj, hlavně kvůli absenci jakékoliv regulace obsahu, který se v rámci sítě vyskytuje a uživatelé jsou tak opět vystaveni nebezpečí spojenému se stahováním neznámých souborů.

[3] V síti peer-to-peer jsou data roztroušená mezi zařízeními v rámci sítě a stažení souboru probíhá získáním a složením těchto souborů prostřednictvím vhodného programu.

7.2 Analýza činnosti úložiště

Po vybrání úložišť, která mají být prověřena, následuje další fáze přípravy. Je potřebné identifikovat režim činnosti každého zvoleného úložiště.

Je potřebné získat informace o způsobu uložení balíčků na úložišti, jakým způsobem server úložiště reaguje a odpovídá na požadavek pro stažení souboru a jakým způsobem probíhá vyhledávání souborů v rámci webových stránek.

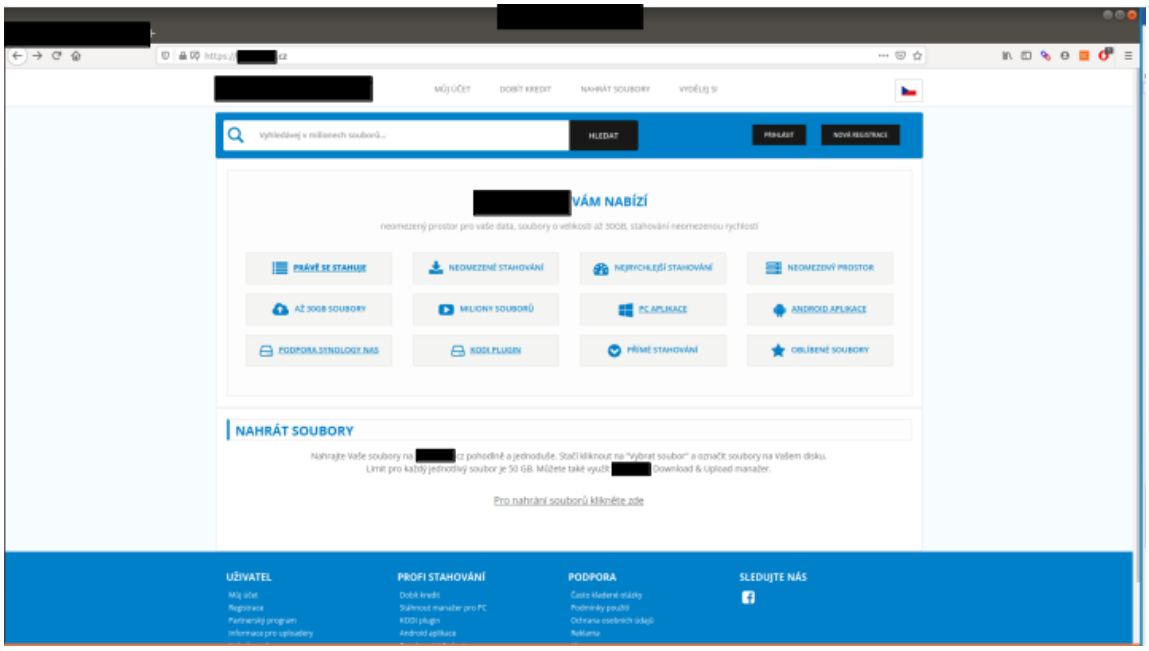
Jakmile zmapujeme režim činnosti úložiště, jsme schopni v dalších krocích zautomatizovat tuto činnost a pomocí programu realizovat komunikaci se serverem a získávání příslušných souborů.

7.3 Získání seznamu souborů uložených na úložišti

Na základě zmapování režimu činnosti úložiště jsme schopni vytvořit program, který bude na serveru schopný vyhledat konkrétní typ souborů. Každý uložený soubor na úložištích má svoji vlastní „domovskou stránku“, ze které je následně realizován požadavek na stažení souboru.

Pro strojovou realizaci tohoto procesu je v jazyce Python vytvořen tzv. web-crawler, tedy program, který dokáže zpracovávat HTML kód požadované stránky, modifikovat a otevírat URL adresy a přijímat odpovědi z takto otevřených stránek.

Činnost web-crawleru bude vysvětlena na několika obrázcích:



Obr. 6 Domovská stránka jednoho ze zvolených úložišť [zdroj vlastní]

Na Obr. 6 je zobrazená domovská stránka zvoleného úložiště. Vyhledávání souborů probíhá zadáním klíčového slova do vyhledávacího pole. V našem případě klíčové slovo: .apk

V rámci předchozího bodu, bylo zjištěno, že vyhledávání souborů lze realizovat změnou URL adresy z původní:

https:// [redacted].cz/

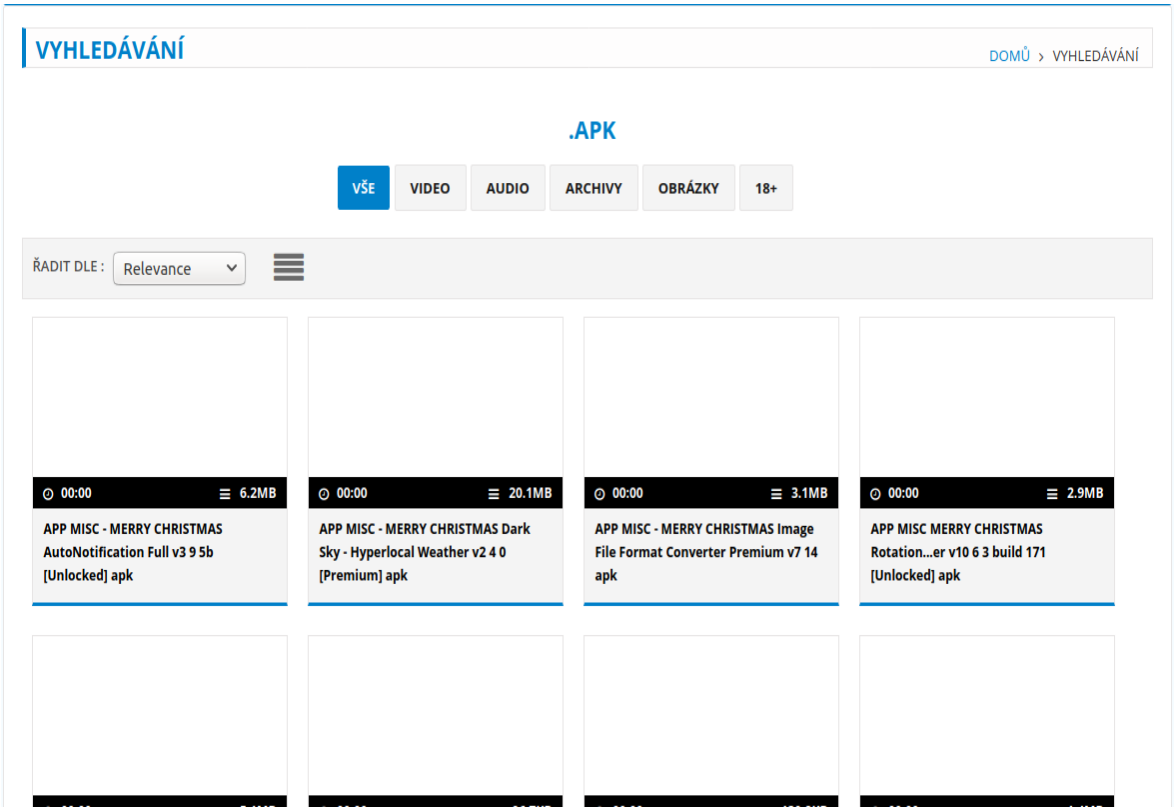
na:

https:// [redacted].cz/.apk/s

kdy je patrné, že klíčové slovo je vloženo do URL adresy s následným příznakem /s (= search).

Na základě toho je možné v rámci programu upravit URL adresu a spustit vyhledávání. Jako výsledek se následně vrací stránka s výsledky všech nalezených souborů (viz. Obr. 7), jak již bylo řečeno, každý soubor má svou vlastní podstránku.

Modifikací URL adresy získáváme stránku s výsledky:



Obr. 7 Stránka s výsledky hledání po zadání klíčového slova .apk [zdroj vlastní]

Po získání stránky s výsledky se provádí poslední krok, a to automatická identifikace a extrakce odkazů pro podstránky jednotlivých souborů.

V jazyce HTML, ve kterém jsou tvořeny webové stránky, je odkaz identifikován označením <href>. V rámci web-crawleru v jazyce Python, který zpracuje HTML kód stránky s výsledky, dochází k vyhledávaní tohoto klíčového a k následné extrakci a zápisu výsledných odkazů do textového souboru.

Výsledný textový soubor s odkazy na podstránky jednotlivých souborů vypadá nějak takto:

Obr. 8 Ukázka výsledného textového souboru s odkazy na stránky jednotlivých souborů na úložišti [zdroj vlastní]

Pro jednotlivá úložiště se výše uvedené metody liší. Každé úložiště proto musí být analyzováno odděleně a pro každé je následně vytvořen specializovaný program, který realizuje uvedený postup. Některá úložiště také implementují prvky, které zabraňují využití programů pro stahování dat. Poté je nutné během identifikace činnosti úložiště také navrhnout metodu pro překonání těchto ochranných prvků.

7.3.1 Torrentová úložiště

V případě extrakce odkazů a práce s torentovými úložišti bývá postup podobný výše uvedené metodice pro práci s HTML kódem. Rozdílem je, že pro stahování souborů ze sítě peer-to-peer slouží tzv. magnetic link.

V HTML kódu s výsledky vyhledávání souborů na webových stránkách torrentových úložištích dochází k dodatečné identifikaci těchto odkazů pomocí klíčového slova: „magnet“ a tyto odkazy jsou následně obdobně ukládány do textového souboru.

7.3.2 Úložiště využívající JavaScript

Pokud úložiště pro tvorbu svých webových stránek využívá JavaScript, nelze použít výše představný postup a klasický HTML web-crawler. JavaScript využívá aplikaci prohlížeče pro renderování výsledných stránek a klasický HTML web-crawler z takovéto stránky vrátí pouze prázdný HTML kód, protože render neproběhl.

Jako řešení tohoto problému je potřeba program v Pythonu rozšířit o funkcionalitu webového prohlížeče. Za tímto účelem se využívá knihovna Selenium, která umožňuje spouštět virtuální webový prohlížeč a programovat jeho činnost v rámci zdrojového kódu. Takto je možné simulovat činnost reálného uživatele (jako jsou kliknutí, čekání na načtení stránky, skrolování apod.) pomocí programu.

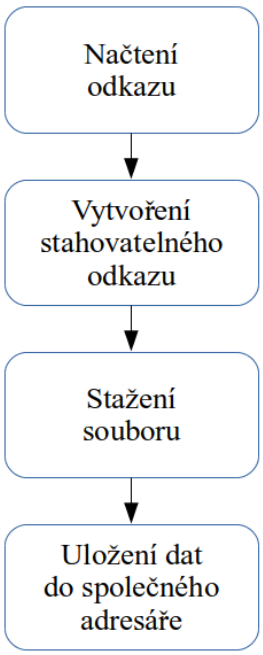
Celý proces začíná spuštěním virtuálního okna prohlížeče bez grafického vykreslování, následně, na základě identifikace činnosti úložiště, dochází k vyhledání příslušných souborů modifikací URL adresy. Následným otevřením této adresy ve virtuálním prohlížeči je renderován HTML kód pomocí JavaScriptu. Program pak čeká na úplné vykreslení stránky a následně stahuje HTML kód pro zpracování a uložení odkazů do textového souboru.

7.4 Automatizované stahování souborů

Po získání odkazů na podstránky jednotlivých souborů, dochází ke spuštění jejich stažení.

Po předchozích krocích víme, jakým způsobem dochází k tvorbě stahovatelného odkazu – tedy odkazu, při jehož otevření se spouští stahování chtěného souboru.

Pro stahování jednotlivých souborů je opět pro každé úložiště vytvořena metoda v jazyce Python, která postupně zpracovává předchystané odkazy a stahuje jednotlivé soubory (Viz. Obr. 9).



Obr. 9 Zjednodušené schéma procesu stahování souborů [zdroj vlastní]

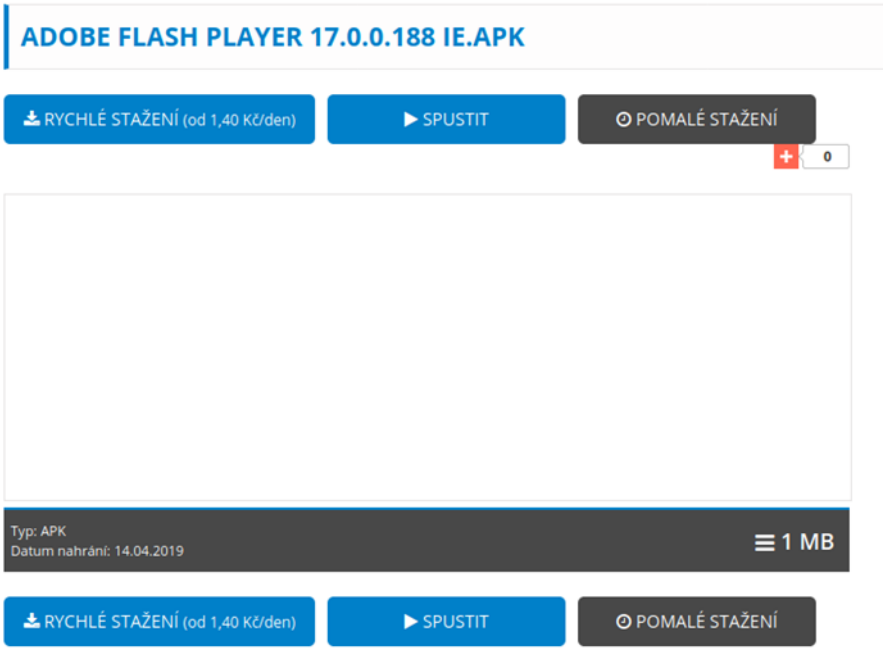
7.4.1 Vytváření stahovatelných odkazů

Analýzou každého úložiště je potřeba zjistit způsob, jakým je realizováno stahování jednotlivých souborů. V případě, že je stahování realizováno pomocí modifikace URL adresy, nebo generováním statického odkazu pro stažení souboru, lze tyto odkazy modifikovat a vytvářet v rámci web-crawleru vytvořeného v jazyce Python.

Mějme odkaz vedoucí ke stránce souboru na úložišti:

https:// [redacted] .cz/11536995/adobe-flash-player-17.0.0.188-ie.apk

Na stránce souboru se nachází několik odkazů:



Obr. 10 Ukázka podstránky úložiště pro konkrétní soubor [zdroj vlastní]

Předchozí analýzou úložiště bylo odhaleno, že odkaz pro bezplatné (pomalé) stažení souboru obsahuje ve zdrojovém kódu HTML klíčové slovo **/free**.

V rámci web-crawleru je tedy načten HTML kód každé ze stránek souborů a pomocí klíčových slov <href> a /free je vyhledán příslušný stahovatelný odkaz:

```
for HTML in soup.find_all('a'):                                    # načítání HTML kódu stránky
    odkaz_na_strance = HTML.get('href')                            # vyhledání klíčového slova href
    if odkaz_na_strance:                                           # v případě nalezení výsledku
        if '/free' in odkaz_na_strance:                            # ověření zda obsahuje klíčové slovo free
            stahovatelny_odkaz = odkaz_na_strance                  # pokud ano -> uložení odkazu, ukončení hledání
            break
stahovatelny_odkaz = 'https://*********.cz' + stahovatelny_odkaz   # sestavení finálního odkazu ke stažení
try:
    soubor = requests.get(stahovatelny_odkaz)                      # zahájení stahování souboru
    with open('CESTA_K_ULOZENI' + nazev_souboru, 'wb') as f:       # uložení souboru do společného adresáře
        f.write(soubor.content)
except:
    print('Soubor neni mozne stahnout: ' + stahovatelny_odkaz)
```

Obr. 11 Zdrojový kód web-crawleru pro získání odkazu a stažení souboru [zdroj vlastní]

Po zpracování a uložení souboru program pokračuje a zpracovává další odkaz v předchystaném seznamu.

7.4.1.1 Stránka využívající JavaScript

V případě, že úložiště využívá pro tvorbu svých webových stránek JavaScript, stejně jako v případě tvoření seznamu souborů, je nutné využít virtuální prohlížeč. Samotné stahovatelné odkazy není možné získat pomocí modifikace URL adresy a vyhledávání klíčových slov ve vzniklém HTML kódu, ale je nutné vyhledat na stránce příslušný prvek (nejčastěji ikona, nebo tlačítko), který spouští stahování. Po identifikaci takového prvku, dochází k programovému kliknutí = dochází k simulaci činnosti reálného uživatele a spuštění stahování:

```
hledany_prvek=driver.find_element_by_link_text('Stáhnout soubor') # vyhledání tlačítka pomocí hledaného textu
hledany_prvek.click()                                             # stisk nalezeného tlačítka = start stahování
```

Obr. 12 Zdrojový kód web-crawleru pracujícím s JavaScriptem [zdroj vlastní]

7.5 Automatizované setřiďování stažených souborů

Soubory stažené z jednotlivých úložišť jsou shromažďovány v jednom společném adresáři. Z tohoto adresáře jsou soubory následně setřiďovány do adresářového systému pro dlouhodobé uložení.

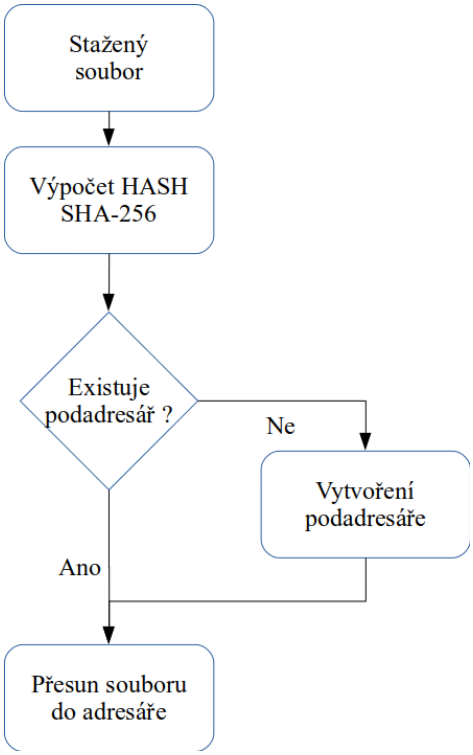
V jazyce Python byla vytvořena setřiďovací funkce, která řídí rozřazování balíčků. Balíčky jsou seskupovány do podmnožin, které mají stejný digitální otisk SHA-256, tedy mají identický datový obsah a liší se pouze názvem, pod kterým byly nalezeny na úložištích.

Schéma algoritmu setřiďování je uveden na Obr. 13.

Pro každý stažený soubor ze společného adresáře je vypočtena hodnota HASH.

V rámci kořenového adresáře pojmenovaném APK, jsou postupně vytvářeny podadresáře, jejichž název odpovídá hodnotě HASH SHA-256 aplikací, které patří do této podmnožiny.

Při zpracování každého soubory je rozhodnuto, zda kořenový adresář obsahuje podadresář pro hodnotu HASH zpracovávaného souboru. V případě, že ano, dochází k přesunu souboru do podadresáře. V případě, že adresář neexistuje, je nejprve vytvořen.



Obr. 13 Zjednodušené schéma algoritmu pro třízení souborů [zdroj vlastní]

Zdrojový kód metody v jazyku Python pro realizaci třízení souborů:

```
sha256=HASH.SHA256(APK_soubor)              # Výpočet HASH pro zvolený APK balíček

if os.path.exists(PWD+sha256+'/'):          # Ověření existence podadresáře
    if nacteny_soubor in os.listdir(PWD+sha256+'/APK/'): # kontrola proti duplicitnímu vložení
        os.remove(DWL+nacteny_soubor)
    else:
        os.rename(DWL+nacteny_soubor,PWD+sha256+'/APK/'+nacteny_soubor) #přesunutí APK souboru
else: # podadresář neexistuje a je potřeba jej vytvořit
    os.mkdir(PWD + sha256)           # vytvoření podadresáře
    os.mkdir(PWD+sha256+'/APK/')     # tvorba vnitřní struktury podadresáře
    os.mkdir(PWD + sha256 + '/T/')   # tvorba vnitřní struktury podadresáře
    os.rename(DWL + nacteny soubor, PWD + sha256 + '/APK/' + nacteny soubor) #přesunutí APK souboru
```

Obr. 14 Ukázka zdrojového kódu funkce pro setřiďování stažených souborů do podmnožin [zdroj vlastní]

Ze zdrojového kódu na Obr. 14 je patrné, že podadresář obsahuje další vrstvy adresářového systému pro ukládání souborů. Výsledná struktura je znázorněna na Obr. 15.

Obr. 15 Detail struktury podadresáře pro konkrétní hodnotu HASH [zdroj vlastní]

V rámci podadresáře jsou vytvořeny 4 složky, díky kterým jsou zjednodušeny další kroky činnosti a práce s výsledky:

- **APK**

  Do této složky jsou přesouvány stažené instalační balíčky rozřazené na základě stejné HASH.

- **APK_H**

  Jakmile dojde k programovému zápisu názvů souborů nacházejících se ve složce APK do databáze (viz následující kapitoly) jsou soubory, jejichž název byl uložen, přesunuty do tohoto adresáře pro dlouhodobé uložení.

- **T**

  Ve složce T jsou uloženy příchozí výsledky ze služby VirusTotal (viz následující kapitoly), které čekají na programový zápis do databáze.

- **H**

  Po zápisu výsledků z adresáře T do databáze, jsou soubory s výsledky přesunuty do tohoto adresáře pro dlouhodobé uložení.

Hlavním přínosem výše zmiňované adresářové struktury je zrychlení procesu analýzy, kdy dochází k seskupování obsahově stejných aplikací. Díky tomu lze odesílat pro analýzu vždy jen jednoho zástupce dané podmnožiny, zároveň je možné mapovat míru distribuce jednotlivých podmnožin a sledovat variaci názvů pod kterými byl datový obsah přístupný na úložištích (viz. Výsledky).

Obr. 16 Podadresáře pro jednotlivé hodnoty HASH
v rámci kořenového adresáře APK [zdroj vlastní]

7.6 Práce s VirusTotal API

Ověřování nebezpečnosti získaných souborů probíhá prostřednictvím API společnosti VirusTotal. Díky API je možné v jazyce Python vytvořit funkci, která prochází výše zmíněný kořenový adresář a pro jednotlivé hodnoty HASH vybírá zástupce ze skupiny aplikací, kterého nahrává na server VirusTotal k bezpečnostnímu prověření. Na serveru jsou sjednoceny desítky antivirových testů a nahraný soubor je otestován každým z nich. Po dokončení testování je získán soubor s výsledky ve formátu JSON.

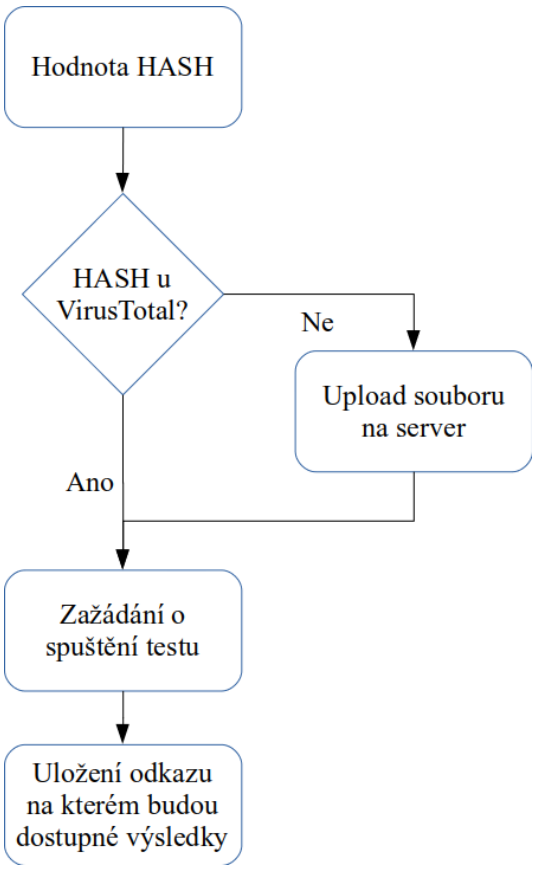
Komunikace s volně přístupným API je omezená počtem 4 dotazů za minutu. Při programové realizaci je nutné integrovat aplikační logiku pro synchronizaci jednotlivých požadavků na API. Zároveň je také implementována logika na programové zpracování případných chybových kódů, které API vrací.

7.6.1 Nahrání souborů a spuštění testování

Samotný proces lze zjednodušit do následujícího schématu (Obr. 17):

1. Program prochází jednotlivé podadresáře pro hodnotu HASH a načítá tuto hodnotu.
2. Následně dochází k ověření, zda v databázích společnosti VirusTotal existuje záznam o této HASH a soubor tedy v minulosti již byl uploadován. V případě přítomnosti není nutné opakovaně nahrávat duplicitní soubory, a tedy ušetřit čas analýzy.

   V případě, že soubor není na serveru VirusTotal nalezen, je nutné jej uploadovat na server (viz. Obr. 18 zdrojový kód funkce).
3. Jakmile je zajištěna přítomnost souboru na serveru, vyšle se požadavek na jeho otestování.



Obr. 17 Zjednodušené schéma programu pro spouštění testů prostřednictvím VirusTotal API [zdroj vlastní]

4. Uloží se soubor, který obsahuje údaje o požadavku na testování – identifikační číslo testu, pod kterým budou po dokončení testování přístupné výsledky.

Zdrojový kód jádra funkce jazyku Python řídící komunikaci s API:

```
j = requestAPI.rescan_request(HASH)                                # Odeslání hodnoty HASH na server
if j['response_code'] == 1:                                        # Případ, kdy se soubor na serveru již nachází
    j = json.dumps(j)                                              # Dochází ke zformátování JSON souboru
    if j:
        f = open(PWD + HASH + '/' + HASH + '_scan.json', 'w+')
        f.write(j)                                                 # Uložení informací o testu do souboru (viz. bod 4)
        f.close()
else:                                                              # Soubor je nutné na server uploadovat
    scan = requestAPI.scan_request(PWD + HASH + '/APK/' + file)    # Jako parametr předána cesta k souboru pro testování
                                                                   # Dochází k uploadu souboru
    j = json.dumps(j)                                              # Zformátování JSON souboru
    if j:
        f = open(PWD + HASH + '/' + HASH + '_scan.json', 'w+')
        f.write(j)                                                 # Uložení informací o testu do souboru (viz. bod 4)
        f.close()
```

Obr. 18 Zdrojový kód funkce pro komunikaci s API a testování souborů [zdroj vlastní]

Vnořená funkce rescan_request pro komunikaci se serverem:

```
def rescan_request(resource):                                      # Vnořená fce pro komunikaci se serverem
    params = {'apikey': '****************************************************************',
              'resource': resource}                                # Identifikační klíč API, předání souboru/HASH pro test
    headers = {
        "Accept-Encoding": "gzip, deflate",
        "User-Agent": "gzip,  My Python requests library example client or username"
    }
    # Odeslání požadavku na testování:
    response = requests.post('https://www.virustotal.com/vtapi/v2/file/rescan', params=params, headers=headers)
```

Obr. 19 Zdrojový kód vnořené komunikační funkce pro spouštění testování souborů [zdroj vlastní]

Funkce rescan_request odesílá požadavek na opětovné ověření již nahraného souboru, který je identifikován hodnotou HASH (identifikační klíč pro přístup k API byl anonymizován).

V případě, že soubor s konkrétní HASH není na serveru k dispozici, je potřeba uploadovat zástupce z podmnožiny pro HASH. Za tímto účelem je využita funkce **scan_request**:

```
def scan_request(file):                                            # Vnořená fce pro komunikaci se serverem
    params = {'apikey': '****************************************************************'}
    files = {'file': (file, open(file, 'rb'))}                     # Identifikační klíč API, předání souboru pro upload
    headers = {
        "Accept-Encoding": "gzip, deflate",
        "User-Agent": "gzip,  My Python requests library example client or username"
    }
    # Odeslání požadavku pro upload a testování testování:
    response = requests.post('https://www.virustotal.com/vtapi/v2/file/scan', files=files, params=params)
```

Obr. 20 Zdrojový kód vnořené komunikační funkce pro upload a spouštění testování souborů [zdroj vlastní]

Funkce realizuje upload i následné otestování souboru a stejně jako předchozí vrátí identifikační číslo testu, pod kterým bude možné dohledat výsledky.

7.6.2 Vyzvednutí výsledků testů

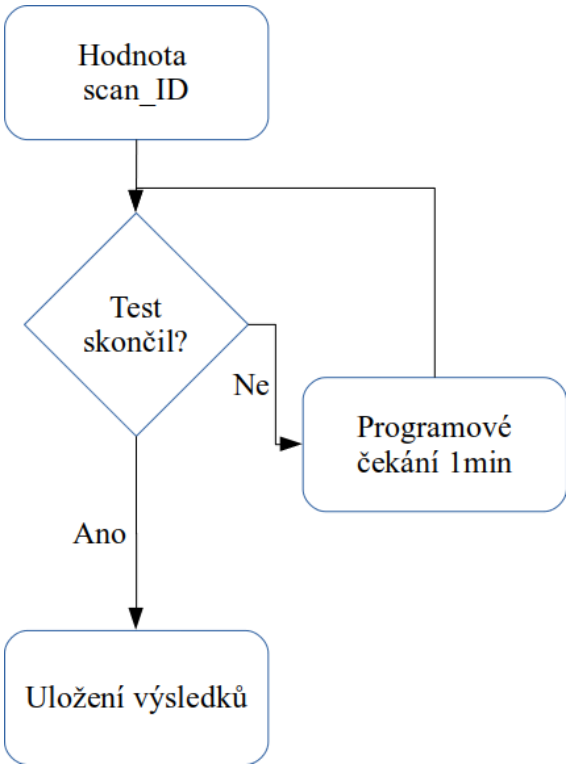
Po spuštění testování pro veškeré hodnoty HASH v adresářovém systému, přichází druhá fáze komunikace – vyzvedávání výsledků bezpečnostního testování.

Samotné vyzvednutí probíhá v několika fázích (viz. Obr. 21)

1. Ze souboru s výsledky vzniklého předchozí fází dochází k načtení hodnoty scan_ID, která identifikuje provedený test a odkazuje na výsledky.

2. V případě, že VirusTotal API vrátí v reakci na scan_ID příznak o dokončení testování. Dochází k uložení výsledků. V opačném případě, dochází k prodloužení čekací doby o 1 minutu a následně dalšímu dotazu na dostupnost výsledků.

3. Výsledky jsou opět vraceny ve formátu JSON, kdy celý tento soubor je ukládán pro další zpracování.



Obr. 21 Zjednodušené schéma pro vyzvednutí výsledků testů prostřednictvím VirusTotal API [zdroj vlastní]

Zdrojový kód funkce v jazyce Python řídící přijímaní výsledků:

```
with open(PWD + HASH + '/' + file) as f:                    # otevření souboru obsahujícího scan_ID
    json_ = json.load(f)
    f.close()
try:
    j = responseAPI.response_request(json_['scan_id'])      # načtení hodnoty scan_ID a následné předání pro API
    j = json.dumps(j)                                       # formátování souboru JSON
    if j:
        os.remove(PWD + HASH + '/' + file)
        f = open(PWD + HASH + '/' + HASH + '.json', 'w+')   # uložení souboru s výsledky bezpečnostního testu
        f.write(j)
        f.close()
```

Obr. 22 Zdrojový kód funkce pro vyzvedávání výsledků ze serveru VirusTotal prostřednictvím API [zdroj vlastní]

Vnořená funkce responce_request realizující přímou komunikaci s API:

```
def response_request(scan_id):                    # Vnořená fce pro komunikaci se serverem
    params = {'apikey': '************************************************************',
              'resource': scan_id}                # Identifikační klíč API, předání scan_ID pro vyzvednutí výsledků
    headers = {
        "Accept-Encoding": "gzip, deflate",
        "User-Agent": "gzip,  My Python requests library example client or username"
    }
    # Odeslání požadavku pro vyzvednutí výsledků:
    response = requests.get('https://www.virustotal.com/vtapi/v2/file/report', params=params, headers=headers)
```

Obr. 23 Zdrojový kód vnořené funkce využité k vyzvednutí výsledků ze serveru [zdroj vlastní]

Výsledný soubor s výsledky obsahuje detailní přehled o provedeném testu. Nachází se zde údaje o datumu testu, identifikátor souboru (HASH), přehled provedených testů, názvy detekovaných hrozeb a v neposlední řadě již dříve zmiňované virus-ratio. Tedy poměr mezi celkovým počtem provedených testů a počtem pozitivních identifikací hrozby.

Soubor je ukládán v rámci adresářové struktury pro další zpracování a zápis do databáze.

7.7 Tvorba databáze a zpracování výsledků

7.7.1 Rozdělení získaných aplikací podle nebezpečnosti:

Předchozími kroky analýzy byly získány mobilní aplikace z neoficiálních zdrojů. Následnou analýzou pomocí API společnosti VirusTotal jsou tyto aplikace podle hodnoty virus-ratio rozděleny do čtyř bezpečnostních skupin:

- **Clean:** množina aplikací, ve kterých nebyla nalezena žádná hrozba
- **Low risk:** množina aplikací s 1 – 5ti testy vyhodnocenými jako pozitivní nález
- **Medium risk:** množina aplikací s 6 – 10ti testy vyhodnocenými jako pozitivní nález
- **High risk:** množina aplikací s více než 10ti testy vyhodnocenými jako pozitivní nález

7.7.2 Automatická tvorba databáze

Pro přehledné uložení výsledků a jednoduchou práci s nimi, byla vytvořena databázová struktura v jazyce MySQL.

Hlavními důvody pro využití technologie MySQL je přehledné ukládání dat a zároveň velmi dobrá možnost využívat databázi v programu jazyku Python nebo webovém skriptu v PHP.

Samotný zápis dat a aktualizace databáze je realizována pomocí programu vytvořeném v jazyce Python. Dochází k postupnému procházení jednotlivých podadresářů pro HASH a vyhledávání souboru JSON s výsledky bezpečnostního testování. Následně jsou ze souboru JSON načítána příslušná data a vytvořen SQL dotaz, který je předán databázi:

```
p = "{:03d}".format(json_['positives'])            #vyhledání počtu pozitivně vyhodnocených testů z výsledků VirusTotal
t = "{:03d}".format(json_['total'])                #vyhledání počtu celkově provedených testu z  výsledků VirusTotal
VR=str(p+'/'+t)                                    #spojení předchozích dvou čísel do jednoho poměrového udaje
date = json_['scan_date']


if ID:                                             #položka se již nachází v databázi
    mySQL.HASH_update(ID[0],VR,date,mySQL.severity(json_['positives'])) #aktualizace hodnoty v tabulce databáze
else:                                              #položka v databázi zatím neexistuje
    mySQL.HASH_insert(HASH,VR,date,mySQL.severity(json_['positives']))  #vytvoření položky v rámci tabulky databáze
    ID = mySQL.DB_GET_ID(HASH)
```

Obr. 24 Ukázka zdrojového kódu funkce pro komunikaci s databází. [zdroj vlastní]

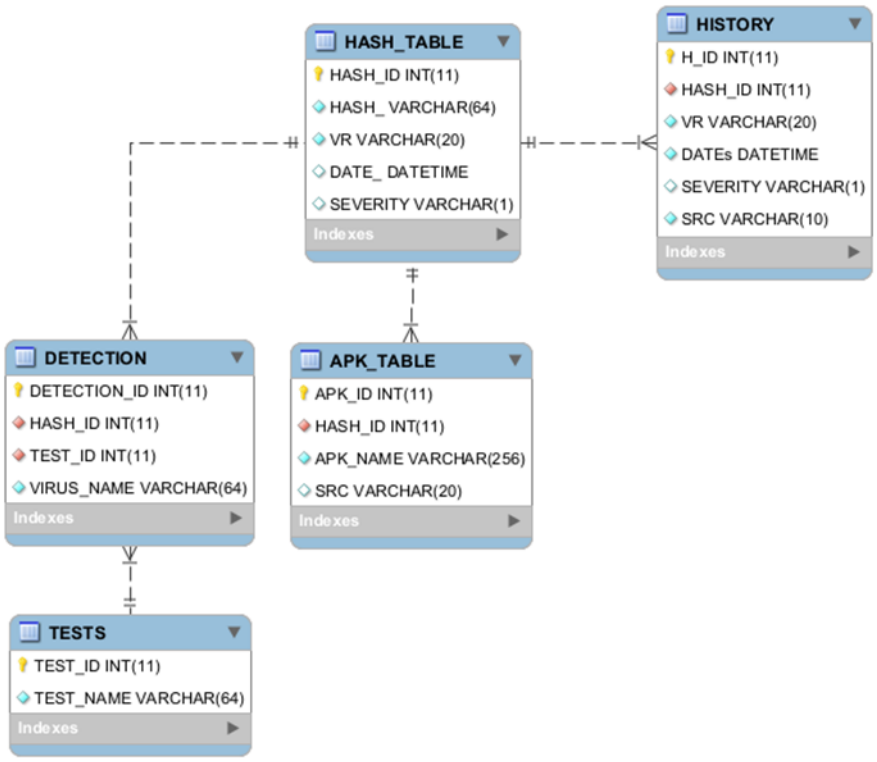
V rámci výše uvedeného zdrojového kódu opět dochází k využití vytvořených vnořených funkcí. Tyto funkce mají podobnou strukturu, kdy přebírají data z parametrů a vkládají je do předchystaného SQL dotazu. Vzniklý dotaz je následně odeslán do databáze:

```
def HASH_insert(HASH,VR,DATE_,SEVERITY):
    try:
        connection = mysql.connector.connect(host='**********',        # připojení k databázi
                                             database='APK_DATABASE',
                                             user='**********',
                                             password='**********')

        sql_insert_query = """ INSERT INTO HASH_TABLE (HASH_,VR,DATE_,SEVERITY)
                           VALUES (%s,%s,%s,%s) """ # předchystaný SQL dotaz, který čeká na vložení parametrů

        cursor = connection.cursor(prepared=True)
        Tuple = (HASH,VR,DATE_,SEVERITY)           # předání přijatých parametrů do struktury Tuple
        result = cursor.execute(sql_insert_query, Tuple) # spojení SQL dotazu a hodnot jednotlivých parametrů
        connection.commit()                        # odeslání dotazu do databáze.
```

Obr. 25 Tvorba SQL dotazu v rámci zdrojového kódu jazyka Python [zdroj vlastní]

Prostřednictvím popsaného Python programu dochází ke zpracování veškerých souborů s výsledky a správě databáze.

Struktura databáze je následovná:



Obr. 26 Struktura mySQL databáze pro ukládání výsledků testování [zdroj vlastní]

Databáze je rozdělena do pěti tabulek:

- **HASH_TABLE:**

  Hlavním prvkem celé databáze je tato tabulka. Je zde tvořen záznam pro každou unikátní HASH, která byla získána předchozími kroky. Pro každou hodnotu je přiřazen jednoznačný číselný identifikátor, který vstupuje do dalších tabulek pro vytvoření relací. Zároveň jsou ukládány údaje o výsledku posledního testu, konkrétně hodnota virus-ratio, datum provedení tohoto testu a na základě virus-ratia je této HASH přiřazena kategorie nebezpečnosti (viz. Rozdělení aplikací podle nebezpečnosti).

- **APK_TABLE:**

  V tabulce APK_TABLE je vytvořena relace mezi jednoznačným identifikátorem HASH a jednotlivými názvy, pod kterými byly soubory odpovídající této HASH nalezeny na webových uložištích. Pro jednu unikátní hodnotu HASH, může v této tabulce být více záznamů, v případě že datový obsah odpovídající hodnotě HASH byl distribuován pod více názvy.

V rámci neveřejných výsledků je také ukládán údaj o původu konkrétního názvu. Tedy ze kterého úložiště byla stažena aplikace s tímto názvem, odpovídající příslušné hodnotě HASH.

- **DETECTION:**
  Představuje vazební tabulku pro uložení dodatečných informací o testování. V rámci testování dochází k ověření bezpečnosti několika desítkami enginů anti-virových společností. Názvy těchto enginů jsou uloženy v tabulce TESTS.
  Pro každý test, který vyhodnotil konkrétní HASH jako hrozbu, vzniká v tabulce DETECTION nový záznam. Dochází k propojení identifikátoru testované HASH a identifikátoru enginu, který prováděl test. V rámci každého propojení se ukládá také název hrozby, kterou příslušný engine v rámci souboru identifikoval.
  Takto je možné uložit detailní údaje o provedených testech, kdy víme, jaké hrozby jsou v rámci stažených aplikací detekovány, popřípadě jakými bezpečnostními enginy.
- **TESTS:**
  Tabulka obsahující názvy jednotlivých enginů anti-virových společností, které se podílely na testování.
- **HISTORY**
  V tabulce HISTORY dochází k zálohování výsledků z tabulky APK_TABLE, v případě, že dojde k novému testování a výsledky v tabulce APK_TABLE jsou aktualizovány.

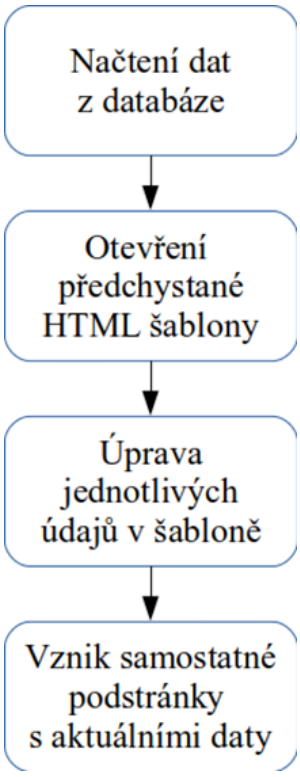
Relace mezi jednotlivými tabulkami umožňují jednoduchou práci s výsledky. Je možné sledovat počty distribucí jednotlivých HASH, variabilitu názvů použitých k distribuci. Monitorovat hrozby identifikované v aplikacích, vyhledávat které soubory jsou nejvíce nebezpečné, nejčastější hrozbu atd.

8 TVORBA WEBOVÉ VIZUALIZACE

Dalším z cílů práce je předávat informace a výsledky zpět veřejnosti. Pro tento účel byla vytvořená webová vizualizace, umožňující snadný přístup k výsledkům.

Vizualizace je vytvořená pomocí technologie HTML5, JavaScript, CSS a PHP. Tvorba a aktualizace webových stránek je řízena programem v jazyce Python.

Tento program načítá data z databáze a do předpřipravených HTML šablon pro jednotlivé stránky, vkládá načtená data. Dochází tak k úpravě hodnot na stránce a vznikají jednotlivé podstránky (viz. Obr 27).



Obr. 27 Schéma procesu tvorby webové vizualizace [zdroj vlastní]

Díky využití skriptů pro generování webové stránky, lze snadno aktualizovat dostupný obsah, udržovat informace aktuální a umožnit případný proces tvorby vizualizace na jiné zařízení.

Samotná vizualizace je vytvářena s důrazem na přehlednost pro uživatele, umožnit i uživatelům, kteří nejsou odborníky na danou problematiku, porozumět uvedeným datům.

Zároveň byl kladen důraz, aby web nabízel dostatečné funkce i pro uživatele z řad odborné komunity a představoval komplexní zdroj informací o napadenosti jednotlivých souborů, aktuálních hrozbách apod.

Potenciální publikum je tedy velmi široké.

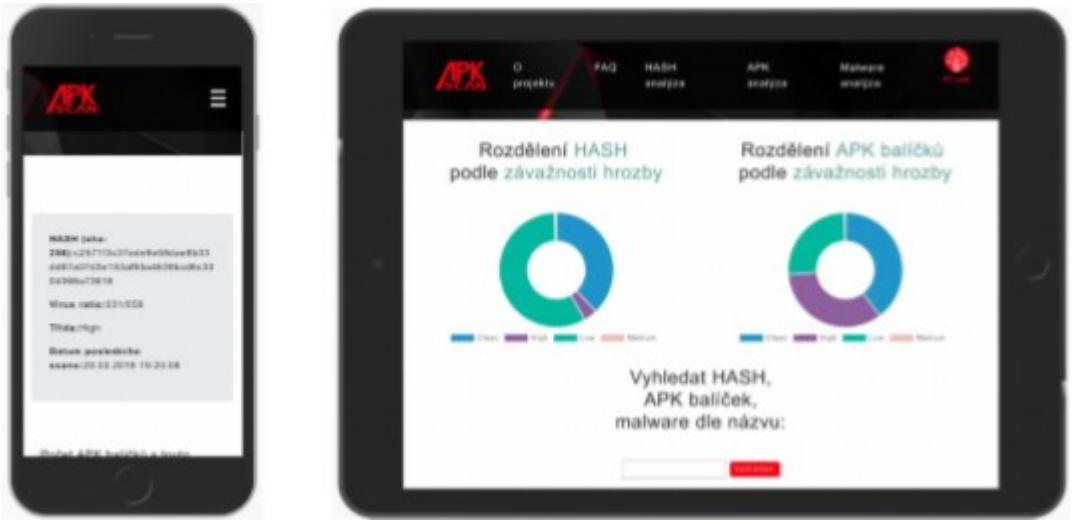
Ukázka zdrojového kódu pro vytváření jedné z podstránek vizualizace:

```
hash_count=mySQL.count_hash()                   #zjištění počtu HASH uložených v databázi
hash_count = int(hash_count[0])                 #zjištění počtu HASH uložených v databázi
for i in range(1,hash_count+1):                 #postupné procházení databáze

    with open('/var/www/html/HASH/template.html', 'r') as file:     #načtení šablony HTML pro úpravu
        data = file.readlines()


    hash = mySQL.DB_GET_HASH(i)                 #načtení hodnoty HASH z databáze
    apk = mySQL.DB_GET_APK(i)                   #načtení pole názvů pod kterými byla HASH nalezena
    virus = mySQL.DB_GET_VIRUS(i)               #načtení výsledků testů - názvů hrozeb identifikovaných pro HASH

    #uprava šablony HTML na příslušných místech --> dochází k vložení dat do HTML kódu:
    data[8]=hash[0]+'\n'
    data[115]='<span style="font-weight: 700">HASH (sha-256):</span>'+hash[0]+'\n'
    data[118]='<span style="font-weight: 700">Virus ratio:</span>' + hash[1]+'\n'
    data[121]='<span style="font-weight: 700">Severity:</span>' + mySQL.rseverity(hash[3])+'\n'
    data[124]='<span style="font-weight: 700">Last scan:</span>' + hash[2].strftime('%d.%m.%Y %H:%M:%S')+'\n'
    data[135]='Number of packages with this HASH:'+str(len(apk))+'\n'
    data[147]='Number of threats decected in packages with this HASH: '+str(len(virus))+'\n'

    insert = []
    for x in apk:                               #postupné vytvoření seznamů odkazů na jednotlivé názvy aplikací
        insert.append('<li>'+x[0]+'</li>\n')
    insert = ''.join(insert)

    data[141]=insert
    with open('/var/www/html/HASH/' + hash[0] + '.html', 'w') as file: #uložení podstránky pro konkrétní HASH
        file.writelines(data)
```

Obr. 28 Zdrojový kód programu pro úpravu HTML šablony a následnou tvorbu podstránky s výsledky [zdroj vlastní]

8.1 Technologie Bootstrap

Díky využití technologie Bootstrap při tvorbě webové vizualizace, je zaručena kompatibilita webových stránek s různými typy obrazovek – od displejů mobilních zařízení, tabletů až po širokoúhlé monitory. Framework Bootstrap zaručí správné vykreslování obrazu a čitelnost stránky:



Obr. 29 Ukázka kompatibility webové vizualizace s různými typy obrazovek [zdroj vlastní]

Detailní popis webové vizualizace možnosti práce s ní se nachází v kapitole 10 popisující výsledky práce.

9 ANALÝZA VYBRANÝCH VZORKŮ MOBILNÍHO MALWARU

V rámci předchozích kapitol byla popsána metodika a vznik automatizovaného systému umožňujícího získávat mobilní aplikace z nelegálních úložišť souborů, odlišovat škodlivé soubory a automaticky zpracovávat výsledky.

V následující kapitole jsou popsány metody statické a dynamické analýzy použité při ruční analýze vybraných vzorků mobilního malwaru.

Pro následnou analýzu, je potenciálně nejzajímavější skupina aplikací, které jsou označeny jako ty nejvíce škodlivé, tedy náležící do množiny High risk (viz. oddíl 7.7.1 Rozdělení aplikací podle nebezpečnosti).

V rámci teoretické části byly vysvětleny postupy používané při analýze mobilních aplikací. Práce využívá pro analýzu mobilních aplikací metody statické analýzy, kdy navržený postup se sestává ze tří postupných kroků:

1. Analýza souboru AndroidManifest.xml
2. Analýza zdrojových souborů
3. Analýza zdrojového kódu

9.1 Analýza souboru AndroidManifest.xml

Jak již bylo vysvětleno, soubor AndroidManifest.xml je konfigurační soubor aplikace, ve kterém musejí být uváděny informace o aplikaci, seznam využitých aktivit, broadcast receiverů apod.

Hlavně zde musí být uveden požadovaný seznam oprávnění vyžadovaných aplikací.

Jakákoliv funkcionalita aplikace, snažící se přistoupit k chráněné části operačního systému, musí zažádat o příslušné oprávnění. V případě, kdy oprávnění není vyžádáno a aplikace se pokusí přistoupit k chráněné části, dochází ke vzniku bezpečnostní výjimky a aplikace je ukončena. Proto je soubor manifestu hlavním styčným bodem naší analýzy. Pro škodlivou funkcionalitu platí stejná pravidla a v případě, že tvůrci mobilní malwaru infikují aplikaci škodlivým kódem. I pro tento škodlivý kód musí být vyžádána příslušná oprávnění. V rámci souboru manifestu tedy škodlivá funkcionalita nemůže být zatajena.

9.1.1 Dekompilace aplikace pomocí nástroje APKTOOL

V rámci teoretické části byl představen nástroj APKTOOL umožňující rozložení aplikace na jednotlivé soubory. Tento nástroj je využit pro extrakci souboru manifestu.

Mějme adresář s aplikací určenou k dekompilaci:

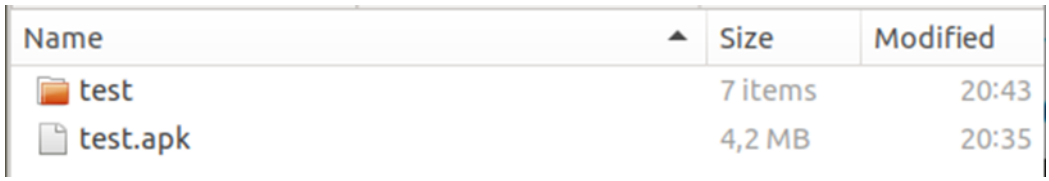
Obr. 30 Postup dekompilace mobilních aplikací – krok 1: aplikace určená k dekompilaci [zdroj vlastní]

Pomocí nástroje APKTOOL spustíme v terminálu dekompilaci aplikace:

```
jan_kincl@jan-kincl-OMEN-by-HP-Laptop:~/test$ apktool d test.apk
I: Using Apktool 2.3.4 on test.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
S: WARNING: Could not write to (/home/jan_kincl/.local/share/apktool/framework),
 using /tmp instead...
S: Please be aware this is a volatile directory and frameworks could go missing,
 please utilize --frame-path if the default storage directory is unavailable
I: Loading resource table from file: /tmp/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
jan_kincl@jan-kincl-OMEN-by-HP-Laptop:~/test$
```

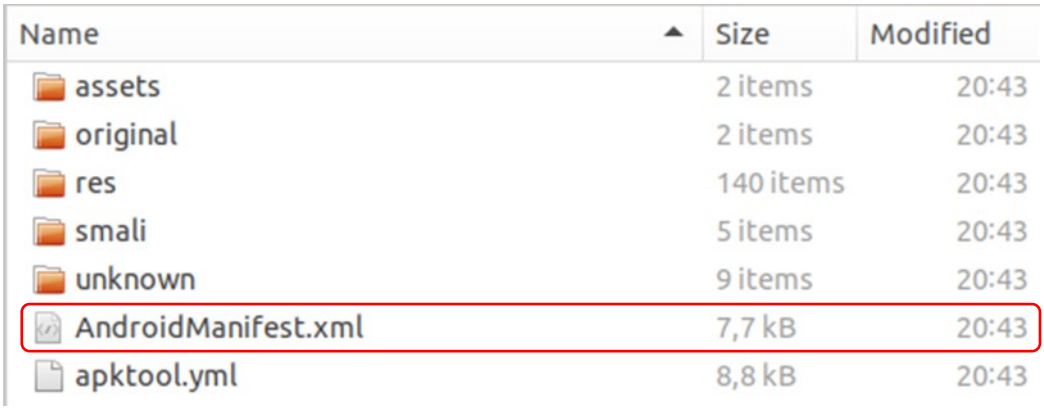
Obr. 31 Postup dekompilace mobilních aplikací – krok 2: spuštění dekompilace aplikace [zdroj vlastní]

Dekompilované soubory jsou uloženy do adresáře se stejným názvem, jaký měla aplikace pro dekompilaci:



Obr. 32 Postup dekompilace mobilních aplikací – krok 3: vznik adresáře s dekompilovanými daty [zdroj vlastní]

V tomto adresáři se pak nacházejí jednotlivé soubory tvořící instalační .apk soubor mobilní aplikace:



Obr. 33 Postup dekompilace mobilních aplikací – krok 4: ukázka vnitřní struktury adresáře vzniklého dekompilací [zdroj vlastní]

Soubor manifestu je přístupný k další analýze. Kdy takový soubor může vypadat nějak takto:

Obr. 34 Postup dekompilace mobilních aplikací – krok 5: ukázka extrahovaného souboru manifestu 1 [zdroj vlastní]

Ukázka seznamu aktivit aplikace:

```
        <activity android:configChanges="keyboardHidden|orientation|screenSize"
android:label="AppsgeyserBrowser" android:launchMode="singleTask"
android:name="com.wDrasticEmulatorFull_4697919.VideoPlayerActivity"
android:screenOrientation="sensor" android:theme="@style/AppTheme"/>
        <activity android:configChanges="orientation" android:label="Ads"
android:name=".ads.vast.VideoAdsLoaderActivity"/>
        <activity android:name="org.nexage.sourcekit.vast.activity.VASTActivity"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"/>
        <activity android:excludeFromRecents="true" android:label="App message"
android:name=".MessageViewer"/>
        <activity android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|
screenSize|smallestScreenSize|uiMode" android:name="com.google.android.gms.ads.AdActivity"/>
        <activity android:label="Loading..." android:name=".BrowserActivity"/>
        <meta-data android:name="com.google.android.gms.version" android:value="@integer/
google_play_services_version"/>
```

Obr. 35 Postup dekompilace mobilních aplikací – krok 5: ukázka extrahovaného souboru manifestu 2 [zdroj vlastní]

Důležitý seznam požadovaných oprávnění pro aplikaci:

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="com.wDrasticEmulatorFull_4697919.permission.C2D_MESSAGE"/>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Obr. 36 Postup dekompilace mobilních aplikací – krok 5: ukázka extrahovaného souboru manifestu 3 [zdroj vlastní]

Při analýze souboru manifestu dochází k postupnému dekompilování všech vzorků určených pro analýzu. V rámci získaných souborů se vyhledávájí znaky vykazující podobnost napříč množinou, tj. takové, které by byly charakteristické pro mobilní malware.

9.1.1.1 Kompilace aplikací pomocí nástroje APKTOOL

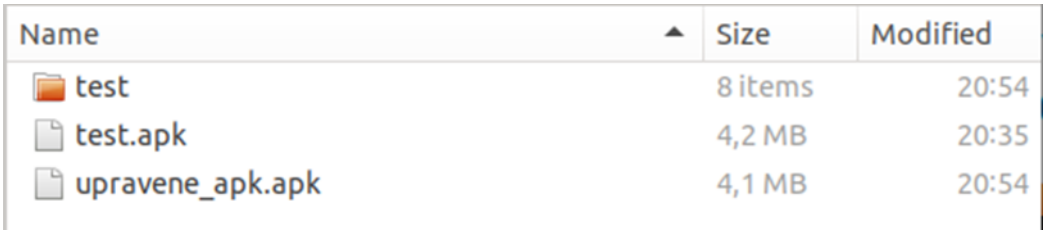
Nástroj APKTOOL ovšem neslouží pouze k dekompilaci jednotlivých aplikací, ale umožňuje také zpětnou kompilaci aplikace a vytvoření nového .apk souboru. V předchozích kapitolách bylo popsáno, že útočníci a tvůrci mobilního malware často využívali tzv. metodu repackage. Ta spočívá ve stažení legitimní aplikace a její dekompilaci, následné úpravě zdrojových kódu a přidání škodlivé funkcionality. Škodlivá aplikace pak často byla distribuována prostřednictvím neoficiálních úložišť mezi uživatele.

Po úpravě jednotlivých souborů v dekompilovaném adresáři, je pomocí nástroje APKTOOL spuštěna kompilace:

```
jan_kincl@jan-kincl-OMEN-by-HP-Laptop:~/test$ apktool b test/ -o upravene_apk.apk
I: Using Apktool 2.3.4
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
S: WARNING: Could not write to (/home/jan_kincl/.local/share/apktool/framework),
using /tmp instead...
S: Please be aware this is a volatile directory and frameworks could go missing,
please utilize --frame-path if the default storage directory is unavailable
W: warning: string 'urlboxHint' has no default translation.
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
jan_kincl@jan-kincl-OMEN-by-HP-Laptop:~/test$
```

Obr. 37 Ukázka kompilace nového .apk souboru pomocí nástroje APKTOOL [zdroj vlastní]

Po kompilaci aplikace obsahuje původní adresář kromě originální aplikace i aplikaci upravenou:

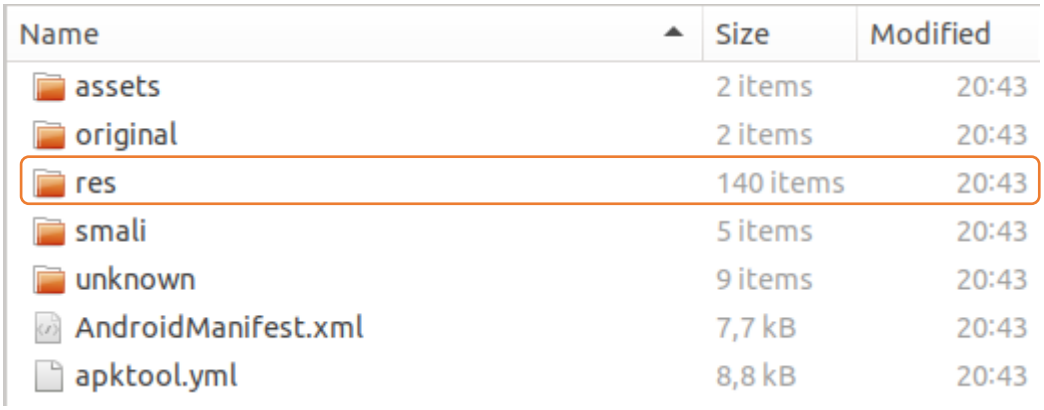


Obr. 38 Ukázka nově vniklé aplikace pomocí nástroje APKTOOL [zdroj vlastní]

9.2 Analýza zdrojových souborů aplikace

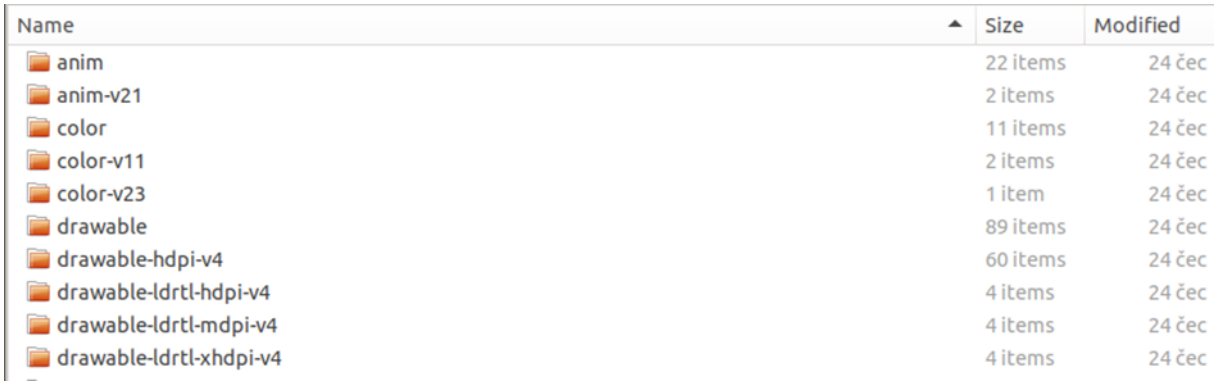
Po dokončení analýzy souborů AndroidManifest.xml se provádí druhý krok, při kterém jsou analyzovány zdrojové soubory aplikací. Tato práce pro identifikaci rysů využívá převážně zdrojové soubory ikon, na základě kterých opět vyhledává možné podobnosti mezi vzorky mobilního malwaru.

Postup začíná dekompilací aplikace, stejně jako v předchozím případě. Ve vzniklém adresáři se následně zkoumá obsah složky res:



Obr. 39 Adresář obsahující zdrojová data použitá v aplikaci [zdroj vlastní]

V rámci složky res, která shromažďuje veškeré zdrojové soubory použité v aplikaci jsou ikony nejčastěji uloženy v adresáři drawable:



Obr. 40 Vnitřní struktura adresáře resources [zdroj vlastní]

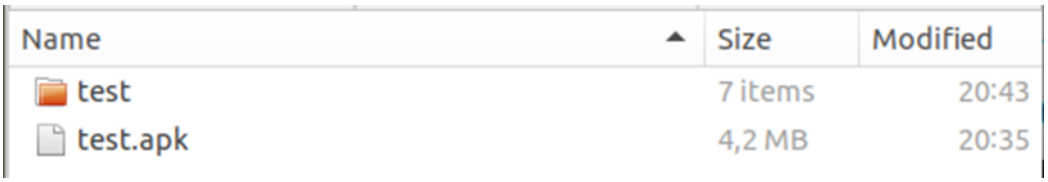
V rámci analyzovaných vzorků opět dochází k vyhledávání podobností v rámci použitých ikon, které by propojovaly vzorky mobilního malwaru.

9.3 Analýza zdrojového kódu aplikace

V rámci teoretické části byly představeny nástroje nejen pro dekompilaci aplikací, ale také pro převod zdrojového kódu využitého v aplikacích Android do jazyku Java pro možnost analýzy činnosti aplikace.

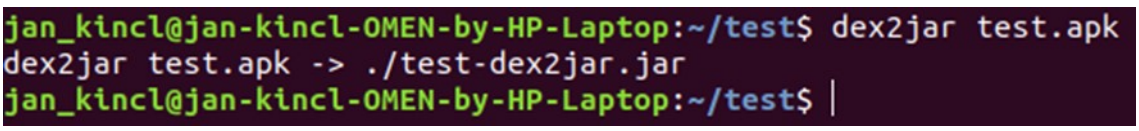
V prvním kroku je potřeba použít nástroj dex2jar, který převede .dex soubory aplikace na .class souboru jazyku Java, které je následně možné analyzovat.

Opět v adresáři s aplikací pro testování:



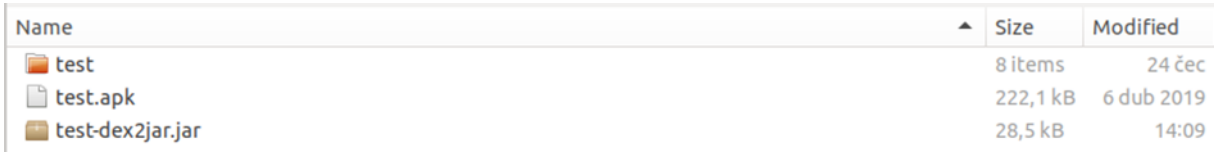
Obr. 41 Adresář s aplikací určenou pro analýzu zdrojového kódu [zdroj vlastní]

Spustíme terminál a použijeme nástroj dex2jar:



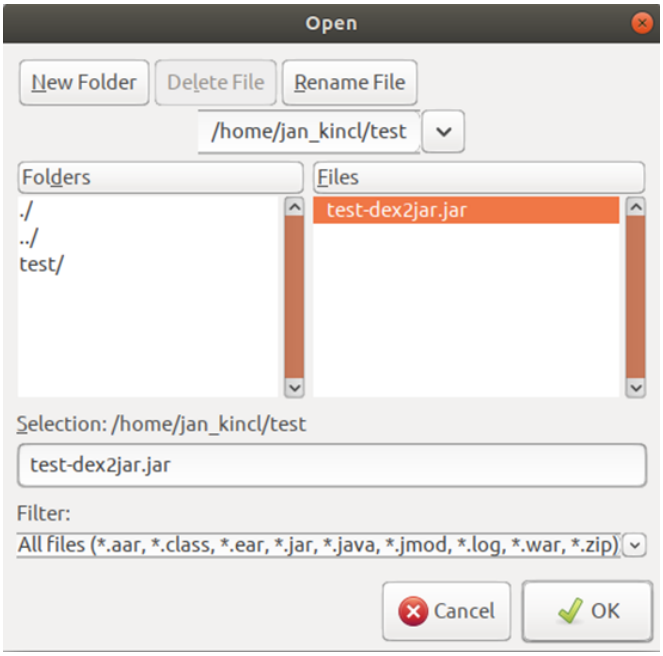
Obr. 42 Ukázka použití nástroje dex2jar při analýze zdrojových kódů [zdroj vlastní]

V původním adresáři vzniká .jar soubor, ve kterém jsou již zdrojové kódy převedeny do jazyku Java:



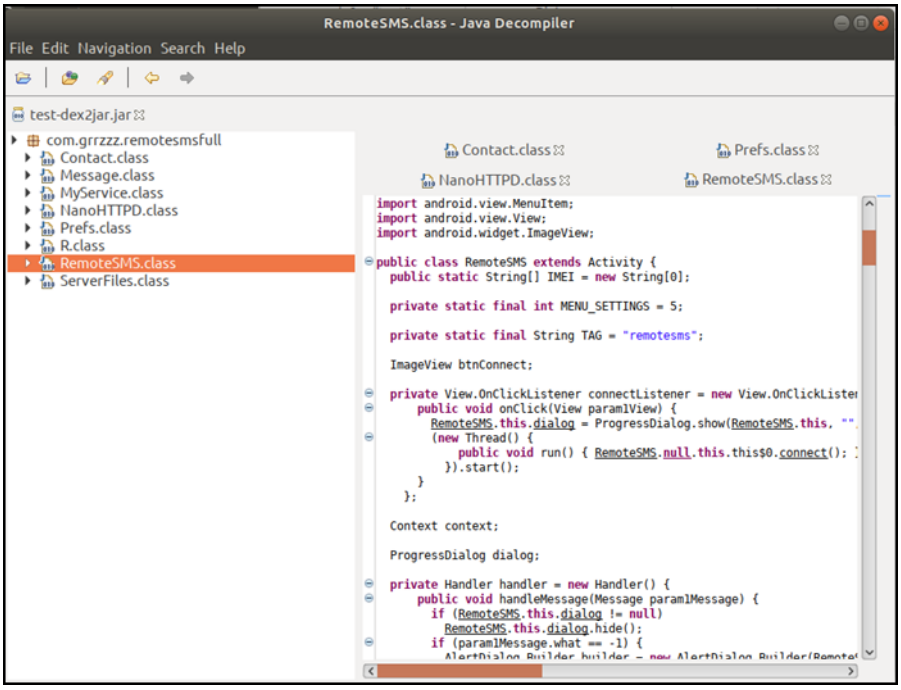
Obr. 43 Vzniklý .jar soubor pro analýzy zdrojových kódů [zdroj vlastní]

Pro zobrazení zdrojových kódu v jazyce Java je nutné využít tzv. Java Decompiler (viz. Teoretická část). Pomocí tohoto nástroje dojde k otevření jar archivu a zobrazení textové reprezentace zdrojového kódu aplikace v jazyce Java:



Obr. 44 Využití nástroje JD-GUI
pro otevření .jar archivu [zdroj vlastní]

Otevřením archivu získáme obrazovku podobnou některým vývojovým prostředím, kdy v levé části vidíme strukturu zdrojového kódu a v pravé části samotný zdrojový kód aplikace:



Obr. 45 Zobrazení dekompilovaného zdrojového kódu pomocí nástroje JD-GUI [zdroj vlastní]

V rámci jednotlivých vzorků malwaru, zvolených pro analýzu, dochází k identifikaci činnosti aplikace analýzou zdrojového kódu. Vyhledávají se znaky a metody opakovaně využívané v rámci zachycených vzorků, znaky podobnosti, které by mohly identifikovat mobilní malware. Zároveň dochází k vyhledávání podobné funkcionality mezi zachycenými vzorky, v případě, že by byly využity stejné útočné techniky.

10 VIZUALIZACE VÝSLEDKŮ

Pro předávání výsledků analýz, informací o nalezených souborech, metodách distribuce apod. zpět veřejnosti. Byla vytvořena webová vizualizace.

Současná bezpečnostní situace týkající se mobilních zařízení není dobrá, nejen z důvodů popsaných v rámci teoretické části, ale také z důvodu nedostatečné informovanosti uživatelů o problematice. Uživatelé si často nejsou vědomi možných rizik spojených s instalací aplikací z neoficiálních zdrojů a při samotné instalaci rovněž nevěnují pozornost průběhu instalace.

Proto je cílem webových stránek zvyšovat povědomí a poskytovat možnost uživatelům seznámit se s problematikou mobilního malwaru. Zároveň web implementuje nástroj, který je možné použít pro rychlou kontrolu bezpečnosti aplikací z uživatelovy strany.

Webová vizualizace je provozována pod záštitou Penetrační laboratoře PTLAB UTB, dostupná na adrese:

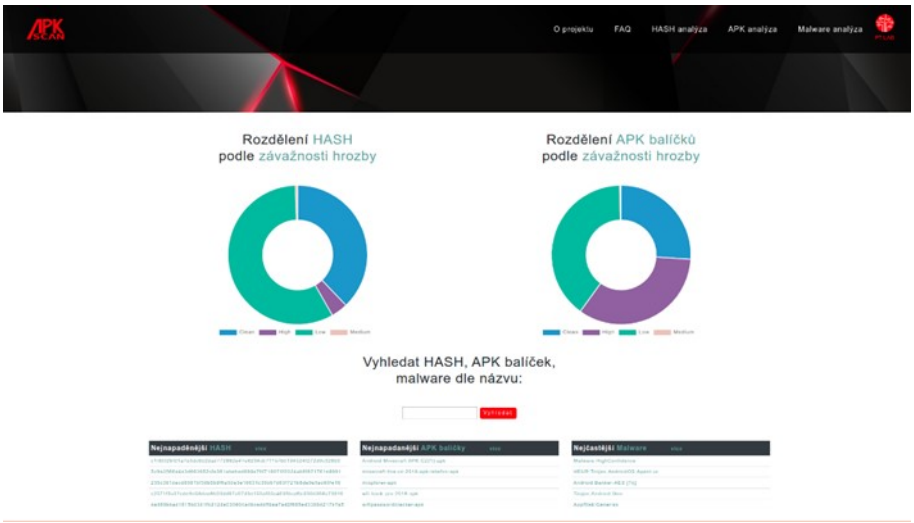
https://av.ptlab.fai.utb.cz/

V rámci popisu jednotlivých částí webové vizualizace budou rovněž popsány jednotlivé výsledky práce.

10.1 Titulní strana webové vizualizace

Titulní strana byla vytvořena s důrazem na přehlednost a vysokou informační hodnotu. Díky automatizovanosti procesu vytváření vizualizace (viz. kapitola 8) jsou na titulní straně uváděny vždy co nejnovější výsledky. Prostřednictvím jednotlivých grafů a tabulek má uživatel obdržet přehledné informace o aktuální situaci bez nutnosti dalšího procházení webu.

Titulní strana zároveň slouží jako rozcestník pro pohyb na stránce. Jednotlivé podstránky budou popsány v rámci dalších podkapitol stejně jako prvky umístěné na titulní straně.



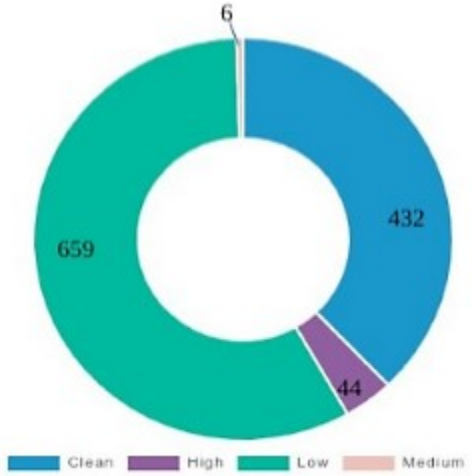
Obr. 46 Titulní strana webové vizualizace [zdroj vlastní]

10.1.1 Rozdělení aplikací do skupin podobné nebezpečnosti

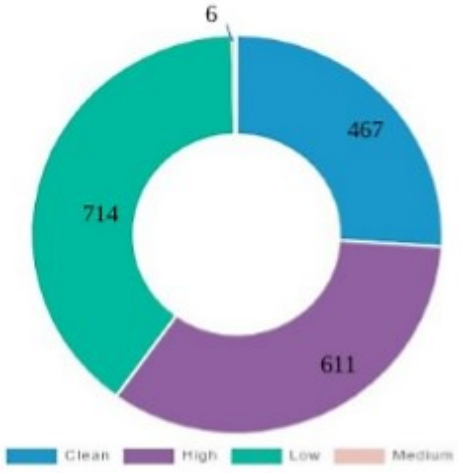
Automatizovaným systémem bylo pro účely analýzy získáno téměř 2000 vzorků mobilních aplikací z neoficiálních úložišť. Každá z aplikací byla vyhodnocena pomocí VirusTotal API (viz. oddíl 7.6) a na základě získané hodnoty virus-ratio rozděleny do 4 skupin nebezpečnosti (viz. oddíl 7.7.1).

Při analýze je na stažené aplikace pohlíženo dvěma způsoby. Pro každou staženou aplikaci je spočten její digitální otisk HASH SHA-256. Na základě této hodnoty jsou aplikace seskupovány do podmnožin se stejným datovým obsahem (viz. oddíl 7.5). Zároveň každá ze stažených aplikací figuruje pod nějakým názvem jako soubor na úložišti. Jeden unikátní datový obsah tedy může být distribuován pomocí libovolného počtu různých odkazů a názvů na úložištích.

Proto je rozdělení do bezpečnostních skupin vyhodnocováno odděleně pro unikátní datové obsahy a jednotlivé distribuce aplikací. Jedním z hlavních prvků webové vizualizace jsou grafy, které znázorňují rozdělení datových obsahů a distribucí do jednotlivých skupin:



Obr. 47 Graf zobrazující rozdělení unikátních datových obsahů do skupin nebezpečnosti [zdroj vlastní]



Obr. 48 Graf zobrazující rozdělení distribucí do skupin nebezpečnosti [zdroj vlastní]

Graf vyobrazený na Obr. 47 zobrazuje rozdělení unikátních datových obsahů do skupin podobné nebezpečnosti. Při tvorbě této statistiky, je každý unikátní datový obsah získaný automatizovaným systémem započítán pouze jednou, tedy bez ohledu na počet nalezených distribucí tohoto obsahu.

Naopak při tvorbě druhého grafu na Obr. 48 dochází k rozdělení všech nalezených distribucí do stejných skupin.

10.1.1.1 Míra distribuce škodlivých aplikací

Množina nejnebezpečnějších aplikací – High risk, je v grafech znázorněna fialovou barvou. Pro další analýzu je právě tato skupina tou nejvíce zajímavou. Jedná se o kolekci vysoce nebezpečného malwaru.

V případě, že porovnáme počet unikátních obsahů z této množiny (44) s celkovým počtem jejich distribucí (611), je jasně patrný obrovský nárůst. Číselně se jedná o nárůst přesahující 1400 %. Pro porovnání, například pro množinu Low-risk aplikací, je nárůst počtu distribucí oproti počtu šířených unikátních obsahů pouze 8 %.

Z výše uvedeného jasně vyplývá snaha tvůrců mobilního malwaru, popřípadě jeho distributorů, o dosažení co největšího počtu distribucí nejškodlivějších aplikací. Zahltit neoficiální zdroje škodlivým softwarem a takto maximalizovat šanci na úspěšnou infikaci uživatelských zařízení.

10.1.1.2 Míra zamoření analyzovaných zdrojů

V předchozích kapitolách bylo vysvětleno, že zdroje analyzovaných aplikací jsou převážně neoficiální česká file-share serverová úložiště. Z rozdělení stažených aplikací do skupin nebezpečnosti (Obr. 48) dále vyplývá, že téměř 75 % všech aplikací, získaných z analyzovaných zdrojů, byly prostřednictvím VirusTotal API vyhodnoceny jako škodlivé (tj. S jedním a více testy vyhodnocenými jako hrozba) – tedy označeny jako malware.

Zároveň necelých 50 % z takto označených malware aplikací, patří právě do skupiny High-risk a jedná se tedy o potenciálně velmi škodlivý software. V kombinaci se závěrem popisujícím distribuci škodlivých aplikací, se opět potvrzuje snaha o distribuci škodlivého mobilního malwaru prostřednictvím neoficiálních zdrojů. Zároveň se potvrzuje předpoklad, že neoficiální zdroje jsou stále využívány pro šíření škodlivého softwaru.

10.1.2 Tabulky aktuálních hrozeb

Po vzoru dalších antivirových společností, byl na titulní stránku implementován tabulkový přehled sjednocující často se opakující výsledky.

Obr. 49 Tabulky aktuálních hrozeb uvedené na titulní straně vizualizace [zdroj vlastní]

10.1.2.1 Tabulka – nejnapadenější HASH

V tabulce je uveden seznam pěti HASH identifikátorů. Jedná se o identifikátory unikátních obsahů, které po analýze prostřednictvím VirusTotal API mají nejhorší hodnotu virus-ratio. Jedná se tedy o identifikátory potenciálně nejškodlivějších aplikací.

10.1.2.2 Tabulka – nejnapadenější APK

V této tabulce je uveden seznam pěti vybraných z názvů, pod kterými byly nalezeny distribuce unikátních obsahů z předchozí tabulky.

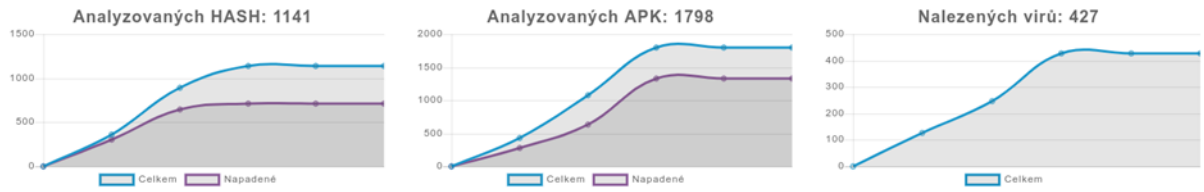
10.1.2.3 Tabulka – nejčastější Malware

Každý z jednotlivých enginů sjednocených v rámci API společnosti VirusTotal, případně nalezenou hrozbu označuje příslušným názvem. V této tabulce jsou uvedeny názvy nejčastěji identifikovaných hrozeb v rámci stažených aplikací.

Všechny tabulky jsou vždy propojeny s detailními informacemi o konkrétní položce uvedené v tabulce.

10.1.3 Grafy časového vývoje

Dalším prvkem titulní strany vizualizace jsou grafy zobrazující průběh analýzy v čase. V grafech jsou uváděny počty stažených unikátních obsahů ke konkrétním datům, stejně jako počty samotných aplikací, které byly staženy. V obou případech jsou pak grafy doplněny o celkový počet zpracovaných souborů. V obou grafech je zároveň vyznačen rozdíl mezi celkovým počtem a počtem souborů, které byly označeny jako škodlivé. Graf zobrazující časový vývoj pro zachycený malware, zobrazuje počet identifikovaných hrozeb v rámci aplikací a jejich celkový počet, rovněž ke konkrétním datům.

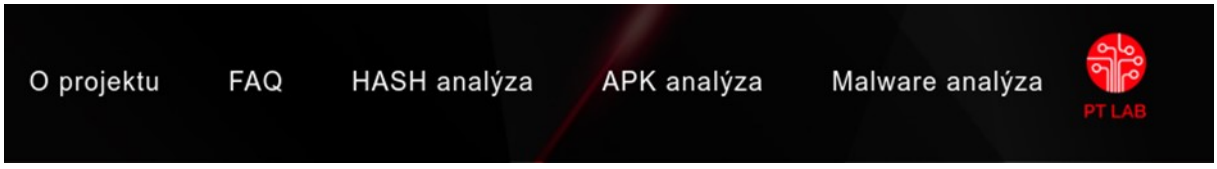


Obr. 50 Grafy časového vývoje uvedené na titulní straně vizualizace [zdroj vlastní]

Pro tvorbu grafů na titulní straně byl využit JavaScript a díky tomu grafy při interakci s kurzorem myši zobrazují detailnější informace o jednotlivých datumech a počtech.

10.2 Jednotlivé záložky s podrobnými výsledky

Kromě titulní přehledu na titulní straně se v rámci webových stránek nachází i další záložky s podrobnými výsledky.

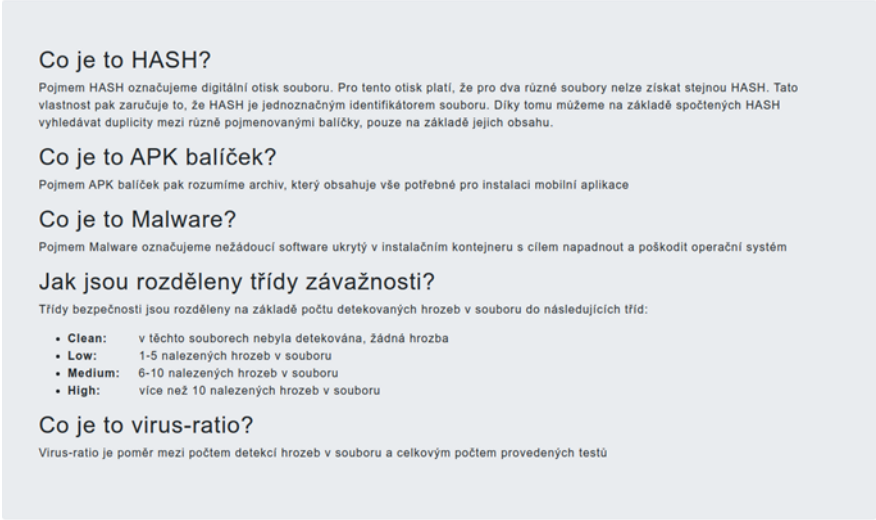


Obr. 51 Detail jednotlivých záložek webové vizualizace [zdroj vlastní]

10.2.1 O projektu + FAQ

V případě těchto záložek se jedná o přehled informací pro uživatele webových stránek. Záložka O projektu seznamuje s projektem, záštitou PTLAB UTB a odkazuje na Fakultu aplikované informatiky.

V záložce FAQ jsou zodpovězeny potenciálně časté dotazy a vysvětleny jednotlivé pojmy jako HASH, APK balíček nebo metriky použité pro rozřazení aplikací. Na tuto stránku je referováno z více míst titulní strany vizualizace, kdy jednotlivé pojmy odkazují právě na své vysvětlení.



Obr. 52 Vysvětlivky uvedené na podstránce FAQ [zdroj vlastní]

10.2.2 HASH analýza

V rámci záložky HASH analýza se nachází tabulka s kompletním seznamem výsledků pro HASH identifikátory:

Název	Virus ratio	Počet duplikátů	Závažnost
c1f80f28f01a7e3dc6b2dad772882e47e8234cb71157b0194524f272d0c32863	034/060	18	H
5c9a3566a4e3d663952cfe361abebad688a79f716971f2024ab6f871761e8991	031/060	6	H
235c381decd8587bf38b5b9f6a50e3e16631c35b87d63f721b8de0e5ac80fe16	031/059	14	H
c2571f3c37ede9e58dae8b03dd87a07d3e153af65a4639bcd6c330d368a73616	031/058	2	H
de468bbad1615b0341fb2124e030904a4bcedd59ea7ad2f685ed3388d217b7a5	030/061	1	H
d258e9243b49f890906bc8bd411a62c7670959782a0edf364e079e7d5c4f7c05	030/060	2	H
c363970a2146b94166fdf78f9184c07c0e9d511f764622670e8711ba33da522e	030/059	17	H
86c4ae4d6f7f06d2e27af1b9f7a2381fb4aa553cef21cab54666689e2e580478	030/058	68	H

Obr. 53 Část tabulky s kompletními výsledky pro jednotlivé HASH dostupná v rámci vizualizace [zdroj vlastní]

V tabulce je vždy uvedena hodnota HASH SHA-256 odpovídající příslušnému datovému obsahu, následně virus-ratio pro danou HASH. Ve sloupci Počet duplikátů je vyjádřen počet zachycených distribucí odpovídajícího datového obsahu napříč analyzovanými zdroji a v posledním sloupci následně zkratka skupiny potenciální nebezpečnosti do které soubor patří.

Všechny HASH identifikátory uvedené v tabulce slouží zároveň jako odkaz na přehled detailních výsledků pro konkrétní soubor. Po otevření tohoto odkazu je uživatel přesměrován na příslušnou stránku:

Obr. 54 Ukázka podstránky s detailními výsledky pro zvolenou hodnotu HASH [zdroj vlastní]

V rámci podstránky s detailními výsledky pro zvolený soubor jsou v hlavičce uvedeny informace – virus ratio, hodnota HASH identifikátoru, třída nebezpečnosti a datum kdy byl tento výsledek získán z VirusTotal API.

Důležitější jsou pak další dva oddíly na stránce – seznam APK balíčků s touto HASH a seznam nalezených hrozeb.

10.2.2.1 Sociotechnické metody distribuce malwaru

V rámci seznamu APK balíčků je uveden počet distribucí, prostřednictvím kterých byla konkrétní HASH k dispozici na nelegálních úložištích a seznam použitých názvů pro distribuci.

Po analýze názvů použitých pro distribuci škodlivých aplikací, byl odhalen nebezpečný trend spojený s šířením mobilního malwaru v rámci neoficiálních úložišť.

V rámci rozřazení datových obsahů a aplikací do skupin nebezpečnosti bylo zjištěno, že nejvíce nebezpečné aplikace jsou aktivně distribuovány v rámci úložišť. Odhalený trend pak ukazuje, že k distribuci škodlivého softwaru tvůrci využívají širokého spektra názvů, pod kterými jsou aplikace dostupné na úložištích.

V rámci uvedeného seznamu by pro uživatele mělo být směrodatné, že jeden datový obsah je šířený pod širokým spektrem názvů aplikací, které se absolutně liší svou funkcionalitou – například pod názvy Whatsapp a Minecraft zároveň. Je principiálně nemožné, aby jeden datový obsah vykazoval funkcionalitu více aplikací.

Pomocí analýzy názvů byly odhaleny sociotechnické metody distribuce mobilního malwaru. Dochází ke zneužívání názvů populárního, často placeného softwaru a pod těmito názvy je distribuován malware. To navazuje na historický kontext uvedený v rámci teoretické části, kdy takto byly šířeny škodlivé aplikace. Uživatelé jsou útočníky oklamání a jistým způsobem motivování ke stahování takto zdarma dostupných aplikací a neuvědomují si riziko spojené s instalací těchto aplikací.

10.2.2.2 Seznam nalezených hrozeb

V rámci tohoto seznamu jsou uvedeny jednotlivé výsledky testů. Konkrétně názvy hrozeb, které jednotlivé testy v aplikaci odhalily. Každá položka tohoto seznamu je odkazem na stránku s podrobnými výsledky pro konkrétní hrozbu. Na této stránce jsou pak uvedeny seznamy HASH identifikátorů a názvů distribucí, ve kterých tato hrozba byla identifikována.

Výzkumní pracovníci si takto mohou seskupit výsledky na základě společné hrozby, pro případnou potřebu analýzy souborů s konkrétní hrozbou (Viz. Oddíl 10.2.4 Malware analýza).

10.2.3 APK analýza

Stejně jako v případě pro jednotlivé HASH je na této záložce uveden tabulkový přehled výsledků pro jednotlivé názvy distribucí.

Název	Virus ratio	Závažnost
Android Minecraft APK CZ(1).apk	034/060	H
HRA MINECRAFT CZ APK(1).apk	034/060	H
Minecraft Hra CZ 1.9.7 APK(1).apk	034/060	H
Minecraft GAME CZ HRA TELEFON FREE APK(1).apk	034/060	H
APK MINECRAFT GAME(1).apk	034/060	H
Minecraft GAME CZ HRA TELEFON FREE APK - kopie(1).apk	034/060	H
MINECRAFT HRA CZ 1.7.9 APK FREE.apk	034/060	H
Minecraft Hra CZ 2018 APK Telefon(1).apk	034/060	H
MINECRAFT CZ HRA APK 2018 1.9.8(1).apk	034/060	H
minecraft-game-apk-free-apk	034/060	H
MINECRAFT CZ GAME FREE APK CZ(1).apk	034/060	H
MINECRAFT GAME APK FREE CZ ANDROID 6(1).apk	034/060	H
Minecraft CZ APK Android(1).apk	034/060	H

Obr. 55 Část tabulky s kompletními výsledky pro jednotlivé distribuce dostupná v rámci vizualizace [zdroj vlastní]

V tabulce je uvedený název, pod kterým probíhala distribuce souboru, virus-ratio a třída nebezpečnosti. Každý z prvků tabulky je opět odkaz na stránku s detailními výsledky. Po kliknutí na odkaz dojde k přesměrování uživatele na stránku s detailními výsledky HASH identifikátoru, který odpovídá zvolené distribuci z tabulky (viz. Předchozí oddíl 10.2.2).

10.2.4 Malware analýza

V rámci záložky Malware analýza je uveden abecední seznam všech hrozeb, které byly identifikovány v rámci jednotlivých testů:

Název
a variant of Android/AdDisplay.AirPush.P potentially unwanted
a variant of Android/AdDisplay.MobiDash.AM potentially unwanted
a variant of Android/Inmobi.C potentially unsafe
a variant of Android/Leadbolt.B potentially unwanted
a variant of Android/Packed.Jiagu.D potentially unsafe
a variant of Android/Remosm.A potentially unsafe
a variant of Android/ScamApp.N potentially unwanted
a variant of Android/Spy.Agent.AHK
a variant of Android/Spy.Agent.AJM
a variant of Android/Spy.Agent.APG

Obr. 56 Část tabulky s kompletními výsledky pro nalezené hrozby dostupná v rámci vizualizace [zdroj vlastní]

Každý prvek je opět odkazem na podstránku s výsledky pro zvolenou hrozbu. V rámci této podstránky je uveden seznam názvů aplikací, ve kterých byla identifikována zvolená hrozba. A následně seznam HASH identifikátorů, které označují unikátní datové obsahy, v rámci kterých byla detekována zvolená hrozba:

Název viru: AdLibrary:Generisk

Počet APK balíčků s tímto virem: 9

- a Camera WiFi LiveStream v1.3.14.apk
- a Advanced Facebook v3.6.11.apk
- JRDPO.apk
- app-misc-merry-christmas-equalizer-music-player-pro-v2.9.8-paid-.apk
- APK Permission Remover (Pro) v1.3.7.apk
- RagingThunder1.0.10.apk
- app-misc-merry-christmas-blue-light-filter-screen-dimmer-for-eye-care-vip-v3.3.2.2.apk
- Max Payne Mobile v1.2 apkmania.com.apk
- MarketMilitiaAMoonForTheSKy.apk

Pocet HASH s tímto virem:9

- 05cc825895de4c7764f0472c7a1495c73a27fc04446602583d30010564747685
- f434e5f425e9b41559b34789289e2d63de19ab48b2f86e52e2ea3cfd8db54175
- 1f118ade35ec58e47b22c8916198c071b7c202e21e86af72d86b5efb5677432e
- 181cdb8841637cfa69285ffecb215c7b5cca79ca7440365e32f34c5ab0bf1e53
- b5bde40f608cb97f13c5f7138527e363e4ac83e25d50b589d792cc04eb647051
- cf48322914a783cd0fec9e225009e152b376a283640144dba3d0a81496b64ddc
- a39576169f29f165093ac0834967de3f17cb41cb93ec99ce1e32e42a1b8c1082
- cfeb3a5c1e07e61d5ffedabaa1342a20baa4cc8d14ed40b2d83c3c9413a0de14
- 9901a94e5c2031006a17f05118a00cdfb0f95ac3d845bee85cac4963acc3138d

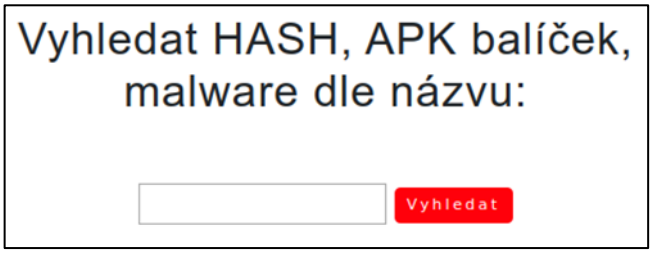
Obr. 57 Ukázka podstránky s detailními výsledky pro zvolenou hrozbu [zdroj vlastní]

Seznam hrozeb v rámci detailních výsledků pro jednotlivé HASH identifikátory odkazuje právě na tyto stránky příslušných hrozeb. A jak již bylo zmíněno, je možné si seskupit soubory na základě nalezené hrozby.

Díky provázanosti stránek podrobných výsledků je pro uživatele jednoduché si všímat názvů používaných k distribuci, seznamu nalezených hrozeb a procházet výsledky analýzy jednotlivých souborů.

10.3 Možnosti využití webové vizualizace

V neposlední řadě je v rámci titulní strany vizualizace implementována možnost vyhledávat výsledky v databázi. Vyhledávání probíhá na základě zadaného názvu pro HASH, APK balíček nebo identifikovanou hrozbu.

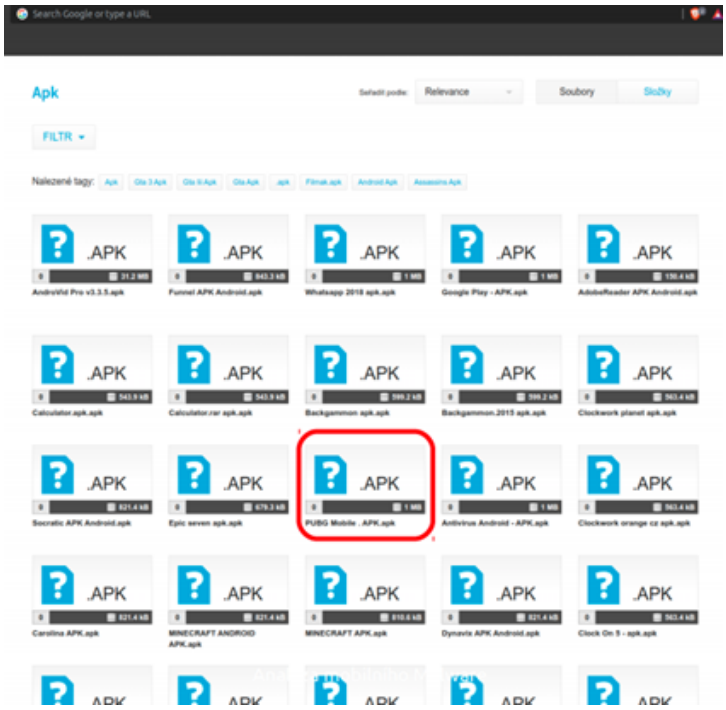


Obr. 58 Vyhledávací pole dostupné
na titulní straně vizualizace [zdroj vlastní]

Po vložení dotazu se spouští PHP skript, který zadané klíčové slovo vyhledá v databázi a v shromáždí seznam možných shod. Tento seznam je následně navrácen uživateli, kdy každý z navrácených prvků slouží jako odkaz na stránku s detailními výsledky. Vyhledávání probíhá v rámci celé databáze a uživatel proto nemusí volit, zda vyhledává hodnotu HASH, název aplikace nebo konkrétní hrozbu.

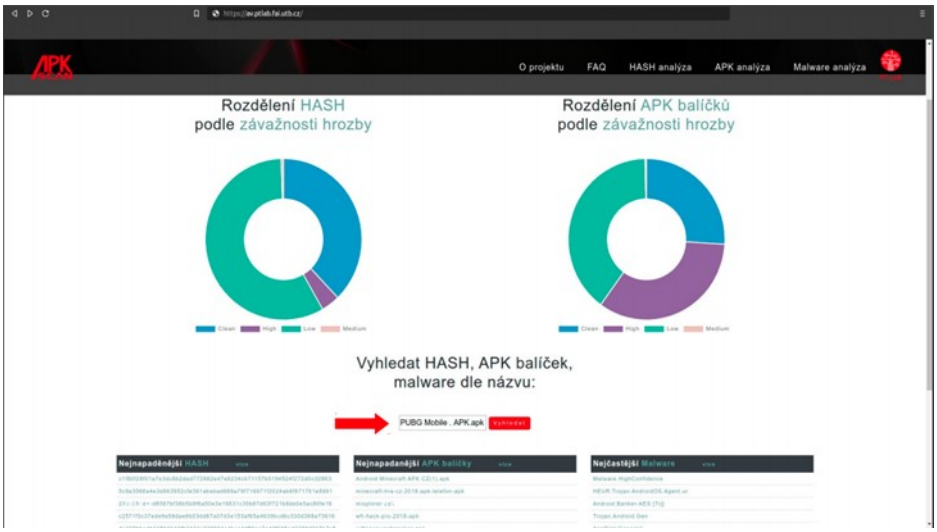
Možnost využití tohoto systému pak spočívá v ověření aplikací dostupných na webových úložištích. Proces bude demonstrován na příkladu.

Uživatel si z webového úložiště chce stáhnout význačnou aplikaci, ale není si jistý bezpečností tohoto souboru:



Obr. 59 Demonstrace možnosti využití
webové vizualizace krok 1. [zdroj vlastní]

V případě, že automatický systém provedl analýzu tohoto úložiště, v databázi se nacházejí výsledky popisující bezpečnost jednotlivých souborů. Uživatel proto může zadat název, pod kterým je soubor k dispozici ke stažení, do vyhledávacího pole na titulní straně (viz Obr. 58).



Obr. 60 Demonstrace možnosti využití webové vizualizace
krok 2. - vyhledání souboru dle názvu [zdroj vlastní]

Pokud se výsledky nacházejí v databázi, seznam možných shod je navrácen uživateli:

Obr. 61 Demonstrace možnosti využití webové vizualizace – výsledky hledání [zdroj vlastní]

Otevřením odkazu pak uživatel přechází na stránku detailních výsledků pro požadovaný soubor a může si ověřit jeho bezpečnost:

Obr. 62 Demonstrace možnosti využití webové vizualizace detailní výsledky pro hledaný soubor [zdroj vlastní]

Přínosem tohoto procesu je možnost pro uživatele si ověřovat bezpečnost souborů pouze na základě názvu dostupné distribuce. Uživatel nemusí příslušný soubor stahovat, aby provedl bezpečnostní test, díky čemuž se nevystavuje riziku spojenému se stahováním a instalací souboru a je maximálně chráněn.

11 VÝSLEDKY STATICKÉ ANALÝZY VYBRANÝCH SOUBORŮ

Rozdělením získaných aplikací vznikly čtyři již zmiňované skupiny aplikací (viz. Oddíl 7.7.1). V rámci skupiny nejvíce nebezpečných aplikací High-risk vzniká unikátní kolekce relevantního malwaru. Z důvodu potenciální škodlivosti a obrovské míře distribuce souborů z této množiny, byla na aplikacích z této kategorie provedena statická analýza s cílem stanovit přesný důvod označení aplikací jako velmi nebezpečný malware a odhalit jejich činnost.

V rámci metodiky byly představeny tři kroky navrženého scénáře analýzy:

- Analýza souboru AndroidManifest.xml
- Analýza zdrojových souborů
- Analýza zdrojového kódu

11.1 Výsledky analýzy souboru AndroidManifest.xml

Při analýze jednotlivých aplikací patřících do množiny High-risk, byla objevena podmnožina aplikací, ve kterých byly soubory manifestu výrazně podobné, takřka stejné.

Nalezený soubor manifestu vypadal přibližně takto:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://
schemas.android.com/apk/res/android" android:sharedUserId="J0bo980P_XlMRqxRh6Cv1NBqq"
android:sharedUserLabel="@string/sharedUserLabel" package="com.android.app"
platformBuildVersionCode="23" platformBuildVersionName="6.0-2438415">
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application android:icon="@mipmap/ic_launcher" android:label="AndroidClient">
        <activity android:name="com.android.app.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Obr. 63 Ukázka souboru manifestu opakujícího se v rámci množiny High-risk s minimální změnou [zdroj vlastní]

Zachycené soubory manifestu měly vždy stejnou strukturu. Opakující se vlastnosti jsou vyznačeny na Obr. 63.

Aplikace patřící do takto vzniklé podmnožiny v souboru manifestu požadovaly pouze jedno oprávnění, konkrétně:

android.permision.WRITE_EXTERNAL_STORAGE

Získáním tohoto oprávnění je aplikaci umožněno manipulovat s datovým úložištěm Android zařízení. Aplikace může vytvářet a mazat soubory v adresářovém systému. Pro zmíněné oprávnění platí, že je vcelku běžně využíváno i v legitimních aplikacích, z tohoto důvodu v této fázi zpracování výsledků nebylo možné stanovit důvod zařazení aplikací do množiny High-risk.

Další vlastností plynoucí z analýzy zmíněných manifestů, je opakující se název pro tzv. package aplikace. Jak je vidět na Obr. 63 název pro package začíná dvojící slov com.android. Tato vlastnost se opakuje v rámci celé nalezené podmnožiny.

Oficiální aplikace přímo zabudované do OS Android obsahují v rámci názvu svého package vždy com.android na začátku (například com.android.calculator je název package vestavěné aplikace kalkulačky Android zařízení). Uvedená skutečnost naznačuje velmi nestandartní chování zachycených aplikací, kdy dochází k padělání názvu pro package.

Zároveň se v rámci zachycených manifestů vyskytují podobné názvy (label) aplikací. Často se vyskytují názvy jako: system, Android, AndroidUpdate, AndroidApplication apod.

Z výše zmiňováních skutečností bylo možné vyvodit závěr, že tvůrci mobilního malwaru dochází k aktivnímu padělání názvů pro aplikaci a package. Padělané aplikace po instalaci do zařízení připomínají názvy systémové aplikace a v telefonu nejsou uživateli podezřelé.

11.2 Výsledky analýzy zdrojových souborů

Metodika tvůrců mobilního malwaru uvedená v rámci předchozího oddílu je nebezpečná svou klamavostí, samotná činnost zmiňovaných aplikací však nebyla odhalena a stále nebylo možné zdůvodnit zařazení aplikací do skupiny High-risk.

Z tohoto důvodu byla nalezená podmnožina podrobena druhému kroku navrhnuté analýzy.

V rámci předchozího kroku byla odhalena snaha o maskování malwaru v zařízení a vydávání aplikací za oficiální software. Na základě toho bylo předpokládáno, že aplikace budou maskovány i jinými způsoby, konkrétně padělaním ikon aplikace.

Následující analýza odhalila, že v rámci odhalené podmnožiny se krom zmiňovaných manifestů a vlastností opakují také použité ikony aplikací.

Ukázka ikon zachycených v rámci podmnožiny:

Obr. 64 Ukázka Ikon zachycených v rámci množiny High-risk [zdroj vlastní]

V případě, že zachycené ikony porovnáme s oficiálními ikonami využívanými v rámci Android OS:

Obr. 65 Ukázka oficiálních ikon využívaných v systému Android [zdroj vlastní]

Je naprosto zřejmá jejich podobnost. Takto byla potvrzena snaha tvůrců mobilního malwaru o aktivní maskování malwaru v zařízení. Aplikace maskované tímto způsobem je velmi obtížné odlišit od oficiálního softwaru v zařízení, viz. Obr. 66

Obr. 66 Ukázka aktivního maskování škodlivých aplikací v zařízení Android [zdroj vlastní]

11.3 Výsledky analýzy zdrojového kódu aplikace.

Předchozími kroky analýzy byla odhalena podmnožina aplikací, jejíž chování je velmi nestandartní a potenciálně nebezpečné. Aplikace z této množiny jsou aktivně maskovány jako součást systému Android, ale režim jejich škodlivé činnosti zatím stále nebyl objasněn.

S cílem odhalit funkcionalitu jednotlivých aplikací byla provedena analýza zdrojového kódu. Jako výsledek analýzy bylo zjištěno, že v rámci podmnožiny se neopakují pouze vlastnosti odhalené předchozími kroky, ale také funkce všech příslušných aplikací je stejná.

Konkrétně se jedná o tři kroky:

11.3.1 Krok první – vytvoření souboru v paměti telefonu a zápis dat

Odhalený zdrojový kód aplikace v prvním kroku kontroluje, zda soubor, který se hodlá vytvořit již v zařízení neexistuje. V případě, že soubor nebyl nalezen, dochází k vytvoření souboru v adresářovém systému a následnému zápisu dat. Data pro zápis jsou získána z adresáře se zdrojovými soubory aplikace.

Zdrojový kód realizující zápis dat:

```
public void install() {
    try {
      if (isInstalled()) {
        Intent intent = new Intent();
        intent.setComponent(new ComponentName(this.id, "com.android.system.MyService"));
        startService(intent);
        return;
      }
      if (!(new
File(getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS).getCanonicalPath() + "/
android.engine.apk")).exists()) {
        (new
File(getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS).getCanonicalPath())).mkdirs();
        (new
FileOutputStream(getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS).getCanonicalPath()
+ "/android.engine.apk")).write(Base64.decode(getString(2130837505), 0));
      }
      run();
      return;
    } catch (Exception exception) {
      return;
    }
  }
```

Obr. 67 Ukázka zdrojového kódu odhaleného malwaru – krok 1 [zdroj vlastní]

Na Obr. 67 je zvýrazněný moment zápisu dat do vytvořeného souboru. Nejen, že dochází k zápisu dat do zařízení bez uživatelova vědomí, ale zároveň je nutné si všimnout, že vkládaný soubor má koncovku .apk (koncovka označuje instalační balíček Android aplikací). V rámci prvního kroku činnosti tedy dochází k vložení nechtěné aplikace do zařízení, bez vědomí uživatele.

11.3.2 Krok druhý – spuštění vložené aplikace

Po vložení dat do zařízení proběhne spuštění instalace vloženého aplikačního balíčku:

```
private void run() {
  try {
    Intent intent = new Intent("android.intent.action.VIEW");
    intent.setDataAndType(Uri.parse("file://" +
getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS +
"/android.engine.apk").getCanonicalPath()), "application/
vnd.android.package-archive");
    startActivity(intent);
    return;
  } catch (Exception exception) {
    return;
  }
}
```

Obr. 68 Ukázka zdrojového kódu odhaleného malwaru – krok 2 [zdroj vlastní]

V rámci zvýrazněného zdrojového kódu na Obr. 68 nejprve dochází k lokalizaci vložené aplikace a následně ke spuštění instalace.

11.3.3 Krok třetí – zobrazení falešné chyby a ukončení původní aplikace

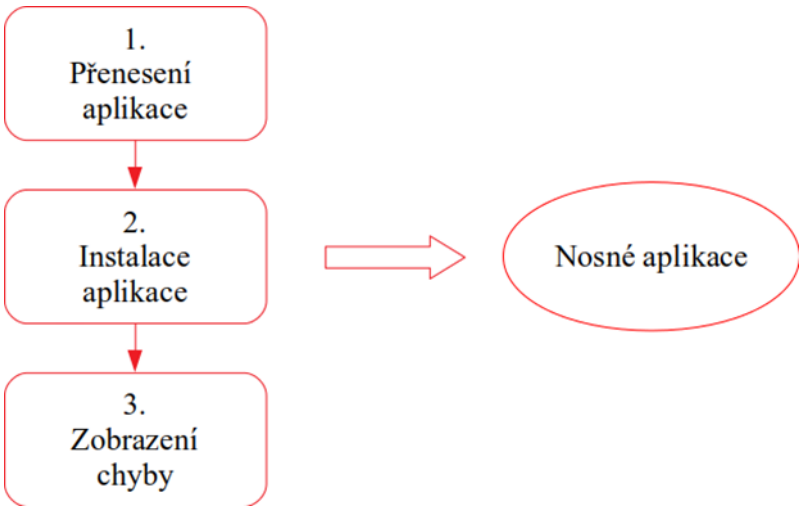
V posledním kroku aplikace, která vkládala do zařízení další aplikaci, zobrazí falešnou hlášku o nekompatibilitě s operačním systémem. V operačním systému Android není tento typ hlášek generován z aplikací, ale jsou v plné režii operačního systému. Po zobrazení falešné chybové hlášky a interakci uživatele, dochází k ukončení aplikace. Tvůrci tohoto malwaru tímto způsobem simulují reálný stav při nekompatibilitě aplikace:

```
private void showAlert() {
  try {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Error 505");
    builder.setMessage("Sorry, this Application is not
compatible with your android version.");
    builder.setCancelable(false);
    builder.setNegativeButton("Close",
new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface
param1DialogInterface, int param1Int)
{ MainActivity.this.finish(); }
        });
    builder.show();
    return;
  } catch (Exception exception) {
    return;
  }
}
```

Obr. 69 Ukázka zdrojového kódu odhaleného malwaru - krok 3 [zdroj vlastní]

11.4 Odhalení množiny nosných aplikací

Pomocí statické analýzy byla odhalena podmnožina aplikací s podobnými vlastnostmi a stejnou funkcionalitou. Aplikace, které patří do této skupiny byly označeny jako množina nosných aplikací:



Obr. 70 Tři kroky činnosti malwarů, které patří do množiny nosných aplikací [zdroj vlastní]

Činnost aplikací patřících do této skupiny je velmi nebezpečná. Nosné aplikace vkládají do zařízení nechtěný software a bez naprostého vědomí uživatele dochází k instalaci aktivního malwaru do zařízení.

Stejnými kroky analýzy pro aplikaci aktivního malwaru bylo odhaleno, že tato aplikace využívá stejné metody aktivního maskování v zařízení, které byly popsány v předchozích oddílech. Opět dochází k padělání systémových názvů. Zároveň bylo odhaleno, že zachycené aplikace aktivního malwaru používají jako ikony průhledné .png soubory. Taková ikona není viditelná v aplikačním menu zařízení.

Výše uvedené skutečnosti zapříčiní, že uživatel má naprosto minimální šanci, že si takto nainstalované aplikace v zařízení všimne. Přirozenou reakcí uživatele na vygenerovanou falešnou hlášku o nekompatibilitě je odinstalování nosné aplikace. V momentě, kdy nosná aplikace zobrazí tuto hlášku, je aktivní malware již nainstalovaný do zařízení a odinstalace nosné aplikace nemá na jeho činnost další vliv. Z důvodu přesvědčení uživatele, že stáhl a odinstaloval nefunkční aplikaci, nedochází ke vzniku jakéhokoliv podezření a pouze se snižuje šance na odhalení aktivního malwaru.

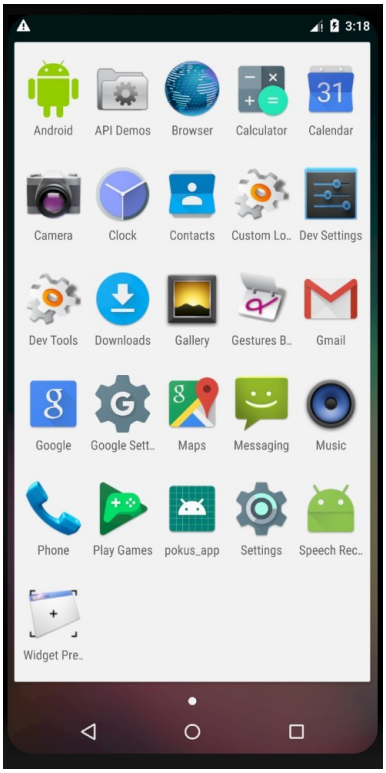
Z těchto důvodů jsou aplikace náležící do této skupiny označeny jako vysoce nebezpečný malware.

11.4.1 Míra distribuce nosných aplikací

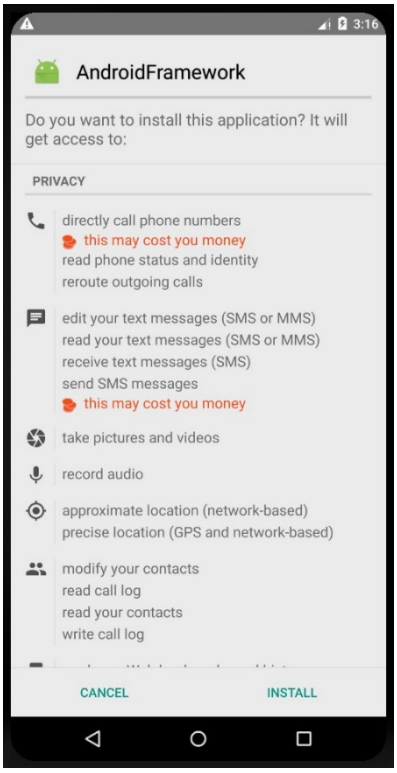
Propojením výsledků analýzy odhalující nosné aplikace se statistikou míry distribuce škodlivého softwaru bylo zjištěno, že v rámci celkového počtu distribucí množiny High-risk, představuje téměř 80 % z tohoto počtu právě distribuce nosných aplikací.

Zároveň bylo zjištěno, že prostřednictvím všech distribucí nosných aplikací, dochází k šíření pouze úzké skupiny aplikací aktivního malwaru. Z těchto skutečností opět vyplývá systematická snaha o distribuci vysoce nebezpečného malwaru prostřednictvím neoficiálních zdrojů.

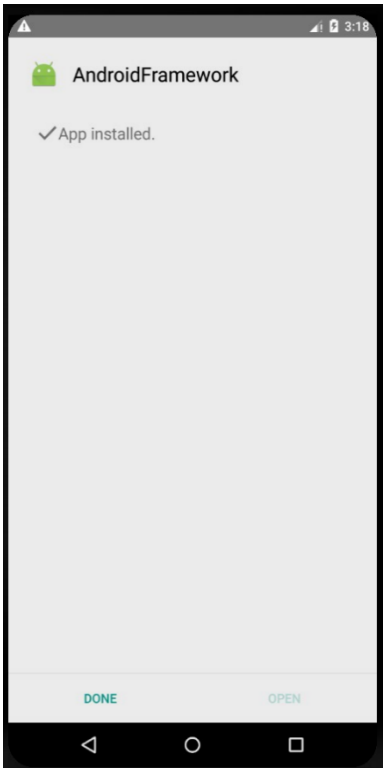
11.5 Ověření výsledků dynamickou analýzou



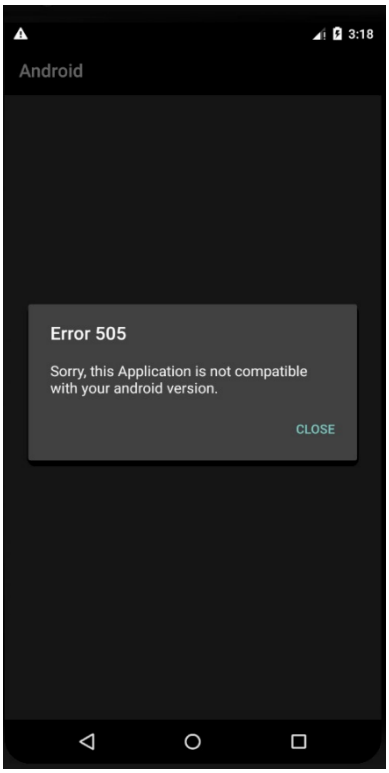
Obr. 71 Dynamická analýza krok 1.: Spuštění nosné aplikace Android v levém horním rohu [zdroj vlastní]



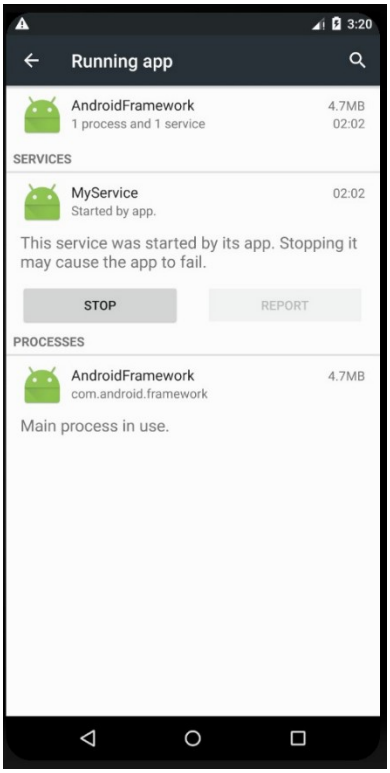
Obr. 72 Dynamická analýza krok 2.: nosná aplikace spouští instalaci vložené aplikace AndroidFramework [zdroj vlastní]



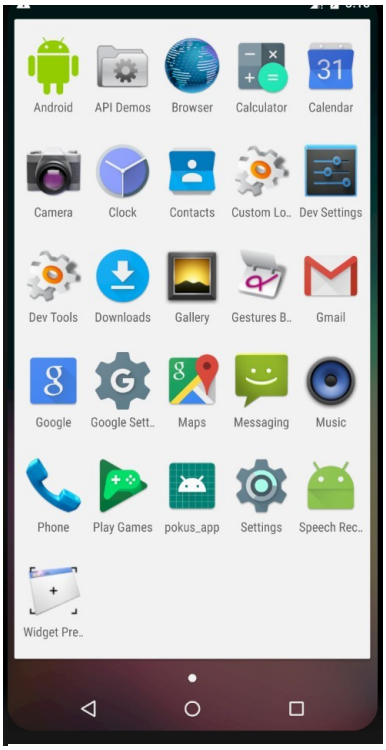
Obr. 75 Dynamická analýza krok 3.: dokončení instalace vkládané aplikace [zdroj vlastní]



Obr. 76 Dynamická analýza krok 4.: nosná aplikace zobrazuje falešnou chybovou hlášku [zdroj vlastní]



Obr. 73 Dynamická analýza krok 5.: Vložená aplikace byla nainstalována a spuštěna v zařízení [zdroj vlastní]



Obr. 74 Dynamická analýza krok 6.: Vložená aplikace není viditelná v uživatelském menu aplikací [zdroj vlastní]

11.6 Aplikace aktivního malwaru

Ze souboru manifestu získaným analýzou aktivního malwaru je patrné, že aplikace aktivního malwaru prostřednictvím velkého počtu oprávnění přebírá nad mobilní zařízením značnou kontrolu:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://
schemas.android.com/apk/res/android" android:installLocation="auto"
android:sharedUserId="JOho980P.XlMRqxRh6Cv1NBqq" android:sharedUserLabel="@string/
sharedUserLabel" package="com.android.system" platformBuildVersionCode="1"
platformBuildVersionName="9b357820-8458-4e23-b51e-9d1540e434ca">
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION"/>
    <uses-permission android:name="android.permission.DUMP"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission
android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
    <uses-permission android:name="android.permission.SEND_RESPOND_VIA_MESSAGE"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.SET_WALLPAPER"/>
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
    <uses-permission android:name="android.permission.TRANSMIT_IR"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
    <uses-permission android:name="android.permission.WRITE_SECURE_SETTINGS"/>
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <uses-permission
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
    <uses-permission
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <uses-permission
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
    <uses-permission
android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS"/>
    <uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
    <uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/
>
    <uses-feature android:name="android.hardware.bluetooth" android:required="false"/>
    <uses-feature android:name="android.hardware.camera" android:required="false"/>
    <uses-feature android:name="android.hardware.camera.autofocus"
android:required="false"/>
    <uses-feature android:name="android.hardware.location" android:required="false"/>
    <uses-feature android:name="android.hardware.microphone" android:required="false"/>
    <uses-feature android:name="android.hardware.telephony" android:required="false"/>
    <uses-feature android:name="android.hardware.wifi" android:required="false"/>
    <application android:hardwareAccelerated="true" android:label="System"
android:largeHeap="true" android:name="activities.Application">
        <service android:directBootAware="true" android:exported="true"
```

Obr. 77 Ukázka souboru manifestu aplikace aktivního malwaru [zdroj vlastní]

Vzhledem k velmi malé šanci na odhalení instalace aktivního malwaru vzniká pro uživatele zařízení obrovské riziko.

Z oprávnění uvedených v manifestu aktivního malwaru lze vyvodit několik hypotetických scénářů pro útok na uživatele.

Ukázková varianta 1:

Kombinací následujících oprávnění:

- android.permission.RECEIVE_SMS
- android.permission.READ_SMS
- android.permission.WRITE_SMS
- android.permission.SEND_SMS

Získá aktivní malware kompletní možnost pracovat s SMS zprávami. Aplikace by byla schopná kontrolovat jejich přijímaní, odesílání, číst jejich obsah nebo SMS zprávy mazat. Kombinací těchto skutečností může docházet k útoku zmiňovaném v rámci teoretické části – odesílání SMS zpráv na čísla se zvýšenou sazbou.

Jak již bylo řečeno, v rámci systému Android jsou na příchozí zprávu nejdříve upozorňovány procesy, které čekají na tuto událost a po obsloužení jednotlivých procesů dochází k upozornění uživatele. Uvedená skutečnost pak znamená, že v případě, kdy malware aplikace po přijetí SMS provede smazání zprávy, upozornění pro uživatele nebude vytvořeno. To umožňuje nepozorované přijímání a odesílání zpráv.

Výše uvedeného scénáře mazání příchozích zpráv využívají aplikace malwaru cílících na příchozí SMS z bank. Může docházet ke krádeži a odesílání příchozích dat a v případě kombinovaného útoku na internetové bankovnictví zároveň k zachytávání potvrzovacích kódu z banky a přeposílání zpět útočníkům. Zachycená SMS z banky je po získání dat odstraněna a uživatel neví, že dochází k autorizaci činnosti v rámci bankovnictví.

Ukázková varianta 2:

Další možná kombinace oprávnění:

- android.permission.INTERNET
- android.permission.DOWNLOAD_WITHOUT_NOTIFICATION
- android.permission.WRITE_EXTERNAL_STORAGE

Nahrává variantám scénářů ransomware útoků, nebo jiné manipulaci s úložištěm, popřípadě nepozorovanému stahování nechtěných souborů a úniku dat.

12 ZÁVĚREČNÁ SHRNUTÍ A VÝSTUPY PRÁCE

Výzkum, který byl v rámci bakalářské práce proveden, trval dva roky. Relevanci získaných výsledků lze doložit zvanou přednáškou pro Policii České republiky, Kriminalistický ústav a útvar OKTE (přednáška proběhla v roce 2019). Ve stejném roce byla práce představena na konference Řízení procesů a aplikace moderních technologií – Kybernetická bezpečnost UTB 2019. Jednotlivé části práce byly také prezentovány v rámci soutěží STOČ 2019 a 2020, na kterých byly vyhodnoceny jako vítězný příspěvek ve své kategorii.

Pro účely výzkumu vznikl unikátní automatizovaný systém, schopný vyhledávat mobilní aplikace na neoficiálních distribučních zdrojích a automaticky vyhodnocovat jejich nebezpečnost. Získaná data jsou laické i odborné veřejnosti k dispozici prostřednictvím webové vizualizace. Vizualizace slouží jako zdroj informací s cílem rozšířit povědomí o problematice mobilního malwaru. Vizualizace rovněž obsahuje funkcionalitu umožňující uživatelům určit míru nebezpečnosti mobilního softwaru (aniž by sami museli provádět stahování nebo instalaci podezřelých aplikací).

Unikátnost popsaného systému spočívá v plošném vyhodnocování bezpečnostní situace v reálném čase a shromažďování APK souborů pro další analýzu.
Získaná unikátní množina relevantního malwaru byla podrobena metodám statické analýzy, díky čemuž byly odhaleny:

- techniky tvůrců mobilního malwaru využívané při tvorbě a distribuci mobilního malwaru,
- postupy aktivního maskování škodlivých aplikací v mobilních zařízeních,
- vzorky velmi nebezpečných nosných aplikací, jejichž cílem byla skrytá infekce koncového zařízení uživatele.

Provedený výzkum umožnil stanovit míru zamoření neoficiálních úložišť škodlivým mobilním softwarem a stanovit míru distribuce jednotlivých podmnožin analyzovaných malwarů a čistých aplikací.
Výsledky práce umožnily vznik souboru metodik, které budou použity v mobilní sekci PTLAB UTB pro penetrační testování mobilních aplikací.

Výsledky zjištěné v rámci výzkumu naznačují vysokou míru nebezpečí, jež jsou spojena s instalací mobilních aplikací z neoficiálních zdrojů, jako jsou file-share servery. Na základě provedeného výzkumu lze zformulovat doporučení pro laickou veřejnost, kterým je instalace aplikací pouze z oficiálních distribučních platforem a vyhýbání se softwaru z ne oficiálních zdrojů.

13 ZAMĚŘENÍ PRO BUDOUCÍ VÝZKUM

Výsledky práce ukazují možné perspektivní směry dalšího výzkumu.

Jako zajímavá se například jeví spolupráce s laboratoří umělé inteligence AILAB UTB s cílem vytvořit klasifikátor mobilního malwaru, který by využíval metody statické analýzy v kombinaci s metodami umělé inteligence a teorií grafů.

Dalším možným směrem výzkumu je optimalizace činnosti automatizovaného systému pro analýzu, který byl vytvořen v rámci bakalářské práce.

Vážná bezpečnostní situace týkající se mobilního malwaru distribuovaného prostřednictvím file-share serverů naznačuje potřebu systematického výzkumu dalších neoficiálních distribučních platforem, jako jsou torrentová úložiště a veřejně dostupné FTP servery.

SEZNAM POUŽITÉ LITERATURY

[1] The history of Android OS: its name, origin and more. Androidauthority [online]. August 18, 2019 [cit. 2020-05-26]. Dostupné z: https://www.androidauthority.com/history-android-os-name-789433/

[2] Graf počtu stažených aplikací. Statista [online]. [cit. 2020-08-04]. Dostupné z: https://www.statista.com/statistics/734332/google-play-app-installs-per-year/

[3] Smartphone Market Share.[online]. Apr 2 2020 [cit. 2020-05-26]. Dostupné z: https://www.idc.com/promo/smartphone-market-share/os

[4] Roll call release [online]. FBI, US Department of homeland security, 23 July 2013 [cit. 2020-05-28]. Dostupné z: https://info.publicintelligence.net/DHS-FBI-AndroidThreats.pdf

[5] Cyber attacks on Android devices [online]. 11/07/2018 [cit. 2020-05-26]. Dostupné z: https://www.gdatasoftware.com/blog/2018/11/31255-cyber-attacks-on-android-devices-on-the-rise

[6] The number of mobile malware attacks doubles in 2018, as cybercriminals sharpen their distribution strategies. Https://www.kaspersky.com/ [online]. March 5, 2019 [cit. 2020-05-26]. Dostupné z: https://www.kaspersky.com/about/press-releases/2019_the-number-of-mobile-malware-attacks-doubles-in-2018-as-cybercriminals-sharpen-their-distribution-strategies

[7] More than one billion Android devices at risk of malware threats. Https://www.which.co.uk/ [online]. 6 Mar 2020 [cit. 2020-05-26]. Dostupné z: https://www.which.co.uk/news/2020/03/more-than-one-billion-android-devices-at-risk-of-malware-threats/

[8] Malware is taking an increasingly large toll. [online]. 2020 [cit. 2020-05-28]. Dostupné z: https://www.safetydetectives.com/blog/malware-statistics/

[9] Co je malware? Eset [online]. [cit. 2020-07-31]. Dostupné z: https://www.eset.com/cz/malware/

[10] DUNHAM, Ken, 31st October 2008n. l. Mobile Malware Attacks and Defense. Syngress. ISBN 9781597492980.

[11] RAMU, Srikanth.V., 2012. Mobile Malware Evolution, Detection and Defense. University of British Columbia.

[12] Bluetooth-Worm:SymbOS/Cabir. F-Secure [online]. [cit. 2020-07-22]. Dostupné z: https://www.f-secure.com/v-descs/cabir.shtml

[13] First SMS Trojan detected for smartphones running Android. Kaspersky lab [online]. [cit. 2020-07-22]. Dostupné z: https://web.archive.org/web/20111231154719/http://www.kaspersky.com/about/news/virus/2010/First_SMS_Trojan_detected_for_smartphones_running_Android

[14] Android Mobile Security Threats. Kaspersky [online]. [cit. 2020-07-22]. Dostupné z: https://www.kaspersky.com/resource-center/threats/mobile

[15] Buing into Mobile Security: Mobile security investments are becoming a priority for CIOs as the lack of visibility into mobile devices continues to increase risk. International Data Group [online]. International Data Group [cit. 2020-07-22]. Dostupné z: https://info.lookout.com/rs/051-ESQ-475/images/idg-report-buying-into-mobile-security.pdf

[16] Stagefright vulnerability allows criminals to send malware by text. CSO [online]. CSO [cit. 2020-07-22]. Dostupné z: https://www.csoonline.com/article/2952741/stagefright-vulnerability-allows-criminals-to-send-malware-by-text.html

[17] OSBORNE, Charlie. Mobile malware evolves: Adware now breaks and roots your phone. ZDNet [online]. [cit. 2020-07-22]. Dostupné z: https://www.zdnet.com/article/mobile-malware-evolves-adware-now-breaks-and-roots-your-phone/

[18] BROOK, Chris. New Banking Trojan Targets Android, Steals SMS. Threat post [online]. [cit. 2020-07-22]. Dostupné z: https://threatpost.com/new-banking-trojan-targets-android-steals-sms/110819/

[19] Google's Android OS: Past, Present and Future. Phone Arena [online]. [cit. 2020-07-22]. Dostupné z: https://www.phonearena.com/news/Googles-Android-OS-Past-Present-and-Future_id21273

[20] ALABASTER, Jay. Android founder: We aimed to make a camera OS. PC world [online]. [cit. 2020-07-22]. Dostupné z:

https://www.pcworld.com/article/2034723/android-founder-we-aimed-to-make-a-camera-os.html

[21] ELGIN, Ben. Google Buys Android for Its Mobile Arsenal. Bloomberg Businessweek [online]. [cit. 2020-07-22]. Dostupné z: https://web.archive.org/web/20110205190729/http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm

[22] T-Mobile officially announces the G1 Android phone. Tech Crunch [online]. [cit. 2020-07-22]. Dostupné z: https://techcrunch.com/2008/09/23/t-mobile-officially-announces-the-g1-android-phone/?guccounter=1

[23] RAJA, Haroon Q. Android Partitions Explained: boot, system, recovery, data, cache & misc. Addictive tips [online]. [cit. 2020-07-22]. Dostupné z: https://www.addictivetips.com/mobile/android-partitions-explained-boot-system-recovery-data-cache-misc/

[24] Canalys: iPhone outsold all Windows Mobile phones in Q2 2009. Apple insider [online]. [cit. 2020-07-22]. Dostupné z: https://appleinsider.com/articles/09/08/21/canalys_iphone_outsold_all_windows_mobile_phones_in_q2_2009.html

[25] Google's Android becomes the world's leading smart phone platform. Canalys [online]. [cit. 2020-07-22]. Dostupné z: https://www.canalys.com/newsroom/google%E2%80%99s-android-becomes-world%E2%80%99s-leading-smart-phone-platform

[26] Android Marks Fourth Anniversary Since Launch with 75.0% Market Share in Third Quarter, According to IDC. IDC [online]. [cit. 2020-07-22]. Dostupné z: https://web.archive.org/web/20121103041944/http://www.idc.com/getdoc.jsp?containerId=prUS23771812

[27] Graf podílu operačních systémů na světovém trhu. *Statista* [online]. [cit. 2020-08-09]. Dostupné z: https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/

[28] CALLAHAN, John. Android now running on over 2.5 billion active hardware devices. Android Authority [online]. [cit. 2020-07-22]. Dostupné z: https://www.androidauthority.com/android-2-5-billion-devices-983534/

[29] CLEMENT, J. Share of mobile phone website traffic worldwide 2018. Statista [online]. [cit. 2020-07-22]. Dostupné z: https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/

[30] CASTILLO, Carlos A., 2011. Android Malware Past, Present, and Future [online]. McAfee, 4-9 [cit. 2020-07-23]. Dostupné z: https://pdfs.semanticscholar.org/5735/6502310474ba9564ec8f581494b8de50b3e5.pdf

[31] Warning on possible Android trojans. F-secure archiv [online]. [cit. 2020-07-23]. Dostupné z: https://archive.f-secure.com/weblog/archives/00001852.html

[32] AndroidOS.Tapsnake: Watching Your Every Move. Broadcom [online]. [cit. 2020-07-23]. Dostupné z: https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=c07a0465-80ae-4c8c-bff7-4e3befd5f229&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments

[33] Update: Security Alert: DroidDream Malware Found in Official Android Market. Lookout archiv [online]. [cit. 2020-07-23]. Dostupné z: https://web.archive.org/web/20110310122738/https://blog.mylookout.com/2011/03/security-alert-malware-found-in-official-android-market-droiddream/

[34] The Mother Of All Android Malware Has Arrived: Stolen Apps Released To The Market That Root Your Phone, Steal Your Data, And Open Backdoor. Android Police [online]. [cit. 2020-07-23]. Dostupné z: https://www.androidpolice.com/2011/03/01/the-mother-of-all-android-malware-has-arrived-stolen-apps-released-to-the-market-that-root-your-phone-steal-your-data-and-open-backdoor/

[35] Android.Bgserv Found on Fake Google Security Patch. Broadcom [online]. [cit. 2020-07-23]. Dostupné z: https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=baca0fbc-87e4-4428-82d3-5cb6a728e0ea&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments

[36] Update: Security Alert: DroidDreamLight, New Malware from the Developers of DroidDream. Lookout archiv [online]. [cit. 2020-07-23]. Dostupné z:

https://web.archive.org/web/20110603132350/http://blog.mylookout.com/2011/05/security-alert-droiddreamlight-new-malware-from-the-developers-of-droiddream/

[37] Cyber attacks on Android devices on the rise. GDATA software [online]. [cit. 2020-07-23]. Dostupné z: https://www.gdatasoftware.com/blog/2018/11/31255-cyber-attacks-on-android-devices-on-the-rise

[38] Roll call release [online]. FBI, US Department of homeland security, 23 July 2013 [cit. 2020-05-28]. Dostupné z: https://info.publicintelligence.net/DHS-FBI-AndroidThreats.pdf

[39] THOMAS, Daniel R., Alastair R. BERESFORD a Andrew RICE, 2015. Security Metrics for the Android Ecosystem. In: Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices - SPSM '15 [online]. New York, New York, USA: ACM Press, s. 87-98 [cit. 2020-07-23]. DOI: 10.1145/2808117.2808118. ISBN 9781450338196. Dostupné z: http://dl.acm.org/citation.cfm?doid=2808117.2808118

[40] The Biggest Splash at BlackHat and DEFCON 2015. Zimperium [online]. [cit. 2020-07-23]. Dostupné z: https://blog.zimperium.com/the-biggest-splash-at-blackhat-and-defcon-2015/

[41] The number of mobile malware attacks doubles in 2018, as cybercriminals sharpen their distribution strategies. Kaspersky [online]. [cit. 2020-07-23]. Dostupné z: https://www.kaspersky.com/about/press-releases/2019_the-number-of-mobile-malware-attacks-doubles-in-2018-as-cybercriminals-sharpen-their-distribution-strategies

[42] More than one billion Android devices at risk of malware threats. Https://www.which.co.uk/ [online]. 6 Mar 2020 [cit. 2020-05-26]. Dostupné z: https://www.which.co.uk/news/2020/03/more-than-one-billion-android-devices-at-risk-of-malware-threats/

[43] Malware is taking an increasingly large toll. [online]. 2020 [cit. 2020-05-28]. Dostupné z: https://www.safetydetectives.com/blog/malware-statistics/

[44] Is Android OS a Viable Option for Industrial Mobile Computers? Lowry Solutions [online]. [cit. 2020-07-23]. Dostupné z: https://lowrysolutions.com/blog/is-android-os-a-viable-option-for-industrial-mobile-computers/

[45] Android Enterprise Overview. Google developers [online]. [cit. 2020-07-23]. Dostupné z: https://developers.google.com/android/work/overview#managed_google_play

[46] Android Device Policy application. Google Play [online]. [cit. 2020-07-23]. Dostupné z: https://play.google.com/store/apps/details?id=com.google.android.apps.work.clouddpc

[47] Handheld Computers. Honeywell [online]. [cit. 2020-07-23]. Dostupné z: https://www.honeywellaidc.com/products/computer-devices/handheld

[48] Pokladní terminál Smart 8. SEP system [online]. [cit. 2020-07-23]. Dostupné z: https://sepsystem.cz/produkt/pokladni-terminal-smart-8#1100090

[49] SBC Micro Browser aplikace. Google Play [online]. [cit. 2020-07-23]. Dostupné z: https://play.google.com/store/apps/details?id=com.saiaburgess.microbrowser

[50] Saia Burgess Controls. Saia Burges Controls [online]. [cit. 2020-07-23]. Dostupné z: https://www.saia-pcd.com/en-gb/

[51] The Android operating system in industrial applications. Machine Building [online]. [cit. 2020-07-23]. Dostupné z: https://www.machinebuilding.net/ta/t0693.htm

[52] YAGHMOUR, Karim, 2013. Embedded Android. O'Reilly Media. ISBN 9781449308292.

[53] VEERABAHU, Maharajan. Embedded related [online]. [cit. 2020-07-24]. Dostupné z: https://www.embeddedrelated.com/showarticle/1107.php

[54] Pros and Cons of Using Embedded Android for a Non-mobile Device. Intellectsoft [online]. [cit. 2020-07-24]. Dostupné z: https://www.intellectsoft.net/blog/pros-and-cons-of-using-embedded-android-for-a-non-mobile-device/

[55] Android Studio aplikace. Android Developer [online]. [cit. 2020-07-24]. Dostupné z: https://developer.android.com/studio

[56] KUMAR, Rajesh et al., 2019. Research on Data Mining of Permission-Induced Risk for Android IoT Devices. Applied Sciences [online]. 9(2) [cit. 2020-07-24]. DOI: 10.3390/app9020277. ISSN 2076-3417. Dostupné z: http://www.mdpi.com/2076-3417/9/2/277

[57] ThingOS. ThingOS [online]. [cit. 2020-07-24]. Dostupné z: https://thingos.io/

[58] Operátor-tablet. In: Mainstaycompany [online]. [cit. 2020-08-03]. Dostupné z: https://www.mainstaycompany.com/wp-content/uploads/2015/06/manufacturing-header-660x436.png

[59] Graf – zastoupení OS. In: Statcounter [online]. [cit. 2020-08-03]. Dostupné z: https://gs.statcounter.com/os-market-share#monthly-201906-202006-bar

[60] TeslaSCADA. In: Teslascada [online]. [cit. 2020-08-03]. Dostupné z: https://teslascada.com/

[61] Report popisující problematiku Stuxnet. In: Wired [online]. [cit. 2020-08-03]. Dostupné z: https://www.wired.com/images_blogs/threatlevel/2010/11/w32_stuxnet_dossier.pdf

[62] Operátor-android-diagnostika. In: Lokalkompass [online]. [cit. 2020-08-04]. Dostupné z: https://www.lokalkompass.de/event/bottrop/c-information/debattenabend-der-linken-zu-industrie-40_e17619#gallery=default&pid=24438

[63] Android application fundamentals. Android developer [online]. [cit. 2020-07-24]. Dostupné z: https://developer.android.com/guide/components/fundamentals

[64] Android Runtime and Dalvik. Android Source [online]. [cit. 2020-07-25]. Dostupné z: https://source.android.com/devices/tech/dalvik

[65] EILAM, Eldad, c2005. Reversing secrets of reverse engineering. Indianapolis: Wiley. ISBN 978-0-7645-7481-8.

[66] Nástroj APKTOOL. Ibotpeaches [online]. [cit. 2020-08-05]. Dostupné z: https://ibotpeaches.github.io/Apktool/

[67] Nástroj dex2jar. Github - pxp1988/dex2jar [online]. [cit. 2020-07-25]. Dostupné z: https://github.com/pxb1988/dex2jar/wiki

[68] Nástroj JD-GUI. Java Decompiler [online]. [cit. 2020-07-25]. Dostupné z: https://java-decompiler.github.io/

[69] Nástroj JADx. Github - Skylot/JADx [online]. [cit. 2020-07-25]. Dostupné z: https://github.com/skylot/jadx

[70] JADx - features. Github - Skylot/JADx [online]. [cit. 2020-07-25]. Dostupné z: https://github.com/skylot/jadx/wiki/jadx-gui-features-overview

[71] Nástroj JEB Decompiler. PNF Software [online]. [cit. 2020-07-27]. Dostupné z: https://www.pnfsoftware.com/jeb/

[72] LI, Jingwei, Bozhi WU a Weiping WEN, 2019. Android Malware Detection Method Based on Frequent Pattern and Weighted Naive Bayes. YUN, Xiaochun et al., ed. Cyber Security [online]. Singapore: Springer Singapore, s. 36-51 [cit. 2020-07-27]. Communications in Computer and Information Science. DOI: 10.1007/978-981-13-6621-5_4. ISBN 978-981-13-6620-8. Dostupné z: http://link.springer.com/10.1007/978-981-13-6621-5_4

[73] Domovská stránka VirusShare. VurusShare [online]. [cit. 2020-07-27]. Dostupné z: https://virusshare.com/

[74] KUMAR, Rajesh et al., 2019. Research on Data Mining of Permission-Induced Risk for Android IoT Devices. Applied Sciences [online]. 9(2) [cit. 2020-07-24]. DOI: 10.3390/app9020277. ISSN 2076-3417. Dostupné z: http://www.mdpi.com/2076-3417/9/2/277

[75] Domovská stránka VirusTotal. VirusTotal [online]. [cit. 2020-07-27]. Dostupné z: https://www.virustotal.com/gui/home/upload

[76] DOĞRU, İbrahim a Ömer KİRAZ, 2018. Web-Based Android Malicious Software Detection and Classification System. Applied Sciences [online]. 8(9) [cit. 2020-07-27]. DOI: 10.3390/app8091622. ISSN 2076-3417. Dostupné z: http://www.mdpi.com/2076-3417/8/9/1622

[77] YERIMA, Suleiman Y., Mohammed K. ALZAYLAEE a Sakir SEZER, 2019. Machine learning-based dynamic analysis of Android apps with improved code coverage. EURASIP Journal on Information Security [online]. 2019(1) [cit. 2020-07-27]. DOI: 10.1186/s13635-019-0087-1. ISSN 2510-523X. Dostupné z: https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-019-0087-1

[78] LIU, Yu et al., 2018. Detecting Android Malwares with High-Efficient Hybrid Analyzing Methods. Mobile Information Systems [online]. 2018, 1-12 [cit. 2020-07-27]. DOI: 10.1155/2018/1649703. ISSN 1574-017X. Dostupné z: https://www.hindawi.com/journals/misy/2018/1649703/

[79] Laboratoř umělé inteligence. AILAB UTB [online]. [cit. 2020-08-05]. Dostupné z: https://ailab.fai.utb.cz/

[80] Laboratoř penetračního testování. PTLAB UTB [online]. [cit. 2020-08-05]. Dostupné z: https://ptlab.fai.utb.cz/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AILAB	Artificial Intelligence Laboratory
AOSP	Android open source project
API	Application Programming Interface
APK, .apk	Android application package
ARM	Advanced RISC Machines
BT	Bluetooth
BYOD	Bring your own device
CLI	Command line interface
CSS	Cascading Style Sheets
DNS	Domain Name System
.exe	Executable file
FAI	Fakulta aplikované informatiky
FBI	Federal Bureau of Investigation
FTP	File Transfer Protocol
GPS	Global Positioning System
HMI	Human Machine Interface
HTC	High Tech Computer corporation
HTML	Hyper Text Markup Language
IDG	International data group
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
Inc.	Incorporated
IoT	Internet of things
JD-GUI	Java decompiler – Graphical user interface
LCD	Liquid Crystal Display

LLC	Limited liability company
LTS	Long Term Support
MMS	Multimedia Messaging Service
OKTE	Odbor kriminalistické techniky a expertíz
OS	Operating system
PC	Personal computer
PLC	Programmable logic controller
PTLAB	Penetration Testing Laboratory
RAM	Random access memory
RFID	Radio-frequency identification
SBC	Saia Burgess Controls
SCADA	Supervisory control and data acquisition
SDK	Software Development Kit
SMS	Short message service
SQL	Structured Query Language
STOČ	Studentská tvůrčí a odborná činnost
URL	Uniform Resource Locator
USD $	United States Dollar
UTB	Univerzita Tomáše Bati ve Zlíně
Wi-Fi	Wireless Fidelity

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD

Přiložené CD obsahuje:

- Bakalářskou práci ve formátu .pdf: BP_JanKincl_2020.pdf