Rok akademicki 2014 / 2015

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

PRACA DYPLOMOWA MAGISTERSKA

Cezary Zawadka

Wykrywanie złośliwych aplikacji w systemie Android na podstawie różnicowej analizy uprawnień

Opiekun pracy
prof. nzw. dr hab. Piotr Gawrysiak

Ocena: .....

.....

Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Kierunek: Informatyka

Specjalność: Inżynieria Systemów Informatycznych

Data urodzenia: 1988.09.24

Data rozpoczęcia studiów: 2012.10.01

Życiorys

Urodziłem się 24 września 1988 r. w Warszawie. W latach 2004 do 2007 uczęszczałem do XIV Liceum Ogólnokształcącego im. Stanisława Staszica w Warszawie. Od października 2007 do września 2012 studiowałem na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na kierunku Telekomunikacja. Po uzyskaniu tytułu inżyniera rozpocząłem studia magisterskie na tym samym wydziale na kierunku Informatyka.

.....
Podpis studenta

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu .....20__ r

z wynikiem .....

Ogólny wynik studiów: .....

Dodatkowe wnioski i uwagi Komisji: .....

.....

.....

STRESZCZENIE

Rosnąca popularność urządzeń mobilnych, połączona z olbrzymim sukcesem systemu Android, który zainstalowany jest na ponad 80% nowych urządzeń, powodują drastyczny wzrost ilości złośliwego oprogramowania. Niniejsza praca zawiera analizę dostępnych mechanizmów bezpieczeństwa, przedstawi największe zagrożenia przedstawia nowego typu oprogramowania analizującego niebezpieczne aplikacje systemu Android. Innowacyjny pomysł polega na dynamicznej analizie uprawnień aplikacji w porównaniu z uprawnieniami innych aplikacji o podobnych funkcjach w celu automatycznego wykrywania zbyt liberalnego dostępu. Wyniki analizy zagrożenia są porównywalne z najnowszymi algorytmami akademickimi dostępnymi w literaturze przedmiotu.

Słowa kluczowe:
Urządzenia mobilne, Android, Bezpieczeństwo, Linux, antywirus, Analiza zagrożeń

Android Malware Detection Based on Cross Application Permission Analysis

Popularity of smart mobile devices combined with overwhelming success of Android operating system, which is now installed on over 80% of new devices, results in dramatic rise in malicious software aimed at Android. This thesis provides detailed analysis of available security features of Android operating system, application distribution chain and third party antivirus software. Furthermore new platform for malware detection is proposed. By comparing permissions between similar application it has achieved detection results comparable with state of the arts academic solutions.

Keywords:
Mobile security, Android, Malware detection, Linux, Information System Security

Spis treści

1. Wstęp

Rosnąca popularność urządzeń mobilnych, połączona z olbrzymim sukcesem systemu Android, który zainstalowany jest na ponad 80% nowych urządzeń [2], powodują drastyczny wzrost ilości złośliwego oprogramowania. Dodatkowo, ilość danych prywatnych i korporacyjnych dostępnych na telefonach i tabletach, niewystarczające zabezpieczenia, niska świadomość użytkowników oraz łatwy sposób spieniężenia przeprowadzonych ataków zwiększają atrakcyjność platformy Android w oczach hakerów łamiących prawo. Według raportów firm produkujących programy antywirusowe ponad 90% złośliwego oprogramowania na urządzenia mobilne dotyczy właśnie Androida [1][3].

Z drugiej strony firma Google, która jest najbardziej odpowiedzialna za rozwój systemu Android, uważa, że aktualne zabezpieczenia są wystarczające a prawdopodobieństwo zainstalowania niebezpiecznych aplikacji wyjątkowo niskie [22].

Stan bezpieczeństwa systemu Android jest jeszcze bardziej interesujący w chwili, gdy ma on być instalowany na zupełnie nowych typach urządzeń, takich jak telewizory, zegarki, aparaty fotograficzne a nawet komputery pokładowe w samochodach osobowych.

1.1. Cel pracy

Biorąc pod uwagę rosnące problemy z bezpieczeństwem systemu Android oraz brak adekwatnych narzędzi ochronnych celem niniejszej pracy była analiza dostępnych mechanizmów bezpieczeństwa, ocena zagrożeń oraz wykonanie nowego typu oprogramowania analizującego niebezpieczne aplikacje systemu Android. Innowacyjny pomysł polega na dynamicznej analizie uprawnień aplikacji w porównaniu z uprawnieniami innych aplikacji o podobnych funkcjach w celu automatycznego wykrywania zbyt liberalnego dostępu.

1.2. Zawartość pracy

W pierwszej części pracy opisano zabezpieczenia i zagrożenia w systemie Android. W rozdziale 2 wykonano analizę mechanizmów bezpieczeństwa dostępnych w systemie Android oraz w czasie dystrybucji i instalacji aplikacji. Rozdział 3 zawiera opis zagrożeń – cele ataków specyficzne dla urządzeń mobilnych oraz przykłady możliwych

realizacji ataku. W rozdziale 4 przedstawiono komercyjne i akademickie sposoby na wykrywanie zagrożeń.

Druga część pracy przedstawia implementację i analizę wykonanego programu do analizy zagrożeń. Rozdział 5 opisuje uniwersalną platformę, którą można rozszerzać o nowe typy algorytmów. Zaimplementowane algorytmy wykorzystują dynamiczną analizą uprawnień aplikacji w porównaniu z uprawnieniami innych aplikacji o podobnych funkcjach. W rozdziale 6 wykonano analizę skuteczności działania zaimplementowanego programu oraz przedstawiono możliwe kierunki rozwoju.

Ostatni rozdział zawiera analizę uprawnień i zagrożeń aplikacji z oficjalnego sklepu Google Play oraz nieoficjalnego, chińskiego sklepu GFAN.

2. Zabezpieczenia systemu Android

W niniejszym rozdziale przedstawione zostały zabezpieczenia, które mają chronić użytkowników systemu – mechanizmy wbudowane w jądro systemu operacyjnego Linux, izolowane środowisko uruchomieniowe, w którym działają zainstalowane programy niezależnych dostawców oraz narzędzia wykorzystywane w trakcie tworzenia, dystrybucji i instalacji aplikacji.

W następnym punkcie zostaną opisane najważniejsze zabezpieczenia systemu operacyjnego i środowiska uruchomieniowego. Następnie zostaną opisane procesy mające zapewnić bezpieczeństwo aplikacji dostępnych w oficjalnych kanałach dystrybucji. Kolejna część zawiera opis wad i podatności większości zabezpieczeń.

2.1. Zabezpieczenia Systemu Android.

Android jest otwartym systemem operacyjnym, bazującym na jądrze Linux zoptymalizowanym do działania na urządzeniach mobilnych. Wykorzystywane są wszystkie najważniejsze narzędzia bezpieczeństwa wbudowane w jądro – mechanizm użytkowników i grup, rozdzielna pamięć procesów z mechanizmem randomizacji przestrzeni adresowej chroniącym przed wstrzykiwaniem kodu. Android obsługuje też oznaczanie stron pamięci, jako niewykonywalne (NX-bit), bezpieczną alokację pamięci (OpenBSD calloc) a także stosuje mechanizm przymusowej kontroli dostępu (Security-Enhanced Linux). Jest on bardziej granularny niż tradycyjny mechanizm kontroli dostępu poprzez identyfikator użytkownika i przynależność do grup [4].

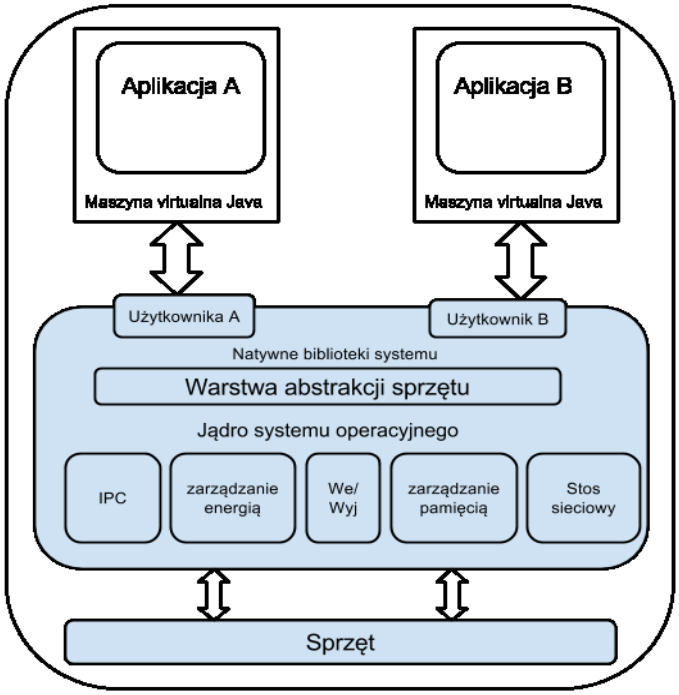
Architektura systemu podzielona jest na warstwy. Funkcje jądra i sterowniki urządzeń udostępniane są platformie aplikacji (application framework) poprzez zestaw natywnych bibliotek. Programy użytkownika korzystają z interfejsu programistycznego udostępnianego przez platformę.. Rysunek 1. Przedstawia środowisko uruchomieniowe aplikacji [6].

Platforma aplikacji implementuje kilka ważnych narzędzi bezpieczeństwa pozwalających na izolację aplikacji, bezpieczną komunikacje między procesami oraz weryfikację uprawnień.

Izolacja aplikacji, tzn. piaskownica (ang. sandboxing): wymuszona jest poprzez uruchamianie każdej aplikacji w oddzielnej instancji maszyny wirtualnej, jako

oddzielny proces systemu operacyjnego przypisany do unikatowego użytkownika z rozłączną, randomizowaną przestrzenią adresową [4].

Komunikacja między aplikacjami może odbywać się jedynie przez mechanizm intencji (ang. intent) – wiadomości zawierających typ przesyłanych danych. Wiadomości są trasowane przez platformę aplikacji z uwzględnieniem uprawnień przydzielonych aplikacji. Aplikacje różnych dostawców (podpisane różnymi certyfikatami) nie mogą współdzielić pamięci, ani korzystać z żadnych innych narzędzi do komunikacji między procesami wbudowanymi w system operacyjny [4].



Rysunek 1. Środowisko uruchomieniowe aplikacji.

Mechanizm uprawnień pozwala na ograniczenie dostępu do najważniejszych funkcji urządzenia, sensorów, oraz danych użytkownika. Aby aplikacja mogą uzyskać dostęp do telefonu, smsów, kontaktów, aparatu czy prawie każdej innej funkcji urządzenia, wymagana jest świadoma zgoda użytkownika. Twórca aplikacji deklaruje w manifeście aplikacji, z jakich uprawnień potrzebuje korzystać. W czasie instalacji użytkownikowi prezentowana jest lista grup uprawnień, jakie mają być nadane nowej aplikacji. W przypadku braku zgody użytkownika instalacja jest przerywana. Uprawnienia, które są zdefiniowane w systemie, jako niegroźnie przydzielane są automatycznie [4].

Kolejnym zabezpieczeniem jest montowanie partycji zawierającej obraz systemu operacyjnego w trybie tylko do odczytu. Ogranicza to możliwość modyfikacji nawet w przypadku, gdy, przez błąd w innym miejscu proces uzyska pełny dostęp do systemu [6].

W wersji 4.2 Systemu Android wprowadzono zupełnie nową usługę zdalnej weryfikacji aplikacji. Polega ona na sprawdzaniu instalowanych aplikacji w usłudze sieciowej dysponującej odpowiednią bazą wirusów. Usługa działa również w przypadku instalacji programów z niezależnych, potencjalnie niebezpiecznych źródeł [4].

W wersji 4.2 dodano również zabezpieczenie przed wysyłaniem smsów na numery premium. W przypadku próby wysłania smsa na numer składający się z kilku cyfr, system zapyta o zgodę użytkownika.[4].

W wersji 5 Systemu Android zostało zaimplementowane opcjonalne szyfrowanie całej pamięci masowej telefonu szyfrem AES. Mechanizm działa na poziomie jądra systemu i wykorzystuje klucz 128 bitowy [4].

Ostatnim ważnym mechanizmem bezpieczeństwa dostępnym w systemie jest możliwość zdalnego usuwania złośliwego oprogramowania. Funkcja ta była dostępna już w pierwszych wersjach systemu i jest wykorzystywana do usuwania niebezpiecznych aplikacji, która znalazła się w oficjalnych kanałach dystrybucji [7].

2.2. Zabezpieczenia w czasie dystrybucji.

Poza mechanizmami wbudowanymi w system operacyjny istnieją dwa narzędzia istotne ze względów bezpieczeństwa, które są wykorzystywane w czasie dystrybucji oprogramowania – podpisywanie cyfrowe aplikacji przez autora oraz automatyczna analiza aplikacji uruchamianej w emulatorze Android Bouncer.

Cyfrowe podpisywanie certyfikatem autora aplikacji pozwala na stwierdzenie autentyczności. Klucz publiczny certyfikatu dołączany jest do archiwum. W trakcie instalacji menadżer pakietów systemu Android weryfikuje poprawność podpisu. Nie ma możliwości instalowania aplikacji niepodpisanych. Dodatkowo, aplikacje podpisane tym samym certyfikatem mogą być uruchamiane w jednym procesie i mają wzajemny dostęp do zasobów.

Zanim podpisana aplikacja będzie dostępna w oficjalnym sklepie, następuje jej automatyczna analiza w narzędziu Android Bouncer. Jest to wirtualne środowisko, w

którym zostaje uruchamiana każda aplikacja zgłoszona do sprzedaży. Ma to symulować uruchomienie na prawdziwym urządzeniu, w trakcie, którego zachowanie aplikacji jest porównywane ze znanymi wcześniej wirusami [8][10][13].

2.3. Wady zabezpieczeń.

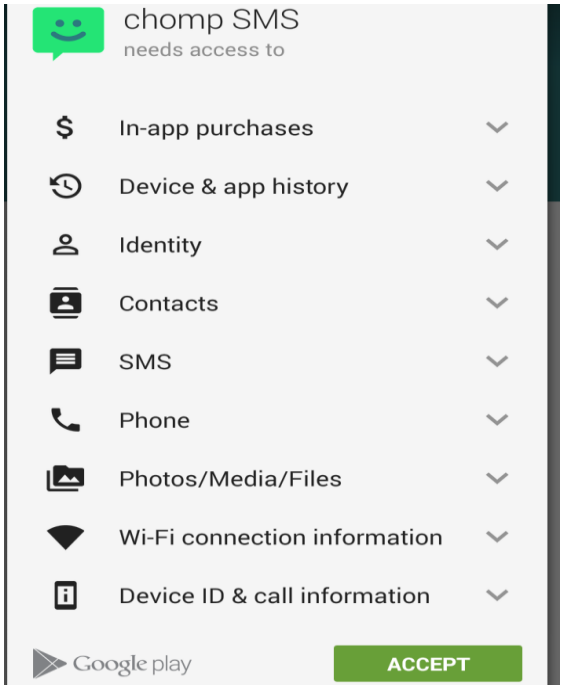
W tym punkcie zostaną opisane znane podatności i wady niektórych mechanizmów opisanych wcześniej.

Najwięcej zastrzeżeń budzi system uprawnień kontrolowanych przez platformę aplikacji – zawiera on kilka wad, które przyczyniają się do uśpienia czujności użytkownika.

W czasie instalowania nowej aplikacji system przydziela jej automatycznie wszystkie niegroźne uprawnienia. Pozostałe wymagają jednorazowej zgody użytkownika. W tym celu prezentowane są grupy, do których należą uprawnienia wykorzystywane przez aplikację. Akceptacja uprawnień odbywa się zgodnie z zasadą wszystko albo nic – użytkownik zgadza się na wykorzystywanie wszystkich uprawnień z przedstawionych grup albo instalacja zostaje przerwana. Nie ma możliwości wyłączenia niektórych grup ani odroczenia decyzji do momentu użycia uprawnienia. Akceptacja grupy uprawnień oznacza akceptację każdego z uprawnień w grupie, co jest naruszeniem zasady minimalnego uprzywilejowania [6]. Ponadto sposób prezentacji przesłania informacje szczegółowe, takie jak to, że aplikacja może narazić użytkownika na koszty, co jest widoczne na rysunku 2. Dodatkowym problemem są nieświadomi i nierozważni użytkownicy, którzy często nie zwracają uwagi na podejrzane kombinacje uprawnień [10].

Wady systemu uprawnień wynikają w dużej mierze z liberalnego podejścia do personalizacji i modyfikacji Androida. Użytkownik może wymienić prawie każdą domyślną aplikację graficznej powłoki systemu, również te najbardziej newralgiczne – skrzynkę sms, książkę adresową, klawiaturę wirtualną a nawet aplikację służącą do wykonywania połączeń telefonicznych. Oznacza to, że biblioteki dostępne dla twórców zwykłych aplikacji umożliwiają wysyłanie smsów, wykonywanie połączeń, uruchamianie innych aplikacji itd. Duża liczba potencjalnie niebezpiecznych interfejsów programistycznych przyczynia się do dużej liczby uprawnień, co powoduje potrzebę jednokrotnej akceptacji wszystkich uprawnień w czasie instalacji – w przeciwnym wypadku działanie programów byłoby często przerywane oczekiwaniem

na zgodę użytkownika. Jednorazowe akceptacja wielu uprawnień powoduje brak czytelności dla użytkownika, który na relatywnie małym ekranie telefonu musi przejrzeć wiele grup uprawnień, żeby móc świadomie zdecydować o instalacji aplikacji.



Rysunek 2. Akceptacja uprawnień aplikacji.

Weryfikator aplikacji wbudowany w system operacyjny również posiada wiele wad. Przede wszystkim jego skuteczność jest bardzo niska, według [15] wynosi zaledwie 15%, a badanie było przeprowadzone na dostępnych publicznie próbkach. Stanowi to wynik znacznie gorszy od rozwiązań firm trzecich [20]. Należy zaznaczyć jednak, że badanie miało miejsce zaledwie kilka miesięcy po wprowadzeniu automatycznej weryfikacji.

Druga, znacznie poważniejsza wada weryfikatora, to sposób jego działania. Analiza odbywa się jedynie w czasie instalacji aplikacji, a identyfikacja złośliwego oprogramowania wykorzystuje jedynie sygnatury (nazwy pakietów, sumy kontrolne) w celu znalezienia informacji w bazie znanych wirusów. Można to łatwo obejść, poprzez randomizację kodu źródłowego lub kodu bajtowego aplikacji. Brak jest ochrony w czasie rzeczywistym, jak to ma miejsce w przypadku nowoczesnych antywirusów stosowanych na komputerach osobistych.

Kolejny poważny problem dotyczy podpisywania oprogramowania. Menadżer pakietów systemu Android nie weryfikuje ścieżki certyfikacji, oznacza to, że można

zainstalować aplikacje podpisane przez samozaufane certyfikaty. Jest to bardzo groźne w połączeniu z możliwością dekompilacji, modyfikacji i ponownej kompilacji cudzej aplikacji. Narzędzia do przeprowadzenia takiego ataku są dostępne [16] [17] [18], wynikiem będzie podrobiona aplikacja, wyglądem nieróżniąca się oryginału, która zostanie zaakceptowana w systemie Android nawet, jeżeli została podpisana innym certyfikatem niż oryginał.

Nie mniej istotne problemy obecne są w procesie akceptacji aplikacji w oficjalnym sklepie. Emulator Android Bouncer analizujący działanie aplikacji, uruchamia ją zaledwie na kilka minut. Wystarczy więc, że aplikacja nie będzie wzbudzać podejrzeń chwile po pierwszym uruchomieniu [8]. Ponad to istnie wiele metod wykrycia działania w środowisku emulowanym czy wirtualizowanym[13].

Niektóre wady analizatora mogłyby być pomijalne gdyby w procesie weryfikacji każdej aplikacji uczestniczył dodatkowo człowiek weryfikujący aplikację na prawdziwym urządzeniu. Wprowadzenie czynnika ludzkiego nie jest niemożliwe pomimo olbrzymiej ilości nowych aplikacji [9]. Przykładem może być platforma iOS i jej sklep App Store gdzie każda aplikacja, oprócz weryfikacji automatycznych, sprawdzana jest przez człowieka. Pozwala to również na odrzucenie mało wartościowych aplikacji, które w przypadku sklepu Google Play stanowią aż 15% [9].

Kolejnym źródłem zagrożeń jest możliwość instalacji aplikacji pochodzących z nieoficjalnych kanałów dystrybucji takich jak niezależne sklepy czy nawet sieci P2P. W takim przypadku nie ma żadnej gwarancji jakości czy bezpieczeństwa. Jest też prawdopodobne, że aplikacja została zmodyfikowana bez wiedzy autora.

Ostatnie zagrożenie to niewystarczająca częstotliwość aktualizacji. Ponad 1/3 urządzeń korzysta ze starych wersji systemu (wcześniejszych niż 4.2), gdzie nie ma dużej części zabezpieczeń. Co więcej, użytkownicy nie otrzymują nawet ważnych poprawek bezpieczeństwa. Jest to związane z modelem, w jakim tworzony jest i instalowany system Android. Dokumentacja systemu informuje, że za aktualizacje bezpieczeństwa odpowiedzialni są producenci urządzeń, ci jednak nie mają żadnego interesu w aktualizowaniu starych urządzeń.

2.4. Podsumowanie

Przedstawiono mechanizmy bezpieczeństwa systemu Android oraz narzędzia procesu dystrybucji mające zapewnić bezpieczeństwo użytkowników. Następnie opisane

zostały istotne wady zabezpieczeń pozwalające na szereg ataków, łatwe pominięcie większości zabezpieczeń lub wykorzystanie nieuwagi użytkowników.

Negatywny obraz bezpieczeństwa systemu operacyjnego Android, jaki wyłania się z niniejszych analiz pokazuje, dlaczego jest on tak popularny wśród autorów złośliwego oprogramowania. Przy tak dużej liczbie użytkowników starannie przygotowana złośliwa aplikacja będzie zawsze skuteczna z punktu widzenia jej twórcy – zainfekuje wiele urządzeń. Jednak z punktu widzenia osoby rozsądnie korzystającej z systemu Android ryzyko utraty pieniędzy czy prywatnych danych jest niewysokie [6], wystarczy zachować podstawowe środki bezpieczeństwa.

3. Zagrożenia bezpieczeństwa systemu Android

Urządzenia mobilne stanowią bardzo atrakcyjny cel ataków. Użytkownicy przechowują na nich bardzo wiele informacji osobistych i korporacyjnych, łatwo jest spieniężyć przeprowadzony atak, a same urządzenia mają skonfigurowane dostępy do chronionych zasobów, sieci prywatnych itp.

Jak pokazano w poprzednim rozdziale, zabezpieczenia tych urządzeń są niewystarczające, ponadto trudno jest egzekwować korporacyjne polityki bezpieczeństwa czy chronić dostęp sieciowy z zewnątrz. Urządzenia są zawsze połączone z publiczną siecią komórkową i często z publiczną siecią Internet. Brakuje też odpowiedniej świadomości i wiedzy na temat bezpieczeństwa wśród użytkowników.

W tym rozdziale zostaną przedstawione zagrożenia bezpieczeństwa urządzeń z systemem Android. W pierwszej kolejności omówione zostaną cele możliwych ataków, a następnie sposoby ich przeprowadzenia.

3.1. Cele ataków

Najłatwiejszym do spieniężenia atakiem jest uzyskanie dostępu do funkcji telefonicznych urządzenia i wysłanie SMS/MMS premium lub wykonanie połączenia na numery dodatkowo płatne. Taki atak nie jest trudny do przeprowadzenia i gwarantuje szybki zarobek [11][14].

Kolejnym celem ataku może być kradzież danych prywatnych, w tym danych osobowych użytkownika, jego znajomych, kontaktów itp. Takie dane mają realną wartość na czarnym rynku. Dodatkowo, po uzyskaniu pełnego dostępu (root) możliwe jest wykradanie haseł do serwisów, z których użytkownik korzysta na urządzeniu, uzyskanie danych korporacyjnych czy innych informacji, które są przechowywane na urządzeniu. Zainfekowany telefon może też posłużyć do uzyskania dostępu do sieci lokalnych lub korporacyjnych wirtualnych sieci prywatnych (VPN), gdy dostęp do nich został skonfigurowany przez użytkownika[6][10].

Trzeci typ ataku to programy szpiegujące (ang. Spyware), które zbierają informacje na temat użytkownika bez jego wiedzy. W przypadku telefonów komórkowych takie oprogramowanie jest wyjątkowo skuteczne ze względu na liczbę czujników i informacji dostępnych na telefonie. Możliwe jest śledzenie lokalizacji,

komunikacji sms, listy wykonywanych połączeń, czasu i sposobu używania telefonu a nawet temperatury i ciśnienia powietrza w otoczeniu użytkownika [10].

Bardziej zaawansowane ataki mogą służyć do podsłuchiwania komunikacji głosowej użytkownika. W przypadku systemu Android, standardowe aplikacji nie mogą przechwytywać dźwięku z mikrofonu czy głośnika w czasie połączeń, ale jedyną barierę stanowi mechanizm uprawnień aplikacji [10][14].

Rozpowszechnionym typem ataku jest prezentowanie użytkownikowi bardzo dużej liczby reklam (ang. Adware). Nie stanowi to bezpośredniego zagrożenia dla danych i nie naraża na koszty, może jednak utrudniać korzystanie z urządzenia. Takie oprogramowani jest powszechnie uważane za złośliwe [6].

Kolejny typ ataku to zainstalowanie mechanizmów pozwalających na zdalne przesyłanie poleceń do aplikacji. Sieć tak zainfekowanych urządzeń (ang. botnet) może zostać wykorzystana w celu wysyłania spamu, blokady jakiegoś serwisu (ang. Distributed Denial of Service) czy nawet „wydobywania" kryptowalut [6][11][23].

Gdy złośliwa aplikacja uzyska pełny dostęp do urządzenia (root) spektrum możliwych ataków jest bardzo szerokie. Możliwe jest wykorzystanie takiego telefonu do blokowania działania sieci komórkowej lub lokalnej sieci bezprzewodowej [10], uniemożliwienie użytkownikowi dostępu do podstawowych funkcji urządzenia, czy wymuszenie opłaty za odblokowanie funkcji (ang. ransomware) [6].

Często złośliwe oprogramowanie na telefony komórkowe wykorzystywane jest w połączeniu z zainfekowanymi komputerami osobistymi. Możliwy jest wtedy atak na usługi wykorzystujące dwustopniową weryfikację, takie jak np. bankowość elektroniczna, w której użytkownik potwierdza każdą operację za pomocą kodu z wiadomości sms. Znane jest też oprogramowanie atakujące komputery osobiste, które jest przenoszone za pomocą pamięci masowej telefonu. Oprogramowanie przygotowuje plik wykonywalny, który zostaje uruchomiony, jako domyślna akcja po zamontowaniu pamięci masowej w komputerze.

3.1. Sposoby ataków

W tym punkcie zostaną pokazane znane sposoby przeprowadzania ataków na urządzenia mobilne. Większość z przedstawionych poniżej sposobów jest specyficzna dla systemu Android lub dla urządzeń mobilnych. Każdy opisany atak może być wykorzystany w celach opisanych w poprzednim punkcie.

Najłatwiejszy sposób ataku to ukrycie złośliwych funkcji w pozornie pożytecznej aplikacji, jest to typowy przykład złośliwego oprogramowania typu koń trojański. Jedyna różnica to potrzeba zadeklarowania większej liczby uprawnień, zależnie od funkcji, do których ma mieć dostęp złośliwa część kodu aplikacji. Taki atak, wykonany z użyciem jedynie oficjalnych interfejsów programistycznych, może być wyjątkowo skutecznie osiągnąć większość celów opisanych w poprzednim punkcie [11][14][23].

Bardziej zaawansowane ataki mogą wykorzystywać podatności platformy Android i jądra systemu operacyjnego w celu przejęcia pełnej kontroli nad urządzeniem (ang. root exploit). Taki atak również może być przeprowadzony z użyciem aplikacji konia trojańskiego, która nawet nie musi mieć wielu dodatkowych uprawnień – wystarczy dostęp do obszaru znanej podatności, która ma zostać wykorzystana. Po uzyskaniu dostępu uprzywilejowanego (root) aplikacja ma pełny dostęp do wszystkich funkcji i zasobów urządzenia, w tym uprawnień systemowych [11].

Niektórzy użytkownicy systemu Android chcąc mieć dostęp do większej liczby funkcji i możliwości sami instalują aplikacje, które odblokowują dostęp do użytkownika root. Programy te wykorzystują znane podatności w celu odblokowania poleceń „su" lub „sudo", pozwalają też na modyfikacje programu uruchomieniowego (ang. boot loader). Nieoficjalne dystrybucje sytemu Android również pozwalają na uruchamianie programów, jako użytkownik root. Zwykle, w takich systemach, dostęp uprzywilejowany jest kontrolowany przez użytkownika za pomocą dedykowanej aplikacji. Stanowi to poważne zagrożenie, jeżeli aplikacja, której nadano permanentny dostęp okazałaby się złośliwa.

Popularnym typem ataku jest przepakowywanie aplikacji [6][12]. Polega to na dekompilacji znanej aplikacji, modyfikacji poprzez dodanie złośliwego kodu i ponownej kompilacji. Z punktu widzenia użytkownika aplikacja nie różni się od oryginału. Jeżeli została prawidłowo podpisana przy ponownej kompilacji to również system Android nie zgłosi zastrzeżeń. Różnice będą polegać na wartościach sum kontrolnych i ewentualnie na liście uprawnień, które potrzebuje aplikacja. Najczęściej w ten sposób modyfikowane są popularne płatne aplikacje instalowane przez użytkowników unikający opłat. Narzędzia umożliwiające taki atak są łatwo dostępne [16][18] ponieważ są wykorzystywane w celu analizy złośliwych aplikacji.

Kolejny wyrafinowany atak polega na zachęceniu użytkownika do zainstalowania kilku niepozornych aplikacji, z których każda ma adekwatne uprawnienia systemowe. Atak wykorzystuje mechanizm komunikacji międzyprocesowej dostępny dla aplikacji podpisanych tym samym certyfikatem. W [11] podano przykład trzech współpracujących aplikacji. Pierwsza, służąca do usuwania zduplikowanych kontaktów ma pełny dostęp do książki adresowej, druga – autoresponder sms, może wysyłać smsy, trzecia ma dostęp do Internetu. Takie aplikacje mogą zostać wykorzystane np. do rozsyłania spamu sms do wszystkich kontaktów użytkownika. Treść takich smsów może być dynamicznie pobierana przez jedną z aplikacji.

Następny sposób ataku to zachęcenie użytkownika do zainstalowania zupełnie niegroźniej aplikacji, która po pewnym czasie informuje o możliwości zainstalowania aktualizacji. Użytkownik pobiera aktualizacje spoza oficjalnego kanału dystrybucji i instaluje ją bez ponownej analizy uprawnień. Dopiero zainstalowana aktualizacja jest złośliwym oprogramowaniem [12].

Podobna metoda ataku opiera się na intensywnym zachęcaniu użytkownika do ściągnięcia aplikacji poprzez wyświetlanie reklam, które mogą być wyświetlane w pożytecznych aplikacjach poprzez banery popularnych systemów reklamowych. Po kliknięciu w baner użytkownik przenoszony jest do przeglądarki, która pobiera złośliwą aplikację. Następnie jest ona instalowana i uruchamiana przez użytkownika, który sądzi, że ma do czynienia z reklamowanym oprogramowaniem [12].

Inny typ ataku z wykorzystaniem pozornie niegroźnej aplikacji wykorzystującej niewiele uprawnień polega na pobraniu złośliwego kodu w czasie działania [20]. Pobrane fragmenty są kodem natywnym jednostki obliczeniowej na urządzeniu. System Android pozwala na uruchomienie takiego kodu ze względów wydajnościowych. Ponieważ nie jest to bajtkod maszyny wirtualnej java to zostanie on uruchomiony bez kontroli uprawnień. Według [23] jest to jedno z największych zagrożeń w systemie Android. Przed wdrożeniem rozszerzenia Security-Enhanced Linux do jądra systemu Android wszystkie znane metody uzyskania dostępu użytkownika root (ang. root exploits) wykorzystywały mechanizm wykonywania kodu natywnego. Pobranie takiego kodu w czasie działania dodatkowo utrudnia wykrycie zagrożenia.

System Android zawiera wiele popularnych bibliotek, z których każda może zawierać podatności. Przykładem może być silnik przeglądarki internetowej WebKit, który jest bardzo popularny również na komputerach osobistych. Znając podatności takiej biblioteki można spreparować odpowiednią stronę internetową w wykorzystania dziury bezpieczeństwa [10][42]. Przeprowadzenie takiego ataku nie wymaga instalowania oprogramowania przez użytkownika. Wystarczy zachęcić do wejścia na odpowiednią stronę poprzez atrakcyjną reklamę. Ataki na popularne biblioteki zawarte w systemie Android mogą być bardzo skuteczne, ponieważ niewiele urządzeń jest w porę aktualizowana przez producentów.

Ostatni typ ataków na urządzenia mobilne to podszywanie się pod sieć komórkową lub lokalną bezprzewodową sieć lokalną (ang. spoofing) [14]. Umożliwiają to podsłuchiwanie komunikacji i może wykorzystywane w precyzyjnych atakach np. w celu pozyskania tajnych informacji gospodarczych. Podatność na takie ataki nie wynika z wad systemu Android a jest spowodowana niedoskonałymi protokołami komunikacji i dotyczy każdego urządzenia korzystającego z tych protokołów.

3.2. Podsumowanie

Przedstawione powyżej cele i sposoby ataku nie wyczerpują możliwych zagrożeń. Urządzenia mobilne są podatne na większość typów zagrożeń znanych z komputerów osobistych, a przez dużą liczbę wartościowych informacji, niewystarczające i nieaktualizowane zabezpieczenia są wyjątkowo atrakcyjnym celem.

Kilka z pokazanych ataków można przeprowadzić za pomocą oficjalnych kanałów dystrybucji aplikacji, jeżeli tylko autor złośliwego oprogramowania zna automatyczne mechanizmy analizujące niebezpieczne aplikacje. Dostęp do wielu atrakcyjnych zasobów i informacji jest chroniony przez system uprawnień i tylko uważny użytkownik będzie w stanie wyłapać zagrożenie. Część z ataków nie wymaga początkowego instalowania oprogramowania, wystarczy spreparowana strona lub dobra reklama. System Android nie pozwala na automatyczne uruchomienie nowo zainstalowanej aplikacji, ale często wystarczy jedno uruchomienie przez oszukanego użytkownika, aby zrealizować atak np. przejąć kontrolę z wykorzystaniem znanej podatności, wysłać sms czy przeczytać całą książkę adresową. Sytuacje pogarsza brak odpowiednich aktualizacji, które powinny być udostępniane przez producentów urządzeń.

4. Wykrywanie zagrożeń

W tym rozdziale omówione zostaną sposoby wykrywania niebezpiecznych aplikacji. W pierwszej części pokazane zostanie, w jaki sposób i z jaką skutecznością działają komercyjne programy antywirusowe firm zewnętrznych. W drugiej części rozdziału zostaną opisane akademickie przykłady nowych sposobów na wykrywanie złośliwych aplikacji dla systemu Android.

4.1. Komercyjne programy antywirusowe

W sklepie Google Play oferowane jest wiele programów „antywirusowych" produkowanych przez firmy znane z branży bezpieczeństwa komputerowego. Aplikacje te mają, według producentów, zapewniać ochronę w czasie rzeczywistym tak jak ich odpowiedniki na komputery osobiste. Ze względu na architekturę systemu Android programy antywirusowe posiadają kilka istotnych wad.

Android nie zapewnia żadnego interfejsu programistycznego dla aplikacji antywirusowych, nie pozwala też na działanie aplikacji z uprawnieniami administracyjnymi czy root. Oznacza to, że aplikacje antywirusowe uruchamiane są z ograniczonymi uprawnieniami dostępnymi dla aplikacji zewnętrznych. Nie mogą one wykonać analizy pamięci innych aplikacji, nie potrafią analizować wywołań systemowych, ruchu sieciowego ani listy otwartych plików przez inne aplikacje. Nie ma możliwości zastosowania nowoczesnych algorytmów bazujących na heurystycznej analizie zachowań [6].

Ochrona w czasie rzeczywistym jest zapewniana poprzez uruchamianie usługi w tle, jednak w przypadku braku zasobów usługa taka może zostać zatrzymana. Ponadto inna aplikacja zewnętrzna z odpowiednimi uprawnieniami może zatrzymać usługę programu antywirusowego. Ostatni problem to brak możliwości zamrożenia lub usunięcia niebezpiecznych aplikacji, jeżeli takie zostaną już wykryte, aplikacja antywirusowa może wtedy jedynie zainicjować usunięcie przez użytkownika [6].

Pomimo tych oczywistych braków, aplikacje antywirusowe potrafią wykryć część zagrożeń. Ich działanie polega na analizie sygnatur, czyli unikatowych identyfikatorów aplikacji lub jej fragmentów. Sygnatury są budowane za pomocą informacji dostarczanych przez interfejsy programistyczne systemu Android. Sygnatury porównywane są z bazą znanych zagrożeń.

4.1.1. Skuteczność komercyjnych programów antywirusowych

W opracowaniu [20] dokonano obszernej analizy skuteczności popularnych programów antywirusowych dla systemu Android. W tym punkcie przytoczono kilka najistotniejszych wyników. Test skuteczności zawierał sześć testów i sprawdzał działanie jedenastu bezpłatnych aplikacji antywirusowych.

Pierwszy test polegał na sprawdzeniu wykrywalności dziesięciu znanych wcześniej próbek złośliwego oprogramowania. Tabela 1 zawiera wyniki pierwszego testu. Kolejne kolumny oznaczają kolejne próbki złośliwego oprogramowania (1-10), zielony kolor tła oznacza prawidłowe wykrycie zagrożenia, czerwony kolor tła oznacza brak wykrycia.

	1	2	3	4	5	6	7	8	9	10	Suma
avast											10/10
AVG											10/10
BitDefender											10/10
ESET											10/10
F-Secure											10/10
Kaspersky											9/10
Lookout											10/10
McAfee											10/10
Norton											10/10
Sophos											10/10
Trend Micro											9/10

Tabela 1 Wykrywanie znanych aplikacji złośliwych

Wykrywanie znanych, niezmodyfikowanych aplikacji złośliwych jest bardzo wysokie, biorąc pod uwagę ograniczony dostęp do informacji, z których budowane są sygnatury.

W drugim teście zmieniono nieznacznie próbki złośliwego oprogramowania poprze dekompilacje i ponowną kompilacje z wykorzystaniem innego certyfikatu do podpisu. Działanie aplikacji nie zostało zmodyfikowane, jedynie wartości sum kontrolnych. Wyniki drugiego testu znajdują się w tabeli 2 i pokazują istotny spadek

wykrywalności już przy niewielkich zmianach. Jedynie program ESET był w stanie wykryć znaczną większość zagrożeń.

	1	2	3	4	5	6	7	8	9	10	Suma
avast											6/10
AVG											4/10
BitDefender											4/10
ESET											9/10
F-Secure											6/10
Kaspersky											3/10
Lookout											7/10
McAfee											2/10
Norton											3/10
Sophos											0/10
Trend Micro											3/10

Tabela 2 Wykrywanie zmodyfikowanych aplikacji złośliwych

Kolejny test polegał na sprawdzeniu wykrywalności znanych aplikacji wykorzystujący podatności do uzyskania dostępu uprzywilejowanego (ang. root exploit). Wyniki testu trzeciego zawiera tabela 3. Wykrywalność tego rodzaju zagrożeń była dość wysoka, sześć z jedenastu testowanych programów wykryła poprawnie wszystkie trzy zagrożenia.

Czwarty test polegał na sprawdzeniu wykrywalności zmodyfikowanych aplikacji z testu 3. Tak jak wcześniej wykonano dekompilacje i ponowną kompilacje, zmieniając jedynie wartości sum kontrolnych. Wyniki testu przedstawione są w tabeli 4. Wykrywalność jest jeszcze niższa niż w przypadku zmodyfikowanych aplikacji złośliwych, dziewięć z jedenastu testowanych programów nie wykryła żadnego zagrożenia. Najlepszy wynik osiągnął program Lookout, który wykrył dwie z trzech próbek.

Piąty test sprawdzał wykrywalność pobrania i uruchomienia złośliwego kodu natywnego. Wyniki zawarto w tabeli 5. Żaden z analizowanych programów nie poradził sobie z wykryciem takiego ataku. Wykrywalność była równa zero.

	1	2	3	Suma
avast				3/3
AVG				0/3
BitDefender				0/3
ESET				3/3
F-Secure				1/3
Kaspersky				3/3
Lookout				3/3
McAfee				3/3
Norton				0/3
Sophos				0/3
Trend Micro				3/3

Tabela 3. Wykrywanie znanych aplikacji uzyskujących dostęp uprzywilejowany.

	1	2	3	Suma
avast				1/3
AVG				0/3
BitDefender				0/3
ESET				0/3
F-Secure				0/3
Kaspersky				0/3
Lookout				2/3
McAfee				0/3
Norton				0/3
Sophos				0/3
Trend Micro				0/3

Tabela 4. Wykrywanie zmodyfikowanych aplikacji uzyskujących dostęp uprzywilejowany.

W ostatnim teście sprawdzono wykrywanie nieznanych wcześniej zagrożeń. Pochodziły one z prac badawczych instytutu Fraunhofera, nie były dostępne publicznie, więc nie mogły znajdować się w bazach zagrożeń programów antywirusowych. Wyniki przedstawia tabela 6. Żadnej badanej aplikacji nie udało się wykryć złośliwego działania nieznanych wcześniej próbek.

	1	Suma
avast		0/3
AVG		0/3
BitDefender		0/3
ESET		0/3
F-Secure		0/3
Kaspersky		0/3
Lookout		0/3
McAfee		0/3
Norton		0/3
Sophos		0/3
Trend Micro		0/3

Tabela 5. Wykrywanie pobrania i wykonania złośliwego kodu natywnego.

	1	2	3	Suma
avast				1/3
AVG				0/3
BitDefender				0/3
ESET				0/3
F-Secure				0/3
Kaspersky				0/3
Lookout				2/3
McAfee				0/3
Norton				0/3
Sophos				0/3
Trend Micro				0/3

Tabela 6. Wykrywanie nieznanych aplikacji złośliwych.

Wyniki testów przeprowadzonych w [20] pokazują, że zewnętrzne programy antywirusowe są skuteczne jedynie przeciwko niezmodyfikowanym aplikacjom złośliwym. Autorzy nie stosowali zaawansowanych technik zaciemniania kodu. Wyniki potwierdzają opisane wcześniej ograniczenia.

Starsze badanie [12] uzyskało skuteczność zewnętrznych aplikacji antywirusowych na poziomie 20-80% dla znanych zagrożeń. W badaniu nie wykorzystano różne wersje kilku złośliwych aplikacji. Nie wykonano modyfikacji próbek, ale część z nich była na tyle nowa, że żaden z testowanych programów nie miał ich jeszcze w bazie sygnatur.

W rozdziale nie uwzględniono usług analizy zagrożenia dostępnych w sieci Internet, które dokonują analizy po przesłaniu pliku apk z binarną wersją aplikacji [6]. Usługi takie mogą uruchamiać aplikacje w emulowanym środowisku gdzie możliwa jest heurystyczna analiza zachowań. Niestety nie jest możliwa ochrona urządzenia użytkownika w czasie rzeczywistym – analiza wymagałaby przesłania całego pliku z aplikacją, które rozmiar często przekracza kilka megabajtów, to natomiast wymaga czasu i użytkownik mógłby już uruchomić aplikację. Ponad to istnieją sposoby wykrycia środowiska emulowanego[13].

4.2. Akademickie algorytmy wykrywania

W tym punkcie przedstawiono kilka akademickich propozycji nowych sposobów na wykrywanie zagrożeń. Każde z opisanych rozwiązań można zaimplementować, jako programy antywirusowe w aktualnej wersji systemu Android. Niektóre z nich, korzystające ze statycznej analizy kodu, wymagają programowej dekompilacji plików apk, co jest możliwe do zaimplementowania. Dostępne są aplikacje android wykonujące programową dekompilacje [44][45][46].

Statyczna analiza uprawnień aplikacji, w połączeniu z algorytmami uczenia maszynowego została opisana w [21]. Jako wejściowego wektora cech użyto binarnych wartości reprezentujących wykorzystanie kolejnych uprawnień przez analizowaną aplikacje, a do klasyfikacji użyto trzech algorytmów: drzewa decyzyjnego J48, drzewa regresyjnego CART (ang. Classification and Regression Trees) oraz algorytmu las losowy (ang random forest). Klasyfikację wykonano dla dwóch zbiorów danych, uzyskując ponad 85% poprawnych klasyfikacji zagrożeń (ang. true positive rate, TPR) przy błędnej klasyfikacji próbek niegroźnych na poziomie poniżej 16% (ang. false positive rate FPR). Najbardziej skuteczny okazał się algorytm klasyfikacji las losowy, z wynikiem powyżej 91% TPR i poniżej 9% FPR.

W opracowaniu [43] zrezygnowano z uczenia maszynowego na rzeczy analizy arbitralnie przyjętych kombinacji uprawnień wykorzystywanych przez badane

aplikacji. Przyjęte kombinacje opracowano na podstawie analizy uprawnień deklarowanych przez znane aplikacje złośliwe i niezłośliwe. Wyniki przedstawiono w tabeli 7 poniżej.

Rozmiar kombinacji	Prawdziwie dodatnie (true positive)	Prawdziwie ujemne (true negative)
2	95.82	44.11
3	94.49	55.36
4	94.39	61.35
5	90.51	61.10
6	83.57	87.53

Tabela 7. Wyniki klasyfikacji na podstawie kombinacji uprawnień.

Dla małej liczby uprawnień wchodzących w skład kombinacji bardzo dużo niegroźnych aplikacji zostaje sklasyfikowana, jako zagrożenie. Dopiero dla kombinacji zawierającej 6 uprawnień liczba poprawnie sklasyfikowanych aplikacji niegroźnych wynosi ponad 87%, przy 83% poprawnie sklasyfikowanych aplikacji niebezpiecznych.

W artykule [25] autorzy wykorzystali informacje ze sklepu Google Play (wówczas Android Market), takie jak cena aplikacji, oceny i recenzje użytkowników czy nawet opis aplikacji. Wykorzystano również uprawnienia deklarowane w pliku manifestu. Do klasyfikacji użyto algorytmu centroidów (ang. k-means clustering). Uzyskano wynik na poziomie 71% poprawnie sklasyfikowanych zagrożeń oraz 71% poprawnie sklasyfikowanych aplikacji niegroźnych.

W opracowaniu [26] autorzy użyli sieci neuronowej, wektorem cech był 128 wymiarowy wektor wartości binarnych oznaczających użycie uprawnień. Uzyskano wyniki na poziomie 65% poprawnie sklasyfikowanych zagrożeń oraz 65% poprawnie sklasyfikowanych aplikacji niegroźnych.

W pracach [24] oraz [27] zastosowano metody uczenia maszynowego oraz wraz ze statyczną analizą wywołań interfejsów programistycznych (ang. API calls tracing). W obu przypadkach uzyskano dokładność powyżej 90%. Dokładność zdefiniowana jest, jako iloraz liczby poprawnie sklasyfikowanych próbek do całkowitej liczby analizowanych próbek.

Ostatni przykład [28] to system analizy statycznej kodu, który tworzy szczegółowy raport na temat działania aplikacji i możliwych zagrożeń. Pozwala to

użytkownikowi na dużo lepsze zrozumienie wykorzystywanych uprawnień i podjęcie świadomej decyzji o instalowaniu aplikacji.

4.3. Podsumowanie

Komercyjne programy antywirusowe posiadają poważne wady wynikające z architektury systemu Android. Potrafią wykrywać jedynie znane wcześniej, niezmodyfikowane zagrożenia a ich działanie może być zakłócone przez inne aplikacje zewnętrzne.

Akademickie programy do analizy zagrożeń mogą być znacznie bardziej skuteczne w walce z nieznanymi zagrożeniami, ale znacznie częściej interpretują łagodne aplikacje, jako zagrożenie.

W kolejnym punkcie opisano nowy pomysł na analizę zagrożeń – wykorzystanie informacji na temat aplikacji podobnych do analizowanej. Nigdzie w literaturze takie podejście nie zostało zbadane.

5. Implementacja nowego algorytmu wykrywania zagrożeń

Analiza działania komercyjnych i akademickich algorytmów wykrywania niebezpiecznych aplikacji pokazuje ich wysoką skuteczność w przypadku znanych sygnatur złośliwego oprogramowani lub dużego podobieństwa do znanych zagrożeń. Algorytmy są znacznie mniej skuteczne w przypadku nowych zagrożeń. Ponadto nie analizują pakietów pod kontem możliwej dekompilacji i modyfikacji znanych aplikacji. Oba te problemy ma rozwiązać nowa klasa algorytmów opracowanych w trakcie realizacji niniejszej pracy.

W kolejnym punkcie znajduje się opis proponowanego algorytmu i możliwe jego implementacje. Następnie opisano wykonaną implementację platformy do analizy zagrożeń oraz 5 niezależnym komponentów platformy odpowiedzialnych za ocenę zagrożenia. Opis skuteczności zaimplementowanych narzędzi znajduje się w następnym rozdziale.

5.1. Propozycja nowej klasy algorytmów

Statyczna analiza pliku manifestu pozwala na uzyskanie wielu informacji dotyczących działania aplikacji, co umożliwia wykrywanie niebezpiecznych aplikacji. Akademickie przykłady statycznej analizy manifestu zostały zaprezentowane w poprzednim rozdziale. Najważniejsze Informacje zawarte w pliku manifestu to:

- opis komponentów aplikacji
- opis bibliotek wykorzystywanych przez aplikację
- lista wykorzystywanych funkcji urządzenia i systemu operacyjnego
- opis danych udostępnianych innym aplikacjom
- opis zdarzeń, których nasłuchuje aplikacja
- lista uprawnień, z których korzysta aplikacja.

Proponowana aplikacja będzie rozszerzać statyczną analizę pliku AndroidManifest.xml o porównanie z metadanymi aplikacji o podobnych funkcjach lub aplikacji z tej samej kategorii. Pozwoli to nie tylko na wykrycie dostępu do newralgicznych uprawnień, ale także na stwierdzenie czy nadane uprawnienia są adekwatne do funkcji aplikacji. Umożliwiłoby to na wykrycie takich zagrożeń jak

szeroko opisywana latarka kolekcjonująca wrażliwe dane prywatne [31] czy modyfikowane popularne płatne aplikacje dostępne poza oficjalnym sklepem.

Podstawowy sposób działania proponowanego algorytmu zakłada analizę pliku AndroidManifest.xml badanej aplikacji oraz wyszukanie podobnych aplikacji dostępnych w sklepie Google Play. Kolejny krok to pobranie uprawnień wszystkich aplikacji podobnych w celu wykrycia zbyt szerokich uprawnień nadanych aplikacji analizowanej.

Opisany algorytm nie ma na celu zastąpienia szeregu metod wykrywania złośliwego oprogramowania. Ma on być jedynie uzupełnieniem arsenału do walki z niebezpiecznymi aplikacjami, pozwalającym na podniesienie skuteczności obecnie dostępnych narzędzi.

Algorytm może zostać zaimplementowany, jako usługa sieciowa wykonująca analizę na podstawie nazwy pakietowej aplikacji i listy uprawnień zadeklarowanych w pliku manifest. W trakcie analizy wykorzystany będzie parser sklepu Google Play w celu pobrania listy podobnych aplikacji i ich uprawnień.

Klient może zostać zaimplementowany, jako zwykła aplikacja dla systemu Android. Program klienta powinien mieć uprawnienie do otrzymywanie powiadomień o nowo zainstalowanych pakietach (aplikacjach), co skutkowałoby pobraniem uprawnień nowej aplikacji i wywołaniem usługi sieciowej w celu analizy. Klient analizatora powinien też mieć możliwość zainicjowania usunięcia pakietu przez użytkownika, jeżeli ten okazałby się niebezpieczny. Wszystkie opisane kroki działania aplikacji klienta są dostępne w API systemu Android – istnieje możliwość nasłuchiwania wiadomości systemowych na temat nowych aplikacji. Pobranie listy uprawnień nowej aplikacji umożliwia klasa PackageManager dostępna w API systemu Android. Istnieje też możliwość implementacji całości analizatora, jako aplikacji systemu Android, bez oddzielnej usługi sieciowej.

Ponieważ jedyne dane wejściowe analizatora to nazwa pakietowa oraz lista uprawnień istnieje możliwość implementacji klienta, jako programu poza systemem Android napisanego jakimkolwiek języku programowania ogólnego przeznaczenia, który umożliwia komunikację się z usługą sieciową analizatora.

5.2. Implementacja algorytmów różnicowej analizy uprawnień.

W trakcje prac zaimplementowano program analizujący zagrożenia udostępniający usługę sieciową REST pozwalającą na zdalne uruchomienie analizy. Program napisany został w języku JAVA z wykorzystaniem Spring Framework . Do uruchomienia wymaga on kontenera aplikacji webowych JAVA (ang. web container), np. Apache Tomcat.

W celu oceny działania programu i algorytmów został zaimplementowany program klienta pozwalająca na wywołanie interfejsu REST. Jest to program napisany w języku JAVA pozwalający na wsadowe uruchamianie analizy dla wielu aplikacji pochodzących ze zbiorów złośliwego oprogramowania oraz ze sklepu Google Play. Weryfikacja działania programu analizującego nie wymagała implementacji klienta dla systemu Android.

W kolejnych podpunktach opisano komponenty i sposób działania programu analizującego. Następnie przedstawiono funkcje programu klienta.

5.2.1. Komponenty programu analizującego

Program analizujący składa się z komponentów zaimplementowanych jako Spring Beans, pozwala na łatwe wstrzykiwanie zależności oraz rozszerzanie aplikacji. Program zawiera następujące komponenty:

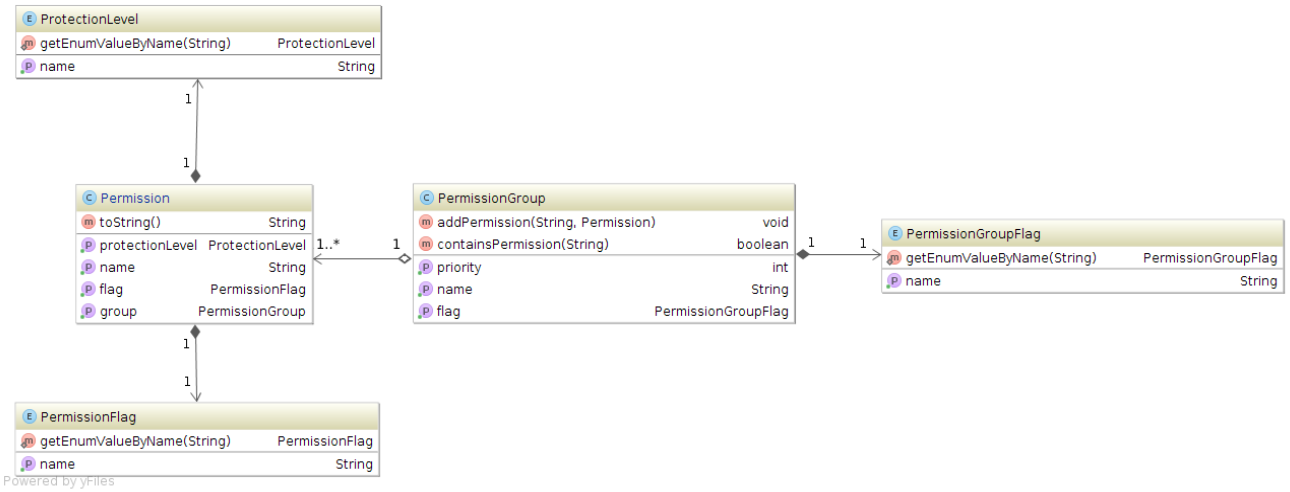
- Usługa parsowania zawartości stron ze sklepu Google Play (ApplicationDescriptionParserService)
- Usłga odczytywania informacji na temat uprawnień aplikacji (GooglePlayCrawlerService)
- Usługa dostarczająca informacje na temat wszystkich uprawnień i grup uprawnień z systemu Android (SystemPermissionsInfoService)
- Usługa dostarczająca dane wejściowe dla algorytmów (AlgorithmDataProviderService)
- Usługa uruchamiająca kolejne krotki algorytmu (AlgorithmExecutor)
- Analizator wyników algorytmów (ResultAnalyser)
- Kontroler serwisu REST udostępniający m. in. usługi analizy zagrożenia. (RestApiController)

- Fabryka obiektów opakowujących wynik działania algorytmu, pozwalająca na konfigurowanie poziomów zagrożeń (RiskScoreFactory)
- Klasy implementujące algorytmy oceny zagrożenie (interfejs AlgorithmStep)
    - Statyczna analiza uprawnień z manifestu (ScoreAppPermissionsStep)
    - Porównanie uprawnień z manifestu z uprawnieniami oryginalnej aplikacji ze sklepu Google (ContrastWithMarketVersionStep**)**
    - Porównanie uprawnień z aplikacjami podobnymi (ContrastWith-SimilarAppsStep)
    - Statyczna analiza uprawnień często implementowanych przez aplikacje złośliwe (ScorePopularMalwarePermissionsStep)
    - Porównanie uprawnień często implementowanych przez aplikacje złośliwe z aplikacjami podobnymi (ContrastWithSimilarApps-PopularMalwarePermissionsStep)

Diagram klas programu analizującego znajduje się w załączniku 2. W kolejnym punkcie opisano schemat uprawnień który jest wykorzystywany przez algorytmy oceniające zagrożenie, następnie opisano działanie aplikacji i interakcję pomiędzy komponentami.

5.2.2. Schemat uprawnień

Program analizujący zawiera klasy odpowiadające uprawnieniom systemu Android. Model uprawnień tworzony jest na początku działania aplikacji na podstawie pliku manifest definiującego uprawnienia w systemie Android. Manifest pobierany jest z repozytorium kodu źródłowego systemu Android [32]. Pełna lista uprawnień dostępnych dla aplikacji zewnętrznych znajduje się w załączniku 4.



Rysunek 3. Diagram obiektów modelu uprawnień.

Model uprawnień składa się z dwóch istotnych klas – uprawnienie (Permission) oraz grupa uprawnień (Permission Group), połączonych relacją wiele – jeden (1..N uprawnień w 1 grupie). Każde uprawnienie określa dostęp do jednego lub kilku zasobów w systemie, grupy uprawnień agregują uprawnienia na potrzeby prezentacji i akceptacji przez użytkownika. Relacje pomiędzy obiektami schematu uprawnień przedstawione są na rysunku 3.

Uprawnienie składa się z następujących atrybutów: nazwa, poziom ochrony, flaga dotycząca kosztów, grupa uprawnień do której należy dane uprawnienie. Poziom ochrony przyjmuje jedną z 4 wartości: normalny(normal), niebezpieczny (dangerous), na podstawie podpisu (signature), systemowy lub na podstawie odpisu (signature|system). Jedynie uprawnienia poziomu normalnego i niebezpiecznego są dostępne dla aplikacji zewnętrznych korzystających z zasobów systemowych. Ustawiona flaga informuje, że dane uprawnienie może bezpośrednio narazić użytkownika na koszty (np. poprzez wysłanie wiadomości SMS). Uprawnienie przechowuje również referencję do grupy uprawnień w której się znajduje.

Grupa uprawnień składa się z nazwy, flagi dotyczącej prywatności, priorytetu grupy oraz listy uprawnień należących do tej grupy. Ustawiona flaga informuje, że uprawnienia z danej grupy dają dostęp do danych prawnych użytkownika, priorytet grupy decyduje o kolejności wyświetlania ich użytkownikowi.

Schemat uprawnień wykorzystywany jest przez algorytmy oceniające zagrożenie. Dostęp do schematu odbywa się poprzez usługę – klasa SystemPermissionsInfoService.

5.2.3. Ocena zagrożenia aplikacji

Wykonywanie analizy wywoływane jest przez usługę sieciową REST (w klasie RestApiController). Ocena zagrożenia rozpoczyna się od przygotowania danych dla algorytmu oceny – pobrania listy aplikacji podobnych do analizowanej poprzez parsowanie strony internetowej z opisem badanej aplikacji w sklepie Google Play (klasa ApplicationDescriptionParserService).

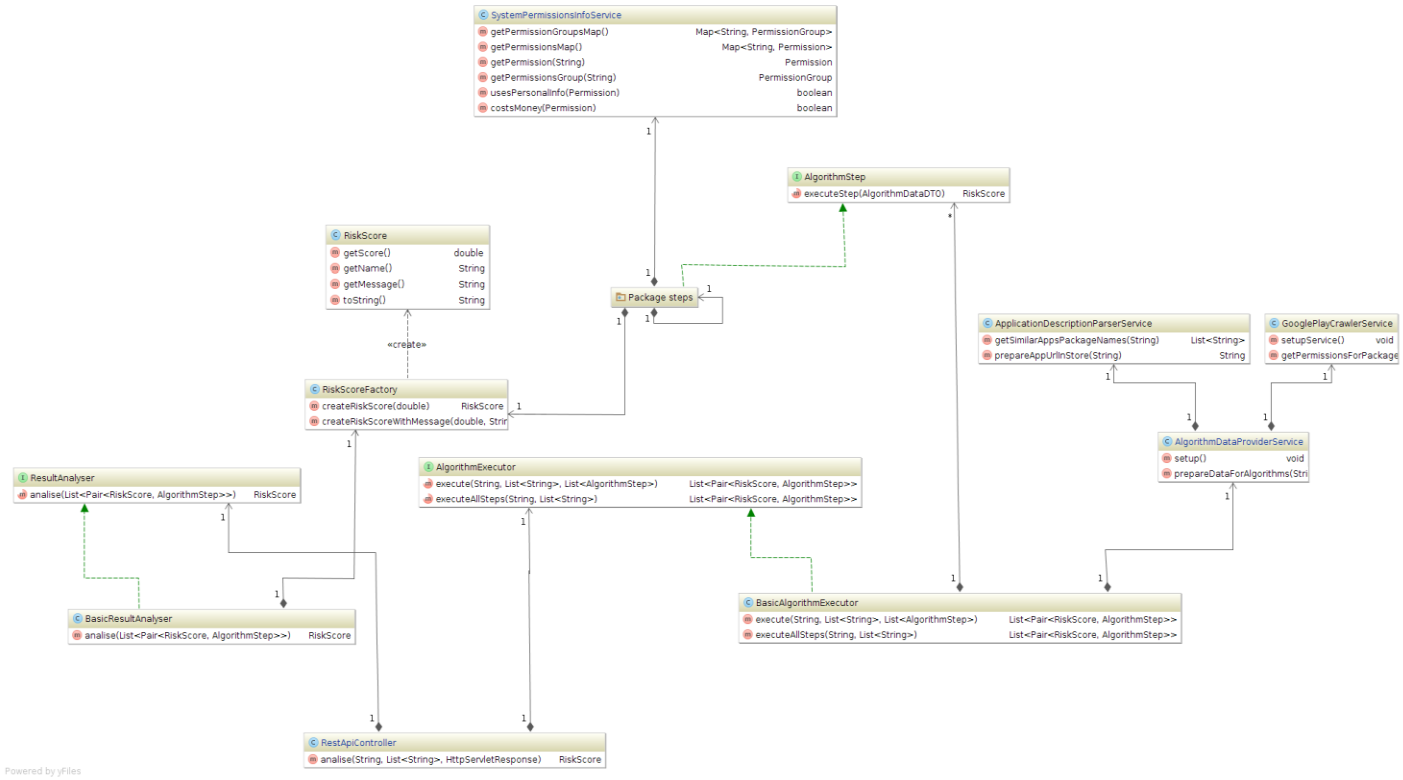
Następnie pobierana jest lista uprawnień każdej aplikacji podobnej oraz uprawnienia badanej aplikacji zadeklarowane w Google Play (klasa GooglePlayCrawlerService). Do implementacji użyto nieoficjalnego interfejsu programistycznego Google Play Crawler JAVA API [35] który pozwala na pełny dostęp do zasobów sklepu poprzez wykorzystanie takich samych protokołów (serializacja Protocol Buffers [39]) i metod uwierzytelnienia jak urządzenia z systemem Android. Dzięki temu możliwe było uzyskanie list uprawnień, które nie są widoczne w przeglądarkowej wersji sklepu.

Przygotowane dane trafiają do wykonawcy algorytmów (interfejs AlgorithmExecutor), który wywołuje niezależne algorytmy (klasy implementujące interfejs AlgorithmStep) i zwraca listę wyników.

Dalej wyniki przetwarzane są przez analizator wyników (interfejs ResultAnalyser) który wydaje ostateczny werdykt (typ RiskScore) zawierający wartość punktową zagrożenia, etykietę poziomu zagrożenia oraz widomość zawierającą opis uzyskania takiego werdyktu. Etykieta poziomu zagrożenia jest interpretacją wartości punktowej. Poziomy zagrożenia oraz odpowiadające im punkty są definiowane w pliku konfiguracyjnym którego listing znajduje się w załączniku 1.

Schemat klas wykorzystywanych w trakcie analizy zagrożenia przedstawiony jest na rysunku 4, klasy algorytmów oceniających znajdują się w pakiecie „steps" widocznym na środku diagramu. Klasa kontrolera (RestApiController) usługi sieciowej jest przedstawiona na dole diagramu. Ma on zależność od interfejsu wykonawcy algorytmów (AlgorithmExecutor) oraz interfejsu analizatora wyników.

W kolejnych punktach opisano 5 niezależnych algorytmów oceniających zagrożenie, oraz sposób działania agregatora wyników.

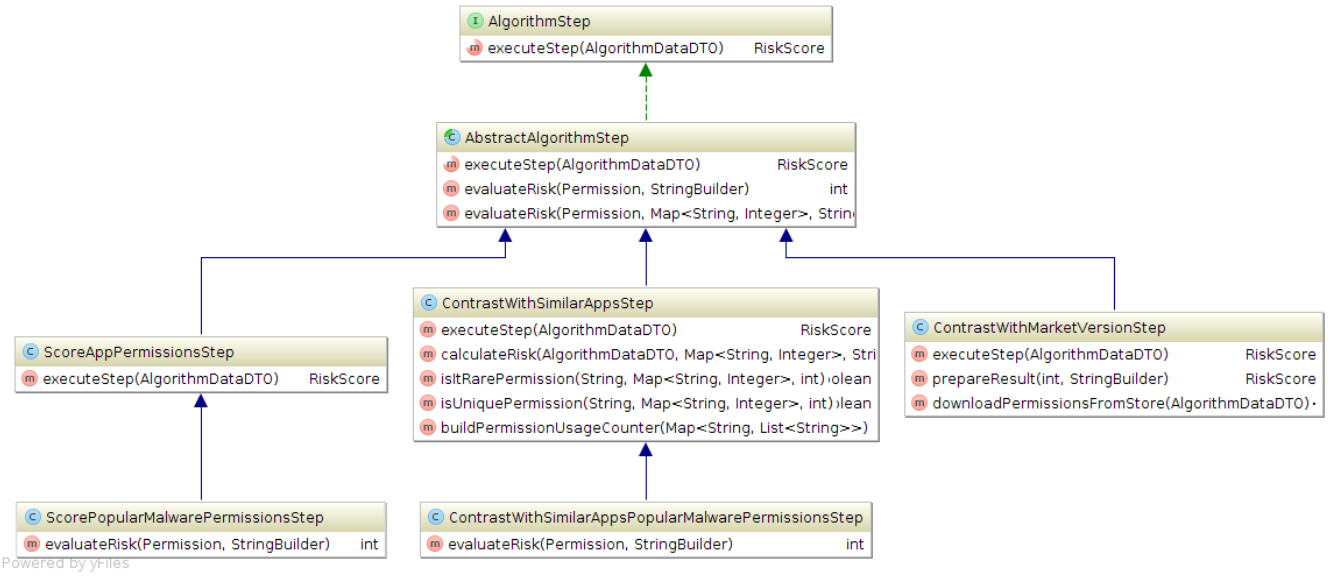


Rysunek 4. Diagram klas wykorzystywanych w trakcie oceny zagrożenia.

5.2.4. Algorytmy oceniające

Zaimplementowano 5 niezależnych metod oceny ryzyka. Klasy algorytmów implementują wspólny interfejs AlgorithmStep a ich instancje tworzone są przez kontekst aplikacji Spring. Pozwala to na łatwe rozszerzanie aplikacji o dodatkowe klasy analizujące. Każdy z algorytmów przypisuje aplikacji liczbę punktów ryzyka oraz zwraca wiadomość z techniczną informacją dotyczącą punktacji.

Hierarchia klas algorytmów przedstawiona jest na rysunku 5. W kolejnych podpunktach opisano działanie wszystkich nieabstrakcyjnych klas .



Rysunek 5. Hierarchia klas oceniających ryzyko.

5.2.4.1. Statyczna analiza uprawnień

Pierwszy, najprostszy algorytm wykonuje statyczną analizę pojedynczych uprawnień aplikacji (ScoreAppPermissionsStep). Punkty przypisywane są za każde uprawnienie w zależności o poziomu ochrony, ustawionej flagi dotyczącej kosztów oraz przynależności do grupy z ustawioną flagą dostępu informacji personalnych.

Wartości punktowe przypisywane za poszczególne rodzaje uprawnień ustawiane są w pliku konfiguracyjnym. Listing pliku konfiguracyjnego wykorzystywanego w trakcie analizy działania programu znajduje się w załączniku 1. Wartości punktów dla algorytmu statycznej analizy uprawnień znajdują się w kluczach rozpoczynających się od frazy „scoreAppPermissions”.

5.2.4.2. Statyczna analiza podejrzanych uprawnień

Algorytm statycznej analizy podejrzanych (ScorePopularMalwarePermissionsStep) uprawnień, podobnie jak poprzedni analizuje pojedyncze uprawnienia. Punkty przypisywane sią za każde uprawnienie typowe dla aplikacji złośliwych i jednocześnie nietypowe dla aplikacji łagodnych. Nie są brane pod uwagę atrybuty uprawnień (poziom ochrony, możliwość narażenia na koszty itd).

Tabela 8 zawiera listę podejrzanych uprawnień za które przypisywane są punkty. Lista została opracowana na podstawie [12] oraz [37].

Skrócona nazwa	Przedmiot uprawnienia
RECEIVE_SMS	Odbieranie i przetwarzanie SMS-ów.
READ_PHONE_STATE	Dostęp do funkcji telefonicznych urządzenia. Aplikacja z tym uprawnieniem może odczytać numer telefonu i identyfikator urządzenia, sprawdzić, czy połączenie jest aktywne, oraz poznać numer, z którym jest nawiązane połączenie
ACCESS_WIFI_STATE	Dostęp do informacji o połączeniach Wi-Fi – np. na sprawdzenie, czy obsługa Wi-Fi jest włączona, oraz odczytanie nazw podłączonych urządzeń Wi-Fi.
READ_SMS	Odczyt SMS-ów zapisanych na telefonie lub na karcie SIM.
RECEIVE_BOOT_COMPLETED	Uruchamianie natychmiast po zakończeniu rozruchu systemu.
WRITE_APN_SETTINGS	Pozwala aplikacji na zmianę ustawień sieciowych i przechwytywanie oraz analizowanie całego ruchu sieciowego –
WRITE_SMS	Zapisywanie wiadomości SMS przechowywanych w telefonie lub na karcie SIM. Szkodliwe aplikacje mogą usunąć wiadomości
WRITE_CONTACTS	Zapisywanie kontaktów.
SEND_SMS	Wysyłanie SMS-ów.

Tabela8. Lista podejrzanych uprawnień

Wartości punktowe przypisywane za poszczególne uprawnienia podejrzane są konfigurowane w pliku. Użyte wartości wypisano w załączniku 1 w kluczach rozpoczynających się od frazy „ScorePopularMalwarePermissionsStep". Istnieje możliwość rozszerzenia za pomocą konfiguracji listy uprawnień podejrzanych – wystarczy analogiczny wpis w pliku.

5.2.4.3. Porównanie uprawnień w Google Play

Kolejny zaimplementowany algorytm porównuje uprawnienia zadeklarowane w manifeście analizowanej aplikacji z uprawnieniami tej samej aplikacji dostępnej w sklepie Google Play (ContrastWithMarketVersionStep).

Różnice pomiędzy wersją analizowaną a wersją ze sklepu są podejrzanie, ale nie oznaczają jednoznacznie, że aplikacja była rekompilowana i modyfikowana. Rozszerzanie funkcji aplikacji powoduje potrzebę deklarowania kolejnych uprawnień.

Kolejne wersje systemu Android zmieniają dostępność niektórych uprawnień dla aplikacji zewnętrznych lub wprowadzają nowe uprawnienia, co również powoduje zmiany w manifeście.

Dla niektórych analizowanych aplikacji istnieje możliwość odczytania dokładnej wersji aplikacji z pliku manifest AndroidManifest.xml i porównania jej z aktualną wersją aplikacji w sklepie. Niestety, dla dostępnych zbiorów złośliwego oprogramowania wersje aplikacji są starsze niż aktualna wersja w sklepie. Ponadto w trakcie realizacji stwierdzono, że nawet nowo instalowane aplikacje ze sklepu mogą mieć wersje różną od tej, prezentowanej na stronie. Wersja może też być podana jako ewaluacja wyrażenia, np. „@string/version_number", co praktycznie uniemożliwia odczytanie aktualnej wersji.

Punkty przypisywane są za każde uprawnienia, które nie jest zadeklarowane w wersji ze sklepu. Wartości punktów, podobnie jak w przypadku analizy statycznej, są konfigurowalne w pliku i zależą od poziomu ochrony, ryzyka kosztów, oraz flagi dotyczącej informacji personalnych. Załącznik 1 zawiera użyte parametry algorytmu w kluczach rozpoczynających się od frazy „contrastWithMarket".

5.2.4.4. Porównanie uprawnień z aplikacjami podobnymi

Kolejny zaimplementowany algorytm porównuje uprawnienia zadeklarowane w manifeście aplikacji z uprawnieniami aplikacji podobnych, które są dostępne w sklepie Google Play (ContrastWithSimilarAppsStep).

Algorytm sprawdza częstość występowania danego uprawnienia wśród aplikacji podobnych. Punkty przypisywane są za każde uprawnienie, które występuje rzadko lub prawie nie występuje (uprawnienie unikatowe). Zakresy częstości występowania uprawnień które interpretowane są jako rzadki i unikatowy są konfigurowane.

Wartości przypisywanych punktów są konfigurowane w pliku i zależą od poziomu ochrony, ryzyka kosztów, flagi dotyczącej informacji personalnych. Wyżej punktowane są uprawnienia unikatowe. Załącznik 1 zawiera użyte parametry algorytmu – klucze rozpoczynające się od frazy „contrastWithSimilarAppsStep".

5.2.4.5. Analiza podejrzanych uprawnień w aplikacjach podobnych

Ostatni algorytm to połączenie porównywania uprawnień aplikacji podobnych z oceną uprawnień typowych dla aplikacji złośliwych (ContrastWithSimilarApps-PopularMalwarePermissionsStep).

Algorytm przypisuje wartości punktowe za te uprawnienia które są rzadkie lub unikatowe wśród aplikacji podobnych i są jednocześnie typowe dla aplikacji złośliwych.

Lista uprawnień typowych dla aplikacji złośliwych może być konfigurowana niezależnie od pozostałych algorytmów. W trakcie badań wykorzystano listę zawierającą takie same uprawnienia, co w przypadku statycznej analizy uprawnień podejrzanych.

Wartości punktowe przypisywane za poszczególne uprawnienia podejrzane wypisano w załączniku 1 w kluczach rozpoczynających się od frazy „ContrastWithSimilarAppsPopularMalwarePermissionsStep". Zakresy częstości występowania uprawnień które interpretowane są jako rzadki i unikatowy są takie same jak w przypadku porównywania uprawnień aplikacji podobnych.

5.2.5. Agregacja i werdykt

Wyniki algorytmów analizowane są przez klasę BasicResultAnalyser implementującą ResultAnalyser. Analizator sumuje wartości poszczególnych algorytmów, konkatenuje komunikaty po czym tworzy ostateczny wynik analizy. Ostateczny wynik analizy oprócz wartości punktowej i komunikatu zawiera nazwę poziomu zagrożenia, który jest prostą interpretacją wartości punktowej.

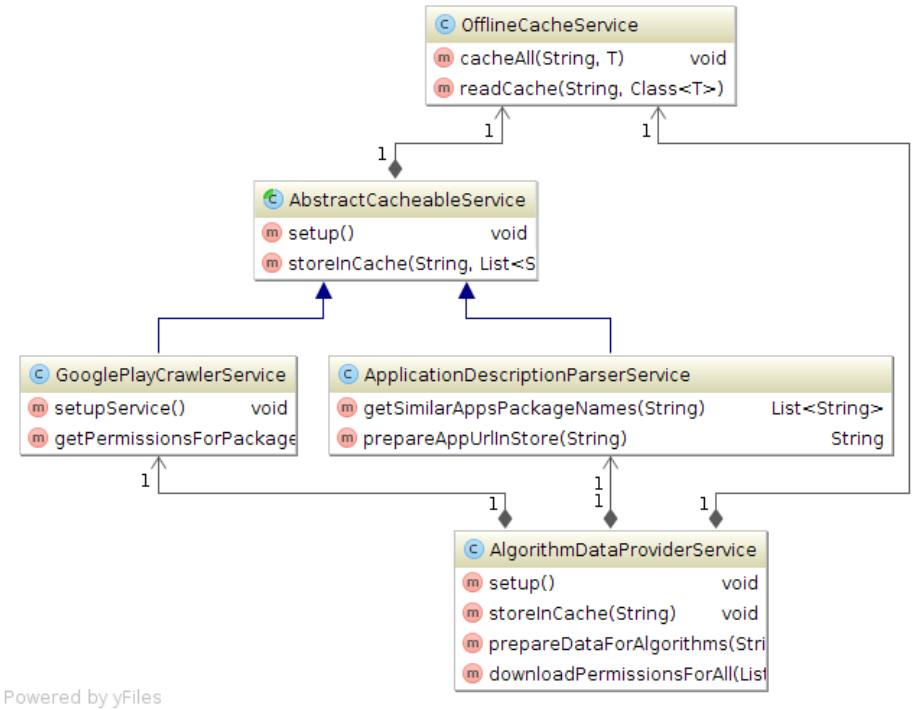
Liczba poziomów zagrożenia, ich nazwy oraz progi punktowe są konfigurowalne. W obecnej konfiguracji zdefiniowano poziomy: LOW(do 200 punktów.), LOWER (201-500 pkt.), NORMAL(501-1000 pkt.), HIGH (1001-2000), ,HIGHER (2001-4000), DANGEROUS (powyżej 4000 punktów). Parametry te znajdują się w załączniku 1 w kluczach „riskScore.names" oraz „riskScore.points".

W trakcie analiz zbiorów aplikacji złośliwych oraz badania popularnych aplikacji z różnych sklepów, pakiety dla których uzyskano zagregowany poziom zagrożenia DANGEROUS traktowane były jak złośliwe oprogramowanie.

5.2.6. Mechanizm pamięci podręcznej

Zaimplementowano generyczny mechanizm pamięci podręcznej. Jest on wykorzystywany w trakcie pobierania listy aplikacji podobnych (ApplicationDescriptionParserService) oraz przy pobieraniu listy uprawnień

(GooglePlayCrawlerService). Dodatkowo zapamiętywane są nazwy pakietowe aplikacji, które nie zostały odnalezione w sklepie Google Play, pozwala to na wczesne zaniechanie analizy(AlgorithmDataProviderService). Diagram klas wykorzystujących pamięć podręczną przedstawiono na rysunku 6.



Rysunek 6. Diagram klas wykorzystujących pamięć podręczną.

Pamięć podręczna jest serializowana do formatu JSON i zapisywana na dysku. Może zostać użyta ponownie po zmianie parametrów aplikacji a nawet w przypadku zmiany logiki komponentów, jeżeli tylko format danych przechowywanych z pamięci podręcznej nie uległ zmianie. Czyszczenie pamięci podręcznej można wykonać poprzez usunięcie pliku. Tekstowy format JSON pozwala też na dowolną zmianę zawartości pliku w celu przeprowadzenia dodatkowych testów.

Wykorzystanie pamięci podręcznej pozwoliło na przyspieszenie działania programu, zwłaszcza ponownego uruchamiania analizy z nowymi parametrami. W ten sposób zmniejszyła się liczba blokad dostępu do sklepu Google Play wynikających z nietypowego zachowania aplikacji. Dodatkowa zaleta rozwiązania to uniezależnienie wyników analizy od zmian w sklepie. Wyniki są powtarzalne, dzięki czemu można zbadać wpływ poszczególnych parametrów konfiguracyjnych na działanie algorytmów.

5.3. Program klienta

W celu oceny działania programu wykrywającego zaimplementowany został klient usługi sieciowej REST. Jest to program napisany w języku JAVA z wykorzystaniem Spring Framework, pozwalający na wsadowe uruchamianie analizy dla wielu plików pochodzących ze zbiorów złośliwego oprogramowania.

Weryfikacja działania programu analizującego nie wymagała implementacji klienta dla systemu Android. Co więcej, taki klient nie umożliwiałby on wsadowego wykonywania analizy. Możliwy sposób implementacji klienta dla systemu Android opisano na początku tego rozdziału w punkcie 5.1.

Dla każdego analizowanego pliku apk, wykonano dekompilację za pomocą narzędzia apktool [16]. Program klienta parsuje pliki AndroidManifest.xml ze zdekompilowanych aplikacji, a następnie wywołuje usługę programu analizującego przekazując nazwę pakietu oraz listę uprawnień. Po odebraniu wyników analizy, wynik zostaje zapisany w pliku.

Możliwe jest też uruchomienie analizy aplikacji ze sklepu Google Play bez pobierania i dekompilowania pliku apk. W takim przypadku jedynym parametrem przesyłanym przez klienta jest nazwa pakietowa aplikacji, która ma zostać przeanalizowana. Program analizatora pobiera ze sklepu Google Play listę uprawnień analizowanej aplikacji a następnie wykonuje jej analizę tak samo, jak w poprzednich przypadkach.

Kolejna ważna funkcja programu klienta to parsowanie sklepu Google Play oraz chińskiego sklepu z aplikacjami GFAN [38] w celu pobrania listy najbardziej popularnych aplikacji.

Ostatnia istotna funkcja klienta to możliwość przeanalizowania listy aplikacji ze sklepu lub zdekompilowanych plików apk pod kontem najpopularniejszych uprawnień i liczby uprawnień deklarowanych przez aplikacje z listy lub z plików.

Na podstawie tych dwóch ostatnich funkcji wykonano analizę uprawnień i zagrożeń najpopularniejszych aplikacji ze sklepu Google Play oraz GFAN. Wynik analizy zaprezentowano w rozdziale 7.

Schemat komponentów Spring programu klienta znajduje się w załączniku 3.

5.4. Podsumowanie

Po analizie literatury dotyczącej wykrywania złośliwego oprogramowania w systemie Android zaproponowano nowy sposób analizy zagrożeń – kontrastowanie analizowanej aplikacji z podobnymi aplikacjami dostępnymi w sklepie Google Play.

W trakcie realizacji algorytmu powstała platforma pozwalająca na implementowanie niezależnych metod wykrywania złośliwych aplikacji, również takich algorytmów, które potrafią wykorzystać informacje na temat uprawnień aplikacji podobnych. Sposób implementacji platformy pozwala na łatwe rozszerzanie o kolejne algorytmy, zmianę implementacji agregatora wyników lub sposobu uruchamiania kolejnych algorytmów.

Wykonano pięć niezależnych algorytmów wykorzystujących część lub komplet danych dostarczanych przez platformę. Najważniejsze zaimplementowane metody oceny ryzyka to porównywanie uprawnień z aplikacjami podobnymi w celu znalezienia uprawnień nadmiarowych oraz porównywanie uprawnień z aplikacjami podobnymi w celu znalezienia uprawnień typowych dla aplikacji złośliwych. Są to autorskie metody których nie znaleziono w literaturze dotyczącej wykrywania złośliwego oprogramowania dla systemu Android..

Komplet parametrów konfiguracyjnych znajduje się w załączniku 1. Wartości tych parametrów zostały dobrane w trakcie realizowania kolejnych algorytmów metodą „ręcznej” optymalizacji na podstawie informacji generowanych przez algorytmy.

Program klienta pozwala na wsadowe uruchamianie analizy aplikacji złośliwych programów ze sklepu Play oraz umożliwia analizę uprawnień i zagrożeń najpopularniejszych aplikacji ze sklepu Google Play oraz GFAN. Dokładny opis uzyskanych wyników przedstawiono w kolejnym rozdziale. Następnie zaprezentowano analizę najpopularniejszych uprawnień aplikacji wykonaną przy pomocy programu klienta.

6. Analiza działania algorytmu

W niniejszym rozdziale zaprezentowano wyniki działania programu analizującego zagrożenia. Rozpoczęto od omówienia sposobu przygotowania danych testowych – aplikacji złośliwych i zaufanych. Następnie znajduje się punkt zawierający wyniki liczbowe oraz ich analizę. Na końcu omówiono zauważone wady i braki zrealizowanego programu oraz opisano możliwe sposoby udoskonalenia.

6.1. Dane testowe

Jako danych testowych użyto zbiorów złośliwych aplikacji dla systemu Android pochodzących z projektu VirusShare[33] oraz bloga Contagio Dump[34]. Są to zbiory nieodpłatne, dostęp do nich wymaga jedynie podania krótkiego wyjaśnienia swoich motywów. Aplikacje złośliwe, udostępnione jako binarne archiwa systemu Android wymagały wcześniejszej dekompilacji za pomocą narzędzia apktool [16].

Jako zbiór aplikacji niezłośliwych wykorzystano popularne aplikacje ze sklepu Google Play. Przygotowanie polegało na pobraniu za pomocą programu klienta listy najpopularniejszych aplikacji z każdej kategorii w sklepie. Lista popularnych aplikacji jest tworzona w sklepie Google Play zależnie od lokalizacji. W tym przypadku aplikacje były pobierane z polskiego adresu IP. Analizowane aplikacje niezłośliwe nie były pobierane ani dekompilowane. Lista uprawnień, wymagana w analizie, pobierana była ze sklepu.

Parametry konfiguracyjne programu – wartości punktowe algorytmów oraz zakresy poziomów zagrożenia opracowane zostały w sposób empiryczny już w trakcie opracowywania kolejnych kroków, a także w trakcie kilku uruchomień testowych analizatora. Z pewnością nie są to wartości optymalne, ale są wystarczające do oceny przydatności programu analizującego zagrożenia. Lista parametrów konfiguracyjnych znajduje się w załączniku 1.

6.2. Wyniki i analiza

Po przygotowaniu danych testowych wykonano analizę 1248 aplikacji złośliwych oraz 35463 aplikacji pochodzących ze sklepu Google Play. Zestawienie wyników zaprezentowano w tabeli 9. Wyniki podzielono na 4 kategorie – aplikacje pochodzące z

bazy VirusShare, aplikacje z Contagio Dump, aplikacje darmowe z Google Play oraz Aplikacje płatne z Google Play.

	Wirusy z bazy Contagio	Wirusy z Virusshare	Popularne darmowe Google Play	Popularne płatne Google Play
Liczba aplikacji / poziom zagrożenia	42	1206	21998	13465
LOW	1 2.38%	75 6.22%	8467 38.49%	6963 51.51%
LOWER	0 0%	67 5.56%	3755 17.07%	1383 10.27%
NORMAL	0 0%	60 4.98%	2428 11.04%	1391 10.33%
HIGH	6 14.29%	102 8.46%	3440 15.64%	1769 13.14%
HIGHER	6 14.29%	161 13.35%	2175 9.89%	1104 8.20%
Niebezpieczne (DANGEROUS)	29 **69.05%**	741 **61.44%**	1733 **7.88%**	882 **6.55%**
Suma niebezpiecznych	**61.70%**		**7.35%**	

Tabela 9. Wyniki analizy

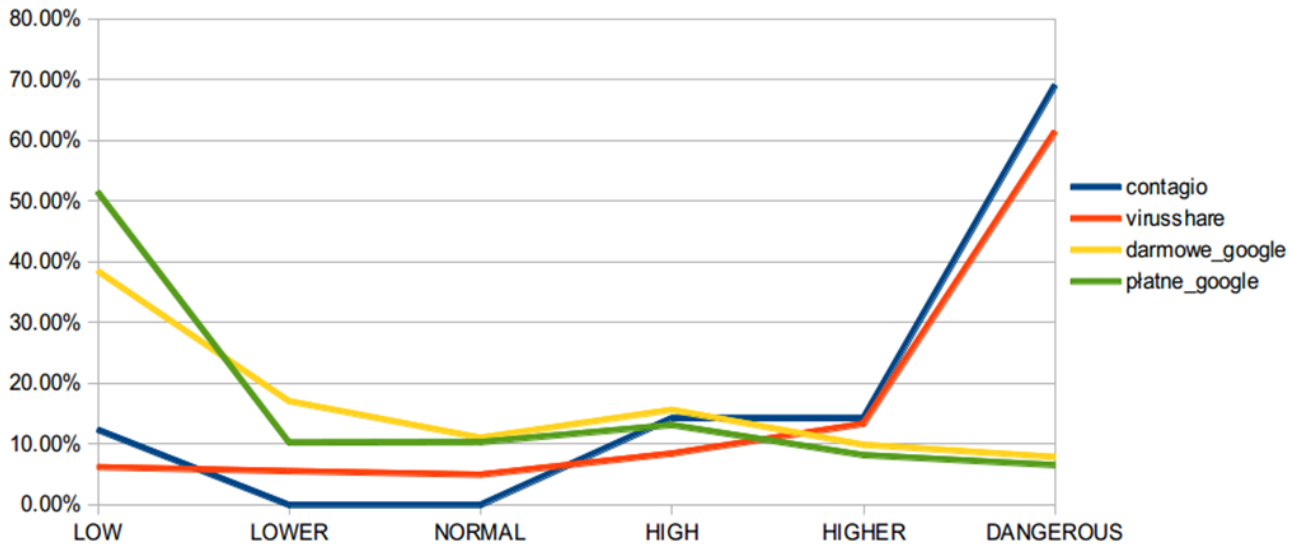
W tabeli pokazano liczbę i odsetek aplikacji zakwalifikowanych do każdego poziomu zagrożenia. Ostatni wiersz tabeli zawiera sumaryczną wartość procentową aplikacji zinterpretowanych jako niebezpieczne w zbiorze wirusów oraz w sklepie Google.

Program analizujący wykrył poprawnie ponad 60% złośliwych aplikacji (ang. true positive rate). Jest to dobry wynik biorąc pod uwagę, że wykorzystuje niezaawansowane algorytmy analizujące głównie różnice w uprawnieniach deklarowanych przez aplikacje podobne.

Program analizujący zakwalifikował około 7% popularnych aplikacji ze sklepu Google Play jako aplikacje niebezpieczne. Nie należy takich przypadków interpretować

to jako błędne – są to aplikacje posiadające zbyt szerokie uprawnienia w stosunku do swoich funkcji. Wykryto w ten sposób, między innymi aplikację dynamicznego tła pulpitu która posiadała 7 z 9 uprawnień typowych dla aplikacji złośliwych, w tym uprawnienia do wysyłania i usuwania smsów. Znaleziono też wiarygodnie wyglądającą aplikację znanej firmy sportowej która miała uprawnienia do odczytywania i modyfikowania skrzynki odbiorczej SMS, rejestru połączeń oraz odczytywania precyzyjnej lokalizacji użytkownika.

Porównując liczbę aplikacji w poszczególnych poziomach zagrożenia warto zwrócić uwagę, że do dwóch najwyższych poziomów zagrożenia zakwalifikowano aż 75% aplikacji złośliwych i mniej niż 18% popularnych aplikacji ze sklepu Google Play. Wykres 1 przedstawia odsetek aplikacji na różnych poziomach zagrożenia dla czterech kategorii. Widać na nim, że większość aplikacji ze sklepu Google Play oceniono jako niegroźne (LOW, LEWER, NORMAL) natomiast większość aplikacji z baz wirusów oceniono jako groźne.

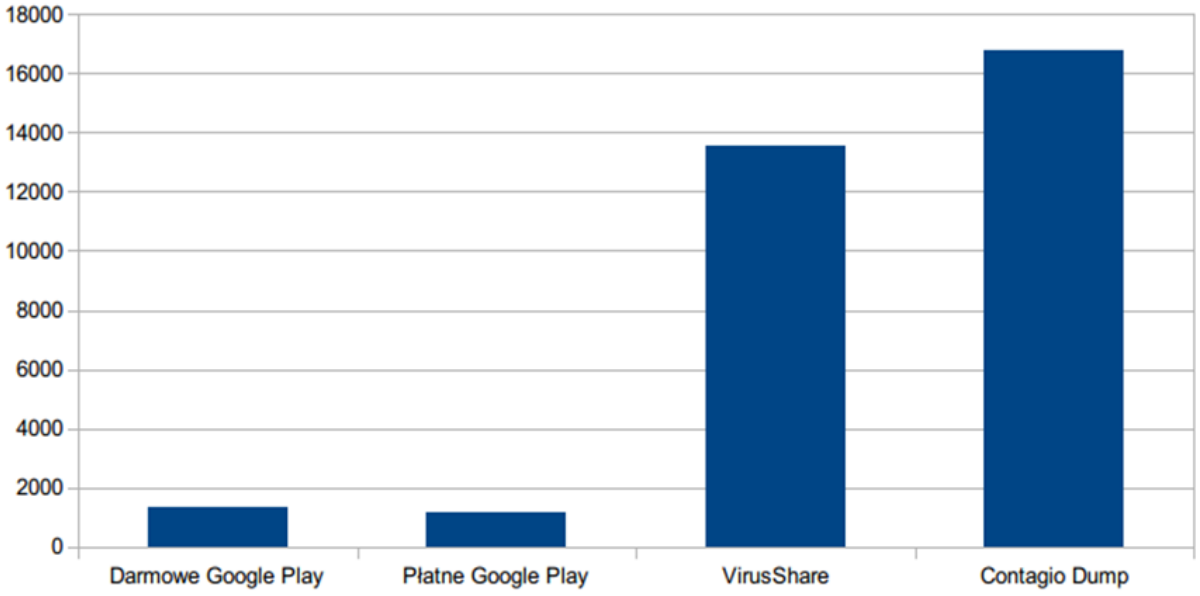


Wykres 1. Odsetek aplikacji na różnych poziomach zagrożenia dla czterech kategorii.

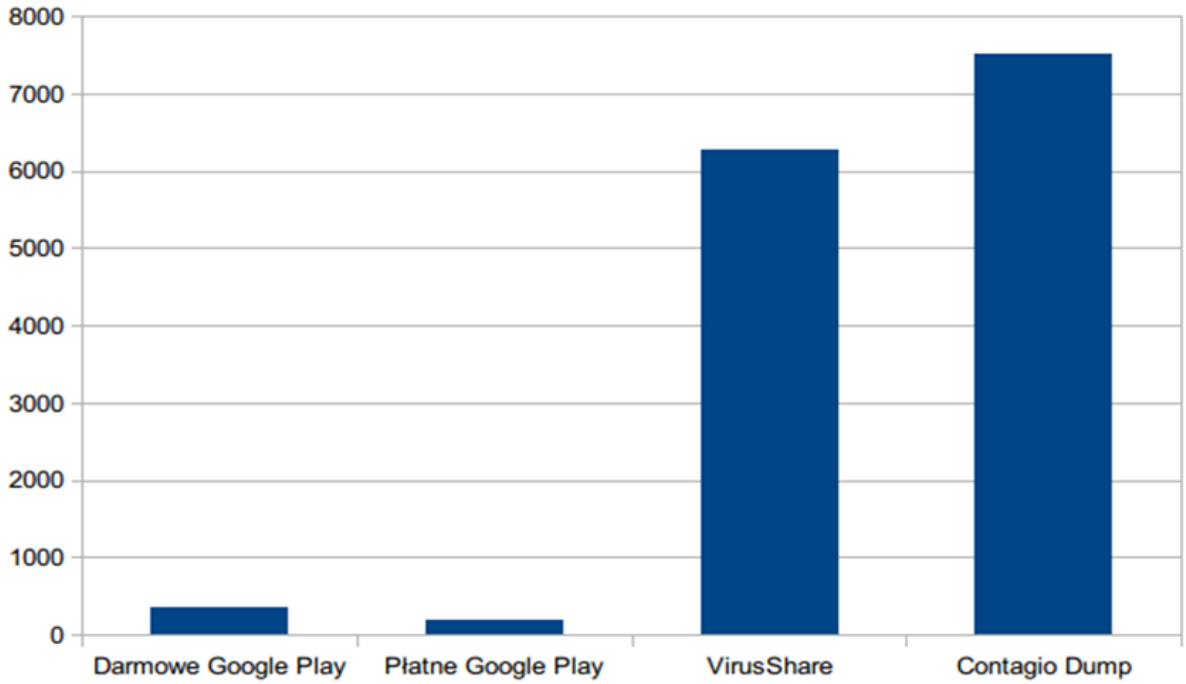
Zbiór aplikacji z projektu Virusshare zawiera stosunkowo dużo aplikacji o niskiej ocenie zagrożenia. Są to zwykle aplikacje korzystające z niewielu niestandardowych uprawnień. Lista ich uprawnień, oraz wyniki poszczególnych algorytmów analizujących nie pozwalają stwierdzić, że aplikacja może być złośliwa.

Wykres 2 przedstawia wartości średnie ocen zagrożenia dla czterech kategorii aplikacji. Warto zwrócić uwagę na kilkukrotnie wyższe wartości zagrożenia aplikacji ze

zbiorów zawierających złośliwe oprogramowanie. Jeszcze większe widoczne są na wykresie wartości środkowej ocen zagrożenia – wykres 3.

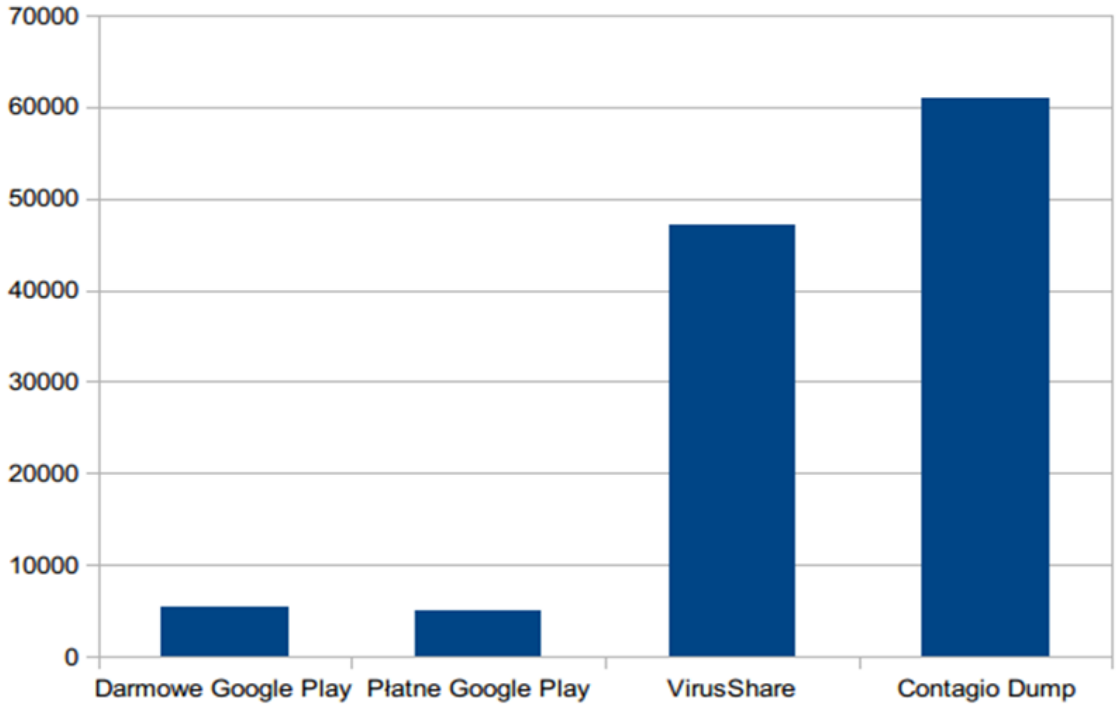


Wykres 2. Średnie wartości punktów zagrożenia analizowanych



Wykres 3. Mediana wartości punktów zagrożenia analizowanych

Ostatni wykres 4 przestawia wartość percentylu 95% ocen zagrożenia dla poszczególnych kategorii. Z analizy mediany i percentylu 95% można wnioskować, że większość analizowanych aplikacji złośliwych otrzymała punkt zagrożenia zdecydowanie przekraczające najwyższy próg – mediana ich wartości wynosi ponad 6000 punktów, najwyższy próg ustawiono na 4000 punktów. Analizowane aplikacje ze sklepu Google Play przekraczają próg aplikacji niebezpiecznych o znacznie mniejsze wartości – percentyl 95% wartości wynosi 5005 dla aplikacji płatnych i 5427 dla aplikacji darmowych.



Wykres 4. Percentyl 95% wartości punktów zagrożenia analizowanych

Przedstawione wyniki pokazują prawidłowe działanie programu analizującego złośliwe oprogramowanie. Prawidłowo wykryto ponad 60% złośliwych aplikacji ze zbiorów testowych, natomiast wynik 7-8% złośliwych aplikacji ze sklepu Google Play zawiera zarówno fałszywe alarmy jak i wykrycie potencjalnie niebezpiecznych aplikacji wykorzystujących zbyt dużą liczbę uprawnień w stosunku do kategorii i oferowanych funkcji.

Oceniając skuteczność warto zauważyć, że zaimplementowane algorytmy analizuję aplikacje jedynie pod kontem uprawnień. Oznacza to, że nie są w stanie

wykrywać niektórych rodzajów zagrożeń, na przykład aplikacji wykorzystujących podatności w celu uzyskania uprawnień użytkownika root. W takim przypadku złośliwa aplikacja ma pełny dostęp do systemu i może działać niezależnie od deklarowanych uprawnień. Inny przykład to aplikacja tworząca furtkę umożliwiająca zdalne sterowanie zachowaniem aplikacji przez właściciela botnetu. W takim przypadku jedyne wymagane uprawnienia to dostęp do Internetu i ewentualnie usługi Google Cloud Messaging [48] pozwalającej na łatwe przesyłanie wiadomości (poleceń) do aplikacji. Oczywiście aplikacja zmodyfikowana, która nie deklaruje żadnych nowych uprawnień również nie zostanie oceniona jako niebezpieczna.

6.3. Wady zaimplementowanego algorytmu

W trakcie wykonywania analizy działania programu wykrywającego złośliwe oprogramowanie zidentyfikowana kilka wad przedstawianego podejścia. Poniżej opisano najważniejsze z nich oraz podano, tam gdzie to możliwe, sposoby na złagodzenie wady.

6.3.1. Brak aplikacji w sklepie Google Play

Pierwszy poważny problem z którym się spotkano to trudność w znalezieniu analizowanych aplikacji w sklepie Google Play. Wyszukiwanie odbywało się za pomocą nazwy pakietowej aplikacji i dla większości próbek złośliwego oprogramowania nie znaleziono odpowiedników w sklepie Google Play. Oznacza to, że nie da się sprawdzić oryginalnej listy uprawnień ani pobrać uprawnień aplikacji podobnych. Takie aplikacje nie były analizowane.

Udało się wykonać analizę jedynie dla 42 na 513 wirusów z bazy Contagio Dump, oraz dla 1206 z 14500 wirusów z Virusshare. Wyniki analizy zawarte na początku rozdziały zawierają tylko te skutecznie przeanalizowane aplikacje.

Istnieje możliwość wyszukania oryginalnej aplikacji w sklepie Google Play za pomocą nazwy wyświetlanej użytkownikowi – da się ją uzyskać zarówno ze zdekompilowanych próbek jak i w sposób programowy na urządzeniu użytkownika w przypadku implementacji klienta na urządzenia Android.

Próby wyszukiwania za pomocą takiej nazwy pokazały, że zwracane wyniki są nietrafione, albo bardzo niejednoznaczne. Wyszukanie aplikacji o nazwie „News

Stream” z bazy Contagio Dump zwraca wiele aplikacji zawierających wiadomości, lecz żadna z nich ma identycznej nazwy jak wyszukiwana fraza. Podobnie jest w przypadku aplikacji o nazwie „Awesome Jokes” z Contagio Dump.

Inny przykład to wyszukanie aplikacji o nazwie „XXX_video” które zwraca wiele wyników, ale żaden z nich nie jest analizowaną aplikacją, co wynika z polityki treści w sklepie Gogle Play – rozrywka dla dorosłych nie jest dostępna.

Dodatkowo problem wyszukiwania po nazwie prezentowanej użytkownikowi to język w którym została podana nazwa aplikacji. Aby wykiwanie było prawidłowe należy użyć tej samej wersji językowej strony sklepu Google Play. W przypadku implementacji klienta na urządzeniu Android nie sposób stwierdzić czy nazwa wyświetlana jest w języku ustawionym na urządzeniu. Możliwe jest też, że nazwa dla języka ustawionego na urządzeniu nie była dostępna i system wyświetla nazwę domyślną.

Wyszukanie aplikacji po nazwie można rozszerzyć o programowe pobranie aplikacji z Google Play, dekompilacje i porównywanie zasobów w celu znalezienia oryginału, ale i to nie gwarantuje sukcesu – zwykle zasoby podrobionych aplikacji są mniejsze, i nawet ikony różnią się od oryginałów. Podróbka gry World of Goo, z bazy Contagio Dump, ma ikonę zawierającą napis „FREE”. Kolejny problem takiego podejścia to możliwość blokady pobrania aplikacji ze względu na lokalizację którą wykryje sklep Google Play.

Pomimo wielu potencjalnych problemów implementacja wyszukiwania nazwy aplikacji może odnaleźć część aplikacji pod które podszywają się złośliwe aplikacje.

6.3.2. Ograniczenia w dostępie do sklepu Google Play

Kolejny problemem jest blokowanie dostępu do sklepu Google Play gdy zostanie wykryte nietypowe zachowanie z jakiejś podsieci – na przykład bardzo częste (kilka, kilkanaście razy na sekundę) wykonywanie takich samych żądań. Wykorzystywany interfejs programistyczny [39] nie jest oficjalnie wspierany przez Google, i nie dokumentacji w której można poznać nakładane ograniczenia.

W celu przeciwdziałaniu blokadom zaimplementowano spowolnienie pobierania listy uprawnień aplikacji podobnych oraz ponawianie pobierania w przypadku błędu. Pozwoliło to na wyeliminowanie problemu blokad kosztem wydłużonego czasu analizy aplikacji. Następnie zaimplementowano mechanizm

pamięci podręcznej w programie analizującym, dzięki czemu ponowne wykonywanie analizy dla tych samych aplikacji jest bardzo szybkie.

6.3.3. Wymagane uwierzytelnienie konta Google

Program analizujący korzysta z nieoficjalnego interfejsu programistyczngo Google Play Crawler JAVA API, który wymaga podania loginu i hasła do konta Google. Na potrzeby wykonania i przetestowania algorytmu założone zostało nowe konto jednak może to powodować kilka problemów. Nie zostało zweryfikowane czy takie konto jest zgodne z regulaminem usług, więc możliwe, że zostanie usunięte bez ostrzeżenia. Nie badano czy można w ten sposób używać konto użytkownika dla którego aktywowano dwuskładnikowe uwierzytelnienie.

Kolejne problemy mogą się pojawić gdy z usługi będzie korzystać więcej klientów. Należy wtedy skorzystać z programistycznego interfejsu uwierzytelnienia programów zewnętrznych.

6.3.4. Wyniki zależne od algorytmów wewnętrznych sklepu Google Play

Lista aplikacji podobnych jest pobierana ze strony Google Play (sekcja „Podobne”) i może być personalizowana w zależności od urządzenia które wyświetla stronę (nagłówek „User Agent” protokołu http), pozostałych nagłówków HTTP żądania, lokalizacji i innych parametrów. Algorytmy decydujące o treści sklepu nie są znane i mogą powodować różne wyniki analizy zagrożeń.

W trakcie przeprowadzania badań stwierdzono, że lista aplikacji popularnych jest zależna od języka i lokalizacji, nie zauważono natomiast różnic w liście aplikacji podobnych.

6.3.5. Jakość i ilość danych testowych

Mankamentem jest jakość badanych aplikacji złośliwych. Większość zawiera bezużyteczną treść a zawartość plików po dekompilacji wygląda jakby były tworzone przez automat. Trudno sobie wyobrazić, żeby mogły nabrać użytkownika szukającego wartościowej aplikacji. Możliwe, że twórcy takich wirusów liczą na jednorazowe uruchomienie przez użytkownika i natychmiastowe przeprowadzenie złośliwej akcji –

przejęcia kontroli poprzez podatność albo spieniężenie ataku przez wysłanie SMSa premium lub wykonanie połączenia telefonicznego.

Ostatni problem to liczba aplikacji złośliwych dla których udało się przeprowadzić analizę – w sumie 1248 próbek. Nie było możliwe użycie narzędzia ADAM [40] które pozwala na przeprowadzenie testów programu antywirusowego dla wielu modyfikacji tej samej próbki złośliwego oprogramowania. ADAM zaciemnia kod złośliwej aplikacje w taki sposób utrudnić wykrycie za pomocą sygnatur. Jest to technika stosowana przez twórców złośliwego oprogramowania. Niestety manifesty aplikacji nie są modyfikowane, poza tym nie ma możliwości modyfikacji nazw uprawnień – aplikacja przestałaby działać.

Pomimo wielu prób, nie udało się uzyskać dostępu do bazy Android Malware Genome Project[41], która była wykorzystywana w wielu publikacjach. Strony projektu oraz strony domowe twórców pokazują, że ich działalność w dziedzinie bezpieczeństwa systemu Android nie jest kontynuowana.

6.4. Możliwości rozwoju platformy

W trakcie analizy działania zaimplementowanych algorytmów zidentyfikowano wiele obszarów potencjalnego usprawnienia i rozwoju, zostały one opisane w kolejnych podpunktach. Przedstawiono też możliwości dodania mechanizmów uczenia maszynowego oraz możliwe wykorzystanie zrealizowanego programu w celu dostarczania informacji użytkownikowi na temat instalowanych aplikacji.

6.4.1. Nowe algorytmy

Architektura rozwiązania pozwala na łatwe rozszerzanie logiki oceniającej zagrożenie. Pierwszy proponowany algorytm miałby wykorzystywać priorytet grupy uprawnień. W obecnej implementacji żaden z algorytmów nie bierze pod uwagę wartości atrybutu priorytet grupy do której należy uprawnienie. Grupy uprawnień wyświetlane użytkownikowi są uporządkowane według priorytetu, oznacza to, że aplikacje należące do grupy o wyższym priorytecie są bardziej niebezpieczne i użytkownik powinien zwrócić na nie uwagę.

Drugi z nowych algorytmów miałby porównywać aplikację ze sklepem Google Play jedynie w przypadku stwierdzenia, że analizowana jest ta sama wersja aplikacji.

Wykrycie różnic oznaczałoby jednoznacznie, że aplikacja została zmodyfikowana i stanowi zagrożenie. Niestety taka sytuacja nie zdarzała się często w przypadku analizowanych aplikacji złośliwych.

Ostatni pomysł na algorytm to modyfikacja programu tak, aby uwzględniać brak aplikacji w sklepie. W obecnej implementacji analizowane są aplikacje które można znaleźć w sklepie, a te stanowią mniejszość badanych pakietów złośliwych. Brak aplikacji może być istotną informacją przy ocenie.

W obecnej implementacji analizator wyników nie posiada prawie żadnej logiki – sumuje jedynie wyniki z algorytmów i przypisuje nazwę poziomu zagrożenia. Można zaimplementować bardziej zaawansowane metody analizy wyników np. sumowanie z wyników algorytmów z różnymi wagami, zaawansowane reguły decyzyjne, czy mechanizm głosowania.

Możliwe jest też dostrojenie parametrów aplikacji. W trakcie realizacji pracy były one regulowane ręcznie na podstawie analizy wyników i logów działania poszczególnych algorytmów. Można wykonać dalszą optymalizację w opisany sposób albo wykorzystać metody uczenia maszynowego, co zostało zaproponowane w kolejnym punkcie.

6.4.2. Wykorzystanie uczenia maszynowego

Zidentyfikowano dwa obszary w których można zaimplementować uczenie maszynowe w celu poprawy działania algorytmu – działanie analizatora (sumatora) wyników oraz optymalizacja parametrów aplikacji.

Wartości zagrożenia z pięciu algorytmów mogą być użyte jako dane wejściowe algorytmu analizatora wyników. Jest to przykład maszyny komisyjnej (ang. Committee machine), w której można zastosować algorytmy uczenia zespołowego (ang. Ensemble learning) czyli poprawy wyniku klasyfikacji poprzez odpowiednie wykorzystanie wcześniejszych algorytmów.

Najprostszy przykład zespołowego podejmowania decyzji to głosowanie w którym odpowiednia liczba punktów zagrożenia z każdego algorytmu interpretowana jest jako zagrożenie. Odpowiednia liczba głosów o zagrożeniu, niekoniecznie większość, oznacza ze wszystkich algorytmów oznacza, że analizowana aplikacja jest niebezpieczna.

Głosowanie może być udoskonalone przez algorytm uśredniania modelu Bayesa (ang. Bayesian model averaging) lub przez algorytm kombinacji modeli Bayesa (ang. Bayesian model combination). Są to algorytmy które wykorzystują klasyfikator bayesowski w celu, odpowiednio, znalezienia optymalnych wag głosu wcześniejszych algorytmów lub znalezienia optymalnych kombinacji wcześniejszych algorytmów.

Innym sposobem udoskonalenie wyników jest zastosowanie sieci perceptronowej w miejscu analizatora wyników. Wejściem pierwszej warstwy neuronów byłby wektor pięciu wyników z algorytmów oceniających zagrożenie. Na wyjściu ostatniej warstwy pojawiałby się wynik klasyfikacji.

Optymalizacja parametrów wejściowych aplikacji (załącznik 1) jest znacznie bardziej skomplikowanym zagadnieniem. Wykonany program można traktować jako funkcję której argumentami są parametry wyjściowe a wynikiem liczba poprawnie zaklasyfikowanych aplikacji, w kategoriach groźny/niegroźny. Optymalizacja parametrów wejściowych jest więc zadaniem znalezienia maksimum globalnego takiej funkcji. W obecnej implementacji aplikacja wykorzystuje 41 parametrów wejściowych, z których część jest ze sobą skorelowana i mogłaby zostać zredukowana, nie mniej znalezienie maksimum globalnego funkcji $f : R^{41} \Rightarrow R$ jest zadaniem bardzo złożonym, które powinno być zaimplementowane w dalszym rozwoju platformy.

6.4.3. Implementacja klienta w systemie Android

Ostatnie intersujące rozszerzenie, to implementacja klienta algorytmu jako aplikacji antywirusowej w systemie Android, o której napisano w propozycji algorytmu w rozdziale 5. Interfejsy programistyczne systemu udostępniają wszystkie dane wymagane do uruchomienia analizy zagrożenia.

Natywny klient w systemie Android mógłby zostać wykorzystany także w celu informacyjnym i uświadamiającym użytkownika o potencjalnych zagrożeniach. Wyniki liczbowe analizy można zwizualizować pokazując na ile groźne są poszczególne aplikacje, podobnie jak wykonano to w [28].

Każda z implementacji pozwalałaby na realne wykorzystanie zaimplementowanych narzędzi w celu ochrony przed złośliwym oprogramowaniem. Jednak w realizacji prac nad programem do analizy zagrożeń implementacja aplikacji dla systemu Android nie była konieczna – wszystkie testy, weryfikacje i analizy

przeprowadzono znaczniej łatwiej i szybciej przy wykorzystaniu komputera osobistego i klienta napisanego w języku JAVA.

7. Analiza uprawnień i zagrożeń najpopularniejszych aplikacji

W trakcie realizacji pracy wykonano analizę wykorzystania uprawnień przez najpopularniejsze aplikacje ze wszystkich kategorii dostępnych w klepie Google Play [47]. Przeanalizowano 21998 aplikacji bezpłatnych oraz 13465 aplikacji płatnych. W tym celu stworzono parser stron www ze sklepu Google Play oraz wykorzystano moduł pobierania uprawnień z algorytmu różnicowej analizy uprawnień. Analizowane aplikacje parsowane były z polskiego adresu IP, co ma wpływ na listę najpopularniejszych aplikacji prezentowaną w sklepie.

Wykonano również analizę 552-óch aplikacji 356-ciu gier z nieoficjalnego chińskiego sklepu GFAN [38]. Aby można było dokonać oceny zagrożenia za pomocą zaimplementowanych algorytmów badano tylko te próbki, które miały swoje odpowiedniki w sklepie Google Play.

Poniżej przedstawiono najciekawsze statystyki i wnioski. Pierwszy punkt zawiera porównanie liczby uprawnień, następnie przedstawiono wyniki analizy zagrożenia uzyskane za pomocą zaimplementowanych wcześniej algorytmów.

7.1. Liczba uprawnień

Sklep Google Play zawiera 44 kategorie aplikacji. Każda aplikacja jest przypisana do co najmniej jednej kategorii. Analizując wyniki warto pamiętać, że kategorie często agregują bardzo różne aplikacje. Nazwy kategorii pochodzą z polskojęzycznej wersji sklepu [47].

W tabeli 10 poniżej pokazano 5 kategorii darmowych aplikacji, które wykorzystują najwięcej uprawnień, w tabeli 11 znajdują się kategorie aplikacji płatnych które wykorzystują najwięcej uprawnień. Interesująca jest duża rozbieżność liczbie deklarowanych uprawnień oraz fakt, że to w aplikacje bezpłatne wymagają więcej uprawnień, nawet w tych samych kategoriach. Aplikacje płatne oferują zwykle więcej funkcji niż bezpłatne odpowiedniki czy wersje demo, powinny więc potrzebować większej liczby uprawnień. Możliwe, że aplikacje darmowe częściej korzystają z dostępu do Internetu w celu wyświetlenia reklam. Innym wyjaśnieniem może być fakt, że wiele popularnych aplikacji z wymienionych kategorii nie posiada wersji płatnych a do działania wersji bezpłatnej wymaga dużej liczby uprawnień.

We wszystkich przytaczanych kategoriach uwagę zwraca maksymalna liczba uprawnień, jest ona kilkukrotnie wyższa od średniej i często oznacza, że aplikacja deklaruje większość przewidzianych uprawnień.

Nazwa kategorii	Średnia liczba uprawnień	Mediana	Maksymalna liczba uprawnień
Widżety	14,20	11	61
Komunikacja	13,74	10	67
Produktywność	11,21	9	84
Społeczności	11,16	9	57
Dla firm	11,09	8	84

Tabela 10. Kategorie bezpłatnych aplikacji z największą średnią liczbą uprawnień.

Nazwa kategorii	Średnia liczba uprawnień	Mediana	Maksymalna liczba uprawnień
Widżety	10.28	9	59
Komunikacja	9.33	7	57
Gry akcji	7,94	8	17
Podróże i informacje lokalne	7,93	7	30
Gry Przygodowe	7,71	8	18

Tabela 11. Kategorie płatnych aplikacji z największą średnią liczbą uprawnień.

Tabele 12 i 13 zawierają kategorie w których aplikacje wymagają najmniejszej liczby uprawnień. W tych kategoriach aplikacje zawierają całą treść dla użytkownika, nie wymagają dostępu do sensorów czy niestandardowych interfejsów programistycznych, co tłumaczy niewielkie liczby deklarowanych uprawnień.

Nazwa kategorii	Średnia liczba uprawnień	Mediana	Maksymalna liczba uprawnień
Biblioteki i wersje demo	4,87	4	44
Edukacja	4,97	4	30
Gry edukacyjne	5.02	4	18
Gry planszowe	5.66	5	24
Medycyna	5.67	5	20

Tabela 12. Kategorie bezpłatnych aplikacji z najmniejszą średnią liczbą uprawnień.

Nazwa kategorii	Średnia liczba uprawnień	Mediana	Maksymalna liczba uprawnień
Personalizacja	2,77	0	36
Gry karciane	3,08	2	23
Komiksy	3,31	3	15
Biblioteki i wersje demo	3,33	1	19
Gry słowne	3,71	3	16

Tabela 13. Kategorie płatnych aplikacji z najmniejszą średnią liczbą uprawnień.

W tabeli 14 porównano liczbę uprawnień aplikacji ze sklepu Google Play i z nieoficjalnego sklepu z aplikacjami GFAN[38]. Bezpłatne aplikacje i gry z oficjalnego sklepu są zgrupowane pod nazwą „Google darmowe", etykieta „Google płatne" zawiera wszystkie aplikacje i gry płatne. Wyniki ze sklepu GFAN są podzielone na aplikacje i gry, z których każda zawiera zarówno programy płatne jak i bezpłatne. Taki podział utrudnia nieco analizę, jednak można wyciągnąć kilka istotnych wniosków. Przede wszystkim widoczna jest bardzo duża dysproporcja w liczbie deklarowanych uprawnień – aplikacje ze sklepu GFAN deklarują ich ponad dwukrotnie więcej.

	Średnia	Mediana	Percentyl 90%	Maksymalna liczba uprawnień
GFAN aplikacje	22.46	21	39	89
GFAN gry	12.92	11	22	69
Google darmowe	7.89	6	15	85
Google płatne	5.34	4	12	59

Tabela 14. Porównanie liczby aplikacji w sklepie Google Play i GFAN.

7.2. Analiza zagrożenia

W tym punkcie przedstawione zostaną wyniki analizy zagrożeń wykonane za pomocą opisanych wcześniej algorytmów. Parametry algorytmów były takie same jak we wcześniejszych analizach. Aplikacje uznawano za niebezpieczne gdy punktowy wynik zagrożenia wynosił ponad 4000.

W tabeli 15 znajdują się kategorie ze sklepu Google Play dla których średnie wartości zagrożenia aplikacji bezpłatnych były najwyższe, tabela 16 zawiera kategorie dla których płatne aplikacje miały najwyższe wartości zagrożenia. Za bardziej niebezpieczne uznane były aplikacje bezpłatne co może wynikać z tego, że rzeczywiście mogą zawierać programy złośliwe. Aplikacje płatne najprawdopodobniej nie zawierają

aplikacji stworzonych jako złośliwe, ponieważ zostaną zainstalowane na mniejszej liczbie urządzeń.

Nazwa kategorii	Średnia wartość zagrożenia
Dla firm	2734,3
Komunikacja	2505,0
Zakupy	2500,6
Transport	2344,7
Widżety	2249,5

Tabela 15. Kategorie aplikacji darmowych z najwyższą średnią wartością zagrożenia.

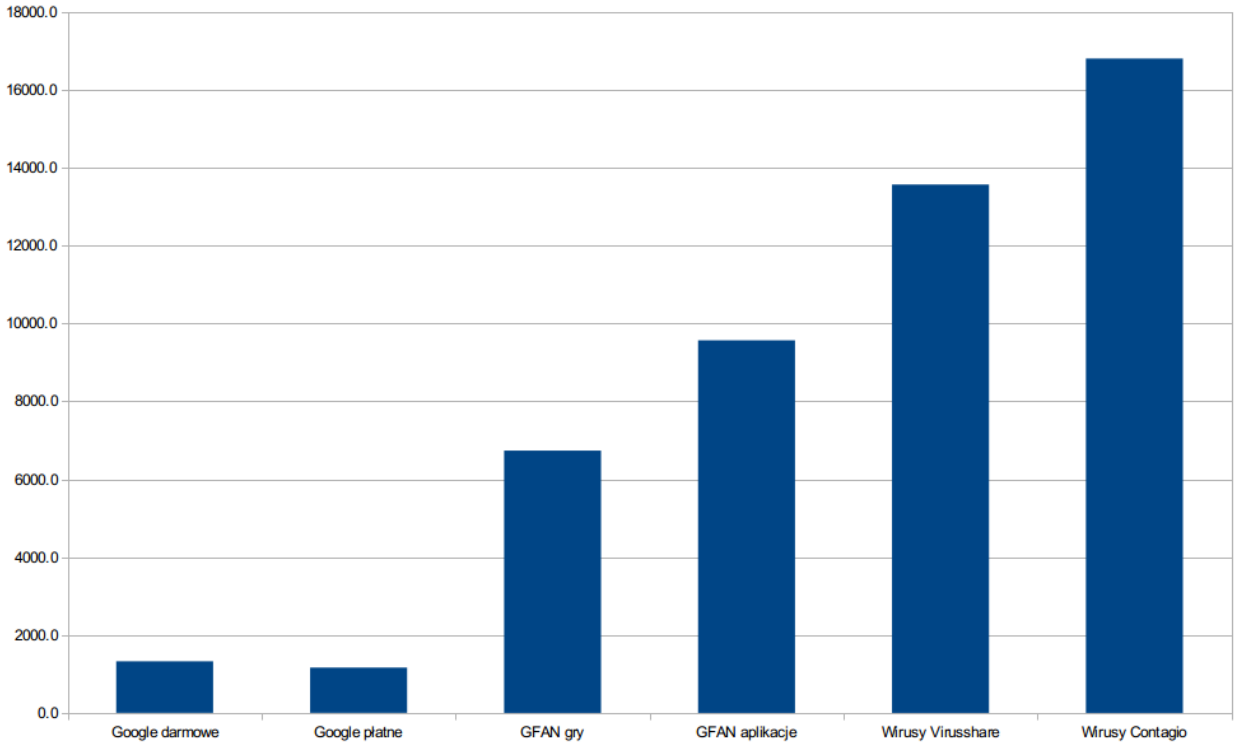
Nazwa kategorii	Średnia wartość zagrożenia
Komunikacja	2486,2
Społecznościowe	2390,3
Widżety	2307,3
Podróże	2246,8
Transport	2139,8

Tabela 16. Kategorie aplikacji płatnych z najwyższą średnią wartością zagrożenia.

Wykonano również analizę zagrożenia dla aplikacji ze sklepu GFAN. W tabeli 17 znajdują się średnie wartości punktowe zagrożenia dla aplikacji pochodzących z różnych źródeł. Grupowanie wykonano jak w punkcie 7.1, dla porównania zamieszczono wyniki analizy aplikacji złośliwych które wykorzystano w rozdziale 6 do analizy skuteczności oceny zagrożenia. Średnie wartości zagrożenia dla aplikacji ze sklepu GFAN wyglądają są wysokie, kilkukrotnie wyższe niż w przypadku aplikacji Google Play, ale nadal istotnie niższe niż w przypadku potwierdzonych wirusów. Wykres 5 przedstawia wizualizację wartości z tabeli 17. Zastosowano skalę liniową.

Źródło aplikacji	Średnia wartość zagrożenia
Google darmowe	1323,4
Google płatne	1162,4
GFAN gry	6734,7
GFAN aplikacje	9565,8
Wirusy Virusshare	13568,2
Wirusy Contagio	16804,6

Tabela 17. Porównanie średnich wartości zagrożenie.



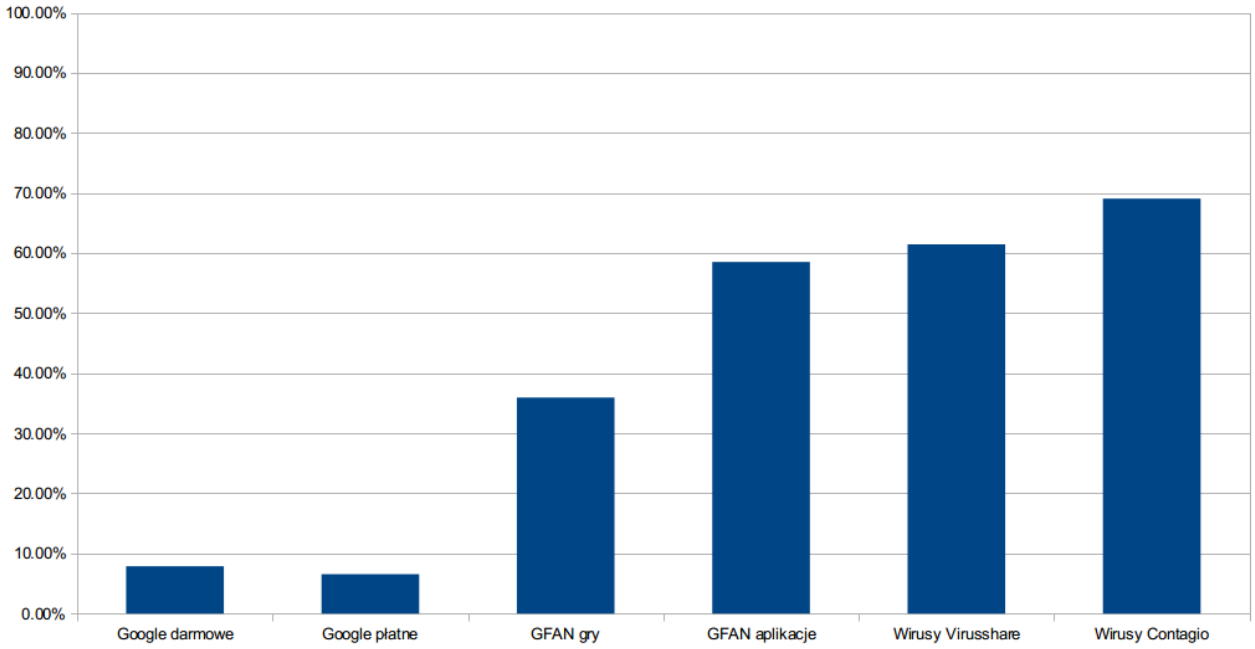
Wykres 5. Porównanie średnich wartości zagrożenie.

W kolejnej analizie sprawdzono jak wartości punktowe przekładają się na klasyfikacje zagrożenia dla aplikacji ze sklepu GFAN. Tabela 18 zawiera odsetek aplikacji które zostały uznane za niebezpieczne (etykieta dangerous). Okazuje się że pomimo widocznej różnicy w wartościach punktowych aplikacje ze sklepu GFAN prawie są tak samo często uznawane za niebezpieczne jak wirusy z bazy Virusshare. Za nieco bezpieczniejsze uznano gry ze sklepu GFAN Wykres 6 przedstawia wizualizację wartości z tabeli 18. Zastosowano skalę liniową.

Źródło aplikacji	Odsetek aplikacji groźnych
Google darmowe	7.88%
Google płatne	6.55%
GFAN gry	35.96%
GFAN aplikacje	58.51%
Wirusy Virusshare	61.44%
Wirusy Contagio	69.05%

Tabela 18. Porównanie liczby aplikacji niebezpiecznych.

.



Wykres 6. Porównanie liczby aplikacji niebezpiecznych.

7.3. Podsumowanie

W tym rozdziale przedstawiono wyniki analizy liczby uprawnień i poziomu zagrożenia aplikacji z sklepu Google Play oraz nieoficjalnego sklepu GFAN. Zauważono istotne różnice w liczbie uprawnień w poszczególnych kategoriach aplikacji Google Play.

Bardzo duża liczba uprawnień aplikacji ze sklepu GFAN jest trudna do wyjaśnienia. Może wynikać z różnic kulturowych. Możliwe, że w dalekiej Azji niewielu użytkowników przejmuje własną prywatnością albo nie jest świadoma zagrożeń.

Gdy aplikacje ze sklepu GFAN analizowane są przez algorytmy różnicowej analizy uprawnień klasyfikowane często klasyfikowane groźne. Może to być spowodowane rzeczywisty zagrożeniem, albo luką w opracowanej metodzie analizy. Jeżeli sklep Google Play nieprawidłowo podpowiada aplikacje podobne dla niezbyt popularnych programów w języku chińskim to wynik analizy jest mocno zaburzony. Mimo ewentualnego błędu wyniki powinny być niepokojące dla użytkowników sklepu GFAN.

8. Podsumowanie i wnioski

Celem pracy była analiza była analiza dostępnych mechanizmów bezpieczeństwa, ocena zagrożeń oraz wykonanie nowego typu oprogramowania analizującego niebezpieczne aplikacje systemu Android.

W ramach pracy powstało obszerne opracowanie zagadnień bezpieczeństwa systemu Android obejmujące mechanizmy wbudowane w system i proces dystrybucji aplikacji. Przeanalizowano skuteczność zewnętrznych aplikacji antywirusowych oraz przedstawiono akademickie rozwiązania dotyczące wykrywania zagrożeń.

Zaimplementowana została platforma do analizy aplikacji Android wykorzystująca analizę statyczną uprawnień aplikacji oraz dynamiczne porównywanie z uprawnieniami innych aplikacji o podobnych funkcjach.

Zmierzona skuteczność analizy zagrożenia jest porównywalna z najnowszymi algorytmami akademickimi dostępnymi w literaturze przedmiotu.

Zaproponowano dalszy rozwój platformy do analizy zagrożeń poprzez implementacji nowych algorytmów albo wykorzystanie optymalizacji globalnej i grupowego uczenie maszynowego (ang. ensemble learning).

Przedstawiono też analizę bezpieczeństwa popularnych aplikacji ze sklepu Google Play oraz nieoficjalnego, chińskiego sklepu GFAN. Analiza wykazała drastyczne różnice w liczbie uprawnień i potencjalnym poziomie zagrożenia na niekorzyść aplikacji ze sklepu GFAN.

Literatura

[1] F-Secure Corpor**a**tion*, H1 2014 Threat Report*. https://www.f-secure.com/documents/996508/1030743/Threat_Report_H1_2014.pdf dostęp 18.01.2015.

[2] International Data Corporation, *Smart-phone OS Market Share, Q3 2014* http://www.idc.com/prodserv/smartphone-os-market-share.jsp dostęp 18.01.2015.

[3] Kaspersky Lab, *Mobile cyber-threats: a joint study by Kaspersky Lab and INTERPOL,* http://media.kaspersky.co m/pdf/Kaspersky-Lab-KSN-Report-mobile-cyberthreats-web.pdf dostęp 17.01.2015.

[4] *Security | Android Developer*, dokumen-tacja systemu Android. https://source.android.com/devices/tech/security, dostęp 10.01.2015.

[5] V. N. Cooper, H. Shahriar, H. M. Haddad. *A Survey of Android Malware Characterisitics and Mitigation Tech-niques.* 2014 11th International Con-ference on Information Technology: New Generations.

[6] P. Faruki et al. *Android Security: A Survey of Issues, Malware Penetration and Defenses*, Communications surveys and tutorials, IEEE, wolumin. PP, numer. 99, grudzień 2014.

[7] *Exercising Our Remote Application Removal Feature,* http://android-developers.blogspot.com/2010/06/exercising-our-remote-application.html, dostęp 10.01.2015.

[8] *Dissecting the Android Bouncer*, https://jon.oberheide.org/blog/2012/06/21/dissecting-the-android-bouncer/, dostęp 10.01.2015.

[9] *Android Operating System Statistics – AppBrain*, http://www.appbrain.com/stats/, dostęp 11.01.2015.

[10] N.Penning, M. Hoffman, J. Nikolai, Y. Wang Mobile, *Malware Security Challeges and Cloud-Based Detection,* IEEE International Conference on Collaboration Technologies and Systems (CTS) , Maj 2014, strony 181-188.

[11] R. Raveendranath et al. *Android Malware Attacks and Countermeasures: Current and Future Directions,* IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), lipiec 2014, strony 137-143.

[12] Y. Zhou, X. Jiang, *Dissecting Android Malware: Characterization and Evo-lution*, IEEE Symposium on Security and Privacy (SP), Maj 2012, strony 95-109.

[13] T. Vidas, N. Christin, *Evading Android Runtime Analysis via Sandbox Detection, ACM,* ASIA CCS '14 Proceedings of the 9th ACM symposium on Information, computer and communications secu-rity, czerwiec 2014.

[14] M. L. Polla, F Martinelli, D. Sgandurra, *A Survey on Security for Mobile Devices*, IEEE Communications Surveys and Tutorials, vol. 15, no. 1, 2013.

[15] X. Jiang, *An Evaluation of the Application ("App") Verification Service in Android 4.2,* http://www.csc.ncsu .edu/faculty/jiang/appverify/, dostęp 20.01.2015.

[16] *Dekompilator aplikacji Android-apktool*, https://code.google.com/p/android-apktool/, dostęp 15.01.2015.

[17] *Dekompilator kodu bajtowegodex2jar* https://code.google.com/p/dex2jar/, dostęp 15.01.2015.

[18] *Analizator aplikacji adroguard,* https://code.google.com/p/androguard, dostęp 15.01.2015.

[19] *App Review* https://developer.apple .com/app-store/review, dostęp 20.01. 2015.

[20] R. Fedler, J. Schütte, M. Kulicke, *On The Effectiveness Of Malware Protection On Android An Evaluation Of Android Antivirus Apps*, Fraunhofer AISEC, kwiecień 2013.

[21] Z. Aung, W. Zaw, *Permission-Based Android Malware Detection*, International journal of scientific & technology research tom 2, wydanie 3, Marzec 2013.

[22] *Mobile anti-virus not needed: Google* http://www.smh.com.au/digital-life/consumer-security/mobile-antivirus-not-needed-google-20140702-zsthl.html, dostęp 01.01.2015.

[23] R. Fedler, et al, *Android OS Securty Risks and Limitations* http://www.aisec.fraunhofer.de/content/dam/aisec/Dokumente/Publikationen/Studien_TechReports/deutsch/AISEC-TR-2012-001-Android-OS-Security.pdf, dostęp 02.02.2015.

[24] Dong-Jie Wu, *DroidMat: Android Malware Detection through Manifest and API Calls Tracing*, 2012 Seventh Asia Joint Conference on Information Security.

[25] A. Abu Samra, *Analysis of Clustering Technique in Android Malware Detection*, IEEE, Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013.

[26] M. Ghorbanzadeh et al. *A Neural Network Approach to Category Validation of Android Applications*, IEEE Computing, Networking and Communications (ICNC) 2013.

[27] N. Peiravian, Z. Xingquan, *Machine Learning for Android Malware Detection Using Permission and API Calls*, IEEE, Tools with Artificial Intelligence (ICTAI) 2013.

[28] L. Batyuk et al. Using Static Analysis for Automatic Assessment and Mitigation of Unwanted and Malicious Activities Within Android Applications, IEEE, Malicious and Unwanted Software (MALWARE) 2013.

[29] F. Al-Qershi, et al. *Android vs. iOS: The security battle,* IEEE, Computer Applications and Information Systems (WCCAIS), 2014.

[30] M.S. Ahmad et al. *Comparison between android and iOS Operating System in terms of security*, IEEE, Information Technology in Asia (CITA) 2013.

[31] *Android Torch App with over 50m Downloads Silently Sent User Location and Device Data to Advertiser,* http://www.theguardian.com/technology/2013/dec/06/android-app-50m-downloads-sent-data-advertisers dostęp 03 czerwca 2014.

[32] *Plik AndroidManifest.xml w repozytorium kodu źródłowego systemu Android*, https://github.com/android/platform_frameworks_base/blob/master/core/res/AndroidManifest.xml , dostęp 02.02.2015.

[33] Blog *contagiodump*, http://contagiodump.blogspot.com, dostęp 03.02.2015.

[34] Baza wirusów VirusShare, http://virusshare.com, dostęp 04.02.2015.

[35] Google Play Crawler JAVA API, https://github.com/Akdeniz/google-play-crawler, dostęp 02.01.2015.

[36] Dokumentacja pakietów system Android, http://developer.android.com/reference/packages.html, dostęp 02.01.2015.

[37] Wei Wang, et al. *Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection*, Information Forensics and Security, IEEE, wolumin 9, wydanie 11.

[38] Sklep z aplikacjami GFAN, http://apk.gfan.com, dostęp 02.02.2015.

[39] Dokumentacja Protocol Buffers, https://developers.google.com/protocol-buffers, dostęp 15.01.2015.

[40] Min Zheng, Patrick P.C. Lee, John C.S. Lui, *ADAM: An Automatic and Extensible Platform to Stress Test Android Anti-Virus Systems*, Proceedings of the 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'12), Heraklion, Grecja, Lipiec 26-27, 2012.

[41] Y. Zhou, X. Jiang, *Android Malware Genome Project*, http://www.malgenomeproject. org, dostęp 10.01.2015.

[42] J. Xuxian, Z. Yajin, *Android Malware*, SpringerBriefs in Computer Science 2013.

[43] S. Liang, X. Du, *Permission-combination-based scheme for Android mobile malware detection*, IEEE International Conference on Comunications, 10-14 czerwiec 2014.

[44] Aplikacja *Show Java - A Java Decompiler*, https://play.google.com/store/apps/details?id=com.njlabs.showjava&hl=pl, dostęp 15.02.2015.

[45] Aplikacja *Dexplorer,* https://play.google.com/store/apps/details?id=com.dexplorer, dostęp 15.02.2015.

[46] Aplikacja *JaDX - Decompiler* https://play.google.com/store/apps/details?id=ua.naiksoftware.jadx, dostęp 15.02.2015.

[47] Sklep z aplikacjami Google Play w polskiej wersji językowej, https://play.google.com/store/apps?hl=pl, dostęp 05.02.2015.

[48] Google Cloud Messaging for Android https://developer.android.com/google/gcm/index.html

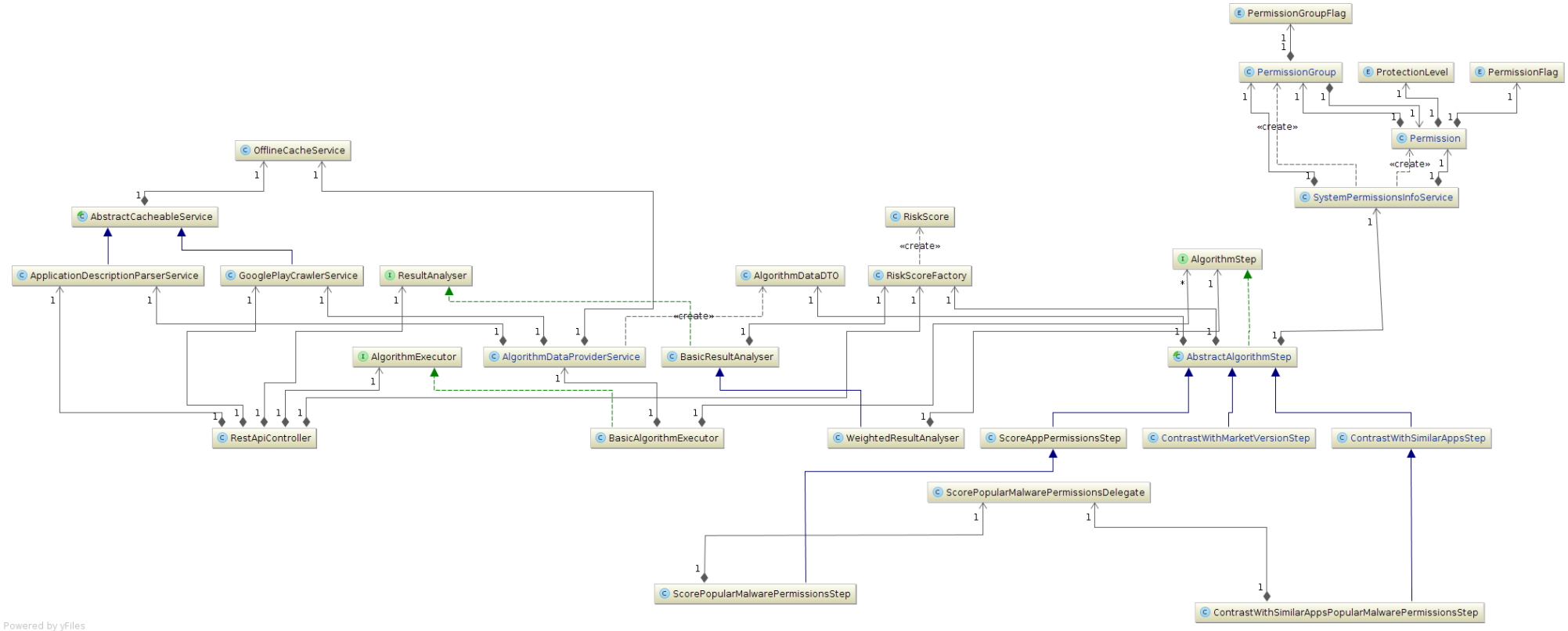
Załącznik 1. Listing parametrów aplikacji

```
# Parametry wymagane przez Google Play Crawler JAVA API, login i hasło są podmienione
crawler.email=NAZWA_KONTA@gmail.com
crawler.password=HASLO_DO_KONTA
crawler.maxRetriesWhenDownloadingAppPermission=0
crawler.sleepTimeOnFailedDownloadInMilliseconds=30000
crawler.sleepTimeOnSuccessfulDownloadInMilliseconds=1000

#nazwy poszczególnych poziomów zagrożenia
riskScore.names=LOW,LOWER,NORMA,HIGH,HIGHER,DANGEROUS

#zakresy punktowe poszczególnych poziomó zagrożenia
riskScore.points=200,500,1000,2000,4000

#parametry algorytmu statycznej analizy uprawnień
scoreAppPermissions.permission.flag.costs.money=500
scoreAppPermissions.permission.protectionLevel.normal=1
scoreAppPermissions.permission.protectionLevel.dangerous=20
scoreAppPermissions.permission.protectionLevel.signature=0
scoreAppPermissions.permission.protectionLevel.signatureOrSystem=0
scoreAppPermissionsStep.permissionGroup.flag.personalInfo=20

# parametry algorytmu statycznej analizy podejrzanych uprawnień
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.SEND_SMS=300
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.RECEIVE_SMS=300
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.READ_PHONE_STATE=100
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.ACCESS_WIFI_STATE=100
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.READ_SMS=300
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.RECEIVE_BOOT_COMPLETED=150
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.WRITE_SMS=300
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.WRITE_CONTACTS=300
ScorePopularMalwarePermissionsStep.riskyPermission.android.permission.WRITE_APN_SETTINGS=300

# parametry algorytmu porównującego uprawnienia z tą samą aplikację ze sklepu Google Play
contrastWithMarket.newPermission.flag.costs.money=4000
contrastWithMarket.newPermission.protectionLevel.normal=500
contrastWithMarket.newPermission.protectionLevel.dangerous=1500
contrastWithMarket.newPermission.protectionLevel.signature=500
contrastWithMarket.newPermission.protectionLevel.signatureOrSystem=500
contrastWithMarket.newPermission.permissionGroup.flag.personalInfo=1500
contrastWithMarket.appNotFoundInMarket=3000

# parametry algorytmu porównującego uprawnienia z podobnymi aplikacjami ze sklepu Google Play
contrastWithSimilarAppsStep.uniquePermissionInSimilarApps.flag.costs.money=3000
contrastWithSimilarAppsStep.uniquePermissionInSimilarApps.protectionLevel.normal=50
contrastWithSimilarAppsStep.uniquePermissionInSimilarApps.protectionLevel.dangerous=200
contrastWithSimilarAppsStep.uniquePermissionInSimilarApps.protectionLevel.signature=0
contrastWithSimilarAppsStep.uniquePermissionInSimilarApps.protectionLevel.signatureOrSystem=0
contrastWithSimilarAppsStep.uniquePermissionInSimilarApps.permissionGroup.flag.personalInfo=300
contrastWithSimilarAppsStep.rarePermissionBracket=0.25
contrastWithSimilarAppsStep.uniquePermissionBracket=0.1
contrastWithSimilarAppsStep.uniquePermissionPenaltyRatio=2.0

# parametry algorytmu porównującego podejrzane uprawnienia z podobnymi aplikacjami ze sklepu
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.SEND_SMS=1000
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.READ_PHONE_STATE=300
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.ACCESS_WIFI_STATE=300
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.READ_SMS=1000
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.RECEIVE_BOOT_
COMPLETED=600
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.WRITE_SMS=1000
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.RECEIVE_SMS=1000
```

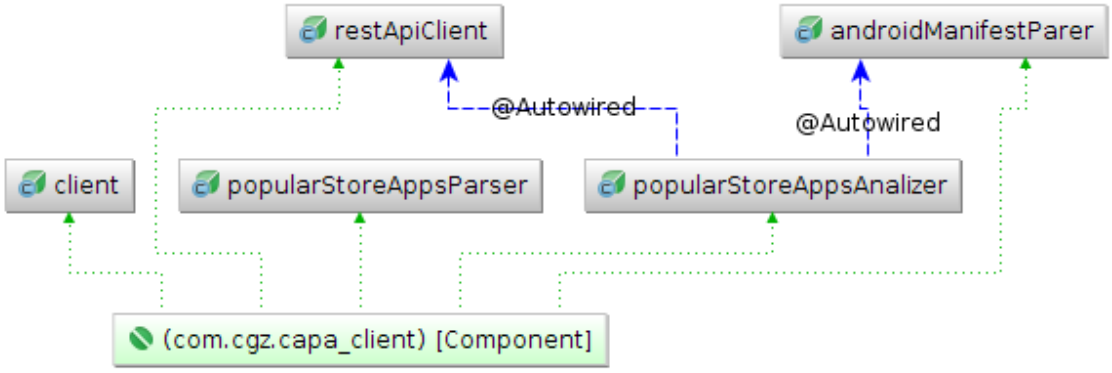
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.WRITE_CONTACTS=1000
ContrastWithSimilarAppsPopularMalwarePermissionsStep.riskyPermission.android.permission.WRITE_APN_SETTINGS=1000

# przykładow parametry analizatora wyników sumującego wyniki z różnymi wagami
WeightedResultAnalyser.weight.scoreAppPermissions=0.5
WeightedResultAnalyser.weight.contrastWithMarket=1
WeightedResultAnalyser.weight.contrastWithSimilarAppsStep=1.2
WeightedResultAnalyser.weight.ScorePopularMalwarePermissionsStep=.6
WeightedResultAnalyser.weight.ContrastWithSimilarAppsPopularMalwarePermissionsStep=1.2

Załącznik 2. Diagram klas programu analizatora



Załącznik 3. Schemat komponentów Spring programu klienta.



Załącznik 4. Lista uprawnień systemu Android

Załącznik zawiera tabelę ze wszystkimi uprawnieniami systemu Android dostępnymi dla aplikacji zewnętrznych w wersji 21 API systemu. Nie zawarto tu uprawnień dostępnych we wcześniejszych wersjach systemu, które zostały później zastrzeżone dla aplikacji systemowych i nie są dostępne w najnowszej wersji.

Kolejne kolumny zawierają skróconą nazwę uprawnienia, poziom ochrony (zagrożenia), wartość flagi informującej o dostępie do danych osobistych użytkownika oraz krótki opis przedmiotu uprawnienia. Opracowano na podstawie [4] oraz [32].

Nazwa	**Zagrożenie**	**Dane prywa-tne**	**Uprawnienie pozwala aplikacji na:**
ACCESS_COARSE_ LOCATION	**GROŹNY**	**TAK**	Określanie przybliżonej lokalizacji.
ACCESS_FINE_LOCATION	**GROŹNY**	**TAK**	Określanie dokładnej lokalizacji dzięki sygnałowi GPS.
ACCESS_LOCATION_ EXTRA_ COMMANDS	normalny		Dostęp do dodatkowych poleceń dostawcy informacji o lokalizacji. Aplikacje z tym uprawnieniem mogą wpływać na działanie urządzenia GPS lub innych źródeł lokalizacji.
ACCESS_MOCK_ LOCATION	**GROŹNY**		Tworzenie pozorowanych źródeł lokalizacji dla potrzeb testów lub instalacji nowego dostawcy informacji o lokalizacji.
ACCESS_NETWORK_ STATE	normalny		Dostęp do informacji o połączeniach sieciowych – np. o dostępnych i połączonych sieciach.
ACCESS_WIFI_STATE	normalny		Dostęp do informacji o połączeniach Wi-Fi – np. na sprawdzenie, czy obsługa Wi-Fi jest włączona, oraz odczytanie nazw podłączonych urządzeń Wi-Fi.
ACCESS_WIMAX_STATE	normalny		Określanie, czy obsługa WiMAX jest włączona, oraz uzyskać informacje o wszystkich podłączonych sieciach WiMAX.
ADD_VOICEMAIL	**GROŹNY**	**TAK**	Dodawanie wiadomości do skrzynki odbiorczej poczty głosowej.
AUTHENTICATE_ ACCOUNTS	**GROŹNY**	**TAK**	Korzystanie z funkcji uwierzytelniania konta usługi AccountManager, w tym funkcji tworzenia kont oraz pobierania i ustawiania ich haseł.
BATTERY_STATS	normalny		Odczytywanie bieżących danych niskiego poziomu o wykorzystaniu baterii.
BLUETOOTH_ADMIN	**GROŹNY**		Konfigurowanie lokalnego telefonu z funkcją Bluetooth oraz na wykrywanie urządzeń zdalnych i parowanie z nimi.
BLUETOOTH	**GROŹNY**		Dostęp do konfiguracji Bluetooth na telefonie oraz na nawiązywanie i akceptowanie połączeń ze sparowanych urządzeń.
BODY_SENSORS	normalny	**TAK**	Dostęp do danych z czujników na ciele użytkownika.
BROADCAST_STICKY	normalny		Wysyłanie transmisji trwałych, które pozostają aktywne po zakończeniu połączenia.
CALL_PHONE	**GROŹNY**	**TAK**	Dzwonienie pod numery telefonów bez wiedzy użytkownika. **Powoduje koszty.**
CAMERA	**GROŹNY**	**TAK**	Dostęp do aparatu.
CHANGE_ CONFIGURATION	normalny		Zmianę bieżącej konfiguracji, na przykład regionu lub ogólnego rozmiaru czcionki.
CHANGE_NETWORK_ STATE	normalny		Zmianę stanu łączności sieciowej.

CHANGE_WIFI_ MULTICAST_ STATE	**GROŹNY**		Odbieranie pakietów wysyłanych przez sieć Wi-Fi przy użyciu adresów połączeń grupowych.
CHANGE_WIFI_STATE	**GROŹNY**		Nawiązywanie i kończenie połączeń z punktami dostępowymi Wi-Fi oraz na zmienianie konfiguracji sieci Wi-Fi w urządzeniu.
CHANGE_WIMAX_STATE	**GROŹNY**		Nawiązywanie i kończenie połączeń z sieciami WiMAX w telefonie.
CLEAR_APP_CACHE	**GROŹNY**		Usuwanie pamięci podręcznej innych aplikacji.
DISABLE_KEYGUARD	**GROŹNY**	**TAK**	Wyłączanie blokady klawiatury i wszystkich związanych z tym haseł zabezpieczających.
EXPAND_STATUS_BAR	normalny		Rozwijanie lub zwijanie paska stanu.
FLASHLIGHT	normalny		Sterowanie latarką.
GET_ACCOUNTS	normalny	**TAK**	Uzyskanie listy kont zapisanych w telefonie.
GET_PACKAGE_SIZE	normalny		Sprawdzenie ilości miejsca zajmowanej przez każdą aplikację.
GET_TASKS	normalny		Pobieranie informacji o aktualnie i niedawno działających zadaniach.
INSTALL_SHORTCUT	GROŹNY		Dodawanie skrótów na ekranie głównym bez interwencji użytkownika.
INTERACT_ACROSS_ USERS	normalny		Zezwala na wszystkie możliwe interakcje między użytkownikami.
INTERNET	GROŹNY		Dostęp do Internetu.
KILL_BACKGROUND_ PROCESSES	normalny		Kończenie procesów innych aplikacji działających w tle.
MANAGE_ACCOUNTS	GROŹNY	**TAK**	Wykonywanie takich operacji, jak dodawanie i usuwanie kont, a także usuwanie ich haseł.
MODIFY_AUDIO_ SETTINGS	normalny		Modyfikowanie globalnych ustawień dźwięku.
NFC	GROŹNY		Dostęp do komunikacji NFC.
PERSISTENT_ACTIVITY	normalny		Trwałe zapisywanie swoich fragmentów w pamięci.
PROCESS_OUTGOING_ CALLS	GROŹNY	**TAK**	Przetwarzanie połączeń wychodzących i zmianę wybieranego numeru.
READ_CALENDAR	GROŹNY	**TAK**	Odczytywanie wszystkich wydarzeń w kalendarzu zapisanych na telefonie, w tym pochodzących od znajomych i współpracowników.
READ_CALL_LOG	GROŹNY	**TAK**	Odczyt rejestru połączeń w telefonie, w tym danych o połączeniach przychodzących i wychodzących.
READ_CELL_BROADCASTS	GROŹNY	**TAK**	Odczyt wiadomości z sieci komórkowej odebranych na urządzeniu.
READ_CONTACTS	GROŹNY	**TAK**	Odczyt danych o kontaktach zapisanych na telefonie.
READ_EXTERNAL_ STORAGE	normalny	**TAK**	Odczyt pamięci zewnętrznej, np. karty SD.
READ_HISTORY_ BOOKMARKS	GROŹNY	**TAK**	Odczyt wszystkich URL-i odwiedzonych przez przeglądarkę oraz wszystkich zakładek w przeglądarce.
READ_PHONE_STATE	GROŹNY	**TAK**	Dostęp do funkcji telefonicznych urządzenia. Aplikacja z tym uprawnieniem może odczytać numer telefonu i identyfikator urządzenia, sprawdzić, czy połączenie jest aktywne, oraz poznać numer, z którym jest nawiązane połączenie
READ_PROFILE	GROŹNY	**TAK**	Odczyt osobistych informacji przechowywanych w profilu użytkownika na urządzeniu (np. imienia i nazwiska lub adresu).
READ_SMS	GROŹNY	**TAK**	Odczyt SMS-ów zapisanych na telefonie lub na karcie SIM.
READ_SOCIAL_STREAM	GROŹNY	**TAK**	Odczyt i synchronizację informacji publikowanych przez użytkownika i jego znajomych w sieciach społecznościowych.
READ_SYNC_SETTINGS	normalny		Odczyt ustawień synchronizacji konta.
READ_SYNC_STATS	normalny		Czytanie statystyk synchronizacji konta.
READ_USER_DICTIONARY	GROŹNY	**TAK**	Odczyt słownika użytkownika.
READ_VOICEMAIL	normalny	**TAK**	Odczyt poczty głosowej.
RECEIVE_BOOT_ COMPLETED	normalny		Uruchamianie natychmiast po zakończeniu rozruchu systemu.
RECEIVE_MMS	GROŹNY	**TAK**	Odbieranie i przetwarzanie MMS-ów.

RECEIVE_SMS	GROŹNY	**TAK**	Odbieranie i przetwarzanie SMS-ów.
RECEIVE_WAP_PUSH	GROŹNY	**TAK**	Odbieranie i przetwarzanie wiadomości WAP.
RECORD_AUDIO	GROŹNY	**TAK**	Nagrywanie dźwięku przez mikrofon.
REORDER_TASKS	normalny		Przenoszenie zadań między tłem a pierwszym planem.
RESTART_PACKAGES	normalny		Kończenie zadań procesów działających w tle.
SEND_SMS	GROŹNY	**TAK**	Wysyłanie SMS-ów. **Powoduje koszty.**
SET_ALARM	normalny	**TAK**	Ustawienie alarmu w zainstalowanej aplikacji budzika.
SET_TIME_ZONE	normalny		Zmianę ustawienia strefy czasowej w telefonie
SET_WALLPAPER_HINTS	normalny		Ustawianie wskazówek dotyczących rozmiaru tapety systemu
SET_WALLPAPER	normalny		Ustawianie tapety systemu
SUBSCRIBED_FEEDS_READ	normalny		Pobieranie szczegółowych informacji na temat obecnie zsynchronizowanych kanałów
SUBSCRIBED_FEEDS_ WRITE	GROŹNY		Zmianę obecnie zsynchronizowanych kanałów. Złośliwe aplikacje mogą zmienić zsynchronizowane kanały
SYSTEM_ALERT_WINDOW	GROŹNY		Wyświetlanie elementów interfejsu nad innymi aplikacjami lub elementami ich interfejsu.
TRANSMIT_IR	normalny		Komunikację poprzez podczerwień.
UNINSTALL_SHORTCUT	GROŹNY		Usuwanie skróty z ekranu głównego bez interwencji użytkownika
USE_CREDENTIALS	GROŹNY	**TAK**	Wykorzystywanie tokenów uwierzytelniania z usługi Zarządcy Kont
USE_SIP	GROŹNY	**TAK**	Korzystanie z usługi SIP do nawiązywania/odbierania połączeń przez Internet.
VIBRATE	normalny		Sterowanie wibracjami
WAKE_LOCK	normalny		Zapobieganie przechodzeniu telefonu w tryb uśpienia
WRITE_CALENDAR	GROŹNY	**TAK**	Dodawanie zdarzeń do kalendarza.
WRITE_CALL_LOG	GROŹNY	**TAK**	Dodawanie wpisu do rejestru połączeń telefonu.
WRITE_CONTACTS	GROŹNY	**TAK**	Zapisywanie kontaktów.
WRITE_EXTERNAL_ STORAGE	GROŹNY	**TAK**	Zapis i odczyt pamięci zewnętrznej, np. karty SD.
WRITE_HISTORY_ BOOKMARKS	GROŹNY	**TAK**	Odczyt historii i zakładek przeglądarki zapisanych na telefonie.
WRITE_PROFILE	GROŹNY	**TAK**	Dodanie osobistych informacji przechowywanych w profilu użytkownika na urządzeniu (np. imienia i nazwiska lub adresu).
WRITE_SETTINGS	normalny		Zmienianie ustawień systemu.
WRITE_SMS	GROŹNY	**TAK**	Zapisywanie wiadomości SMS przechowywanych w telefonie lub na karcie SIM. Szkodliwe aplikacje mogą usunąć wiadomości
WRITE_SOCIAL_STREAM	GROŹNY	**TAK**	Wyświetlanie informacji publikowanych przez znajomych w sieciach społecznościowych. Aplikacja z tym uprawnieniem może tworzyć wiadomości, które wyglądają jakby pochodziły od znajomych.
WRITE_SYNC_SETTINGS	normalny		Modyfikowanie ustawień synchronizacji z kontem.
WRITE_USER_DICTIONARY	normalny		Zapis do słownika użytkownika.
WRITE_VOICEMAIL	normalny	**TAK**	Modyfikowanie i usuwanie wiadomości z poczty głosowej

Tabela 19. Lista uprawnień systemu Android.

Załącznik 5. Zawartość płyty CD

Do pracy została załączona płyta CD-ROM zawierająca:

- Treść niniejszej pracy w kilku popularnych formatach.
- Kod programu analizatora zagrożeń
- Kod programu klienta analizatora