МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПЗВО «МІЖНАРОДНИЙ КЛАСИЧНИЙ УНІВЕРСИТЕТ
імені ПИЛИПА ОРЛИКА»
Економіко-технологічний факультет
Кафедра інженерних технологій

Кваліфікаційна робота
на здобуття освітнього ступеня магістра
за освітньою програмою «Комп'ютерна інженерія»
зі спеціальності 123 «Комп'ютерна інженерія»
на тему: **«ІНФОРМАЦІЙНА БЕЗПЕКА ІНФОРМАЦІЙНИХ СИСТЕМ НА БАЗІ ПЛАТФОРМИ ANDROID**»

Виконав:
здобувач II курсу, групи КІ -20-24
Калинка Володимир Володимирович

Керівник:
к.т.н., доцент кафедри інженерних технологій
Гайша Олександр Олександрович

Миколаїв – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HDD	–	Hard Disk Drive
RBA	–	Remote Biometrical Assurance
SAML	–	Security Assertion Markup Language
SNMP	–	Simple Network Management Protocol
WAN	–	Wide Area Network
ЗІ	–	Захист інформації
ІКМ	–	Інформаційно-комунікаційні мережі
ІКС	–	Інформаційно-комунікаційні системи
ІКТ	–	Інформаційно-комунікаційні технології
ІТ	–	Інформаційні технології
ІТР	–	Інженерно-технічні робітники
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення
ПК	–	Персональний комп’ютер
СЗІ	–	Система захисту інформації
СКД	–	Система контролю доступу
ТЗІ		Технічний захист інформації

ВСТУП

Швидкий розвиток комп'ютерної техніки, стрімкий процес мініатюризації, широке поширення персоналізованих мультимедійних технологій – все це втілилося у тому, що на початку 2020-х років у кожного соціалізованого громадянина є в наявності мобільний телефон, причому у переважній більшості (особливо у людей молодого та середнього віку) мова йде конкретно про смартфони. Ці пристрої, на відміну від звичайних мобільних телефонів, призначені не лише для комунікації у голосовому режимі із іншими абонентами мобільних та стаціонарних мереж, а й можуть виконувати ще багато усіляких різноманітних функцій (звідси походить і назва «смартфони», оскільки, як відомо, англійською «smart» означає «розумний»).

Зважаючи на дуже високу складність сучасних смартфонів, насправді дуже важко швидко оцінити, яку конфіденційну інформацію несе у собі смартфон того, чи іншого користувача (особливо, беручи до уваги, що набір програмного забезпечення кожного пристрою може відрізнятися, залежно від особливостей та потреб його власника). Однак, існують і деякі універсальні дані, що потребують захисту. Це, наприклад, банківський рахунок, який може управлятися через мобільний телефон спеціальним додатком, із яких найбільш поширеним в Україні є Privat24.

Дуже часто користувач не вводить пароль кожного разу, коли хоче скористатися захищеними сервісами, а вмикає опцію «бути активним весь час» і тоді будь-хто (інший користувач, що випадково бере в руки смартфон, чи злочинний мобільний додаток типу вірусу чи трояну) зможе також отримати доступ до цих, нібито захищених, ресурсів.

Можна вигадувати і інші ситуації, коли під загрозою опиняються не просто персональні дані законного користувача смартфону, а його фінанси, репутація, особисте життя, а в деяких, особливо тяжких випадках, і особиста безпека та/або безпека близьких до нього людей. Відповідно, надзвичайно

актуальною є задача аналізу інформаційної безпеки «розумних» мобільних систем та вироблення заходів організаційного та технічного (ймовірно, програмного) характеру для нейтралізації знайдених загроз.

Враховуючи вищенаведене, можна сформулювати наступну мету дослідження: поліпшити безпеку інформації, що зберігається на смартфонах під управлінням ОС Android, шляхом проектування та реалізації відповідної системи захисту, чи її важливої частини (що нейтралізує деякі суттєві загрози).

Для досягнення поставленої мети необхідно вирішити наступні задачі дослідження:

- проаналізувати особливості ОС Android, важливі з точки зору питань кібербезпеки, виконати аналіз загроз системі;

- проаналізувати існуючі методи та засоби захисту мобільних систем на базі ОС Android;

- обрати загрози, що не перекриваються (або недостатньо перекриваються) існуючими засобами;

- спроектувати систему захисту для інформаційних систем на базі ОС Android, що нейтралізує загрозу (декілька загроз), обраних на попередньому кроці;

- здійснити програмну реалізацію окремих ланок спроектованої системи захисту;

- здійснити тестування розробленого програмного продукту;

- проаналізувати результати роботи системи захисту, зокрема, оцінити її ефективність та зробити висновки по роботі.

Методи дослідження, застосовані у даній роботі, знаходяться у галузі кібербезпеки.

Практичне значення роботи полягає у тому, що розроблену систему захисту можна застосовувати для реальних мобільних платформ, зведених на базі ОС Android.

Оскільки розроблене програмне забезпечення має універсальний характер, то в перспективі розроблені ідеї можуть бути поширені на споріднені операційні системи (iOS, Windows 8/10, тощо).

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Особливості будови ОС Android, суттєві з точки зору галузі кібербезпеки

Поняття операційної системи з'явилося у світі комп'ютерної техніки фактично із самого початку її широкого поширення. Одиничні вироби у вигляді унікальних електронно-обчислювальних машин (ЕОМ), або мікроконтролерних систем теоретично можуть працювати і без операційної системи, а тільки виконуючи у безперервному циклічному режимі набір підпрограм. За своєю суттю операційна система є надзвичайно складною службовою програмою (по суті – це найбільш яскравий приклад системного програмного забезпечення), яка надає необхідні для різних прикладних програм користувача можливості по взаємодії з апаратним забезпеченням комп'ютера (починаючи із самого простого, такого, як файлова система) та з самим користувачем шляхом використання пристроїв введення-виведення. Головною функцією ОС є надання прикладним програмам гарного уніфікованого програмного інтерфейсу, аби вони могли будувати власний інтерфейс користувача, коректно виконувати операції вводу-виводу, здійснювати комунікацію з іншими процесами та пристроями комп'ютера, одним словом усе, що не відноситься до бізнес-логіки програмного забезпечення і т.п.

Після появи та широкого поширення «розумних» мобільних телефонів – «смартфонів» - з'явилася можливість і для них розробляти власне програмне забезпечення (мобільні додатки), тому тут також, як і на ПК, ключовою складовою усієї програмно-апаратної платформи є застосована операційна система. На сьогоднішній день існує ряд операційних систем, що можуть бути встановлені на мобільні пристрої (наприклад, Windows 8, Windows CE, Symbian, Palm OS, BlackBerry OS і т.д.), однак число їхніх прихильників по всьому світу є дуже малим. Реально на сьогоднішній день

активно використовуються всього лише дві операційних системи для смартфонів: Android від Google та iOS від компанії Apple.

Ці дві системи є абсолютно несумісними, мають різну архітектуру і програмне забезпечення для них розробляється шляхом застосування різних підходів (та, відповідно, засобів розробки). Програмні продукти для Apple розміщуються в Apple iStore, додатки для Android розміщуються в Google Play Market. Додатки для iOS зводяться у середовищі XCode (яке до речі доступне тільки для OS X, тобто може завантажуватися лише на Mac-системах) мовою Swift або C++. У розробників для ОС Android більше різноманітних інструментів та мов програмування: починаючи від Java і закінчуючи C#. Беручи до уваги усі відмінності між двома цими ОС, розробникам слід визначитися із цільовою платформою і тут слід урахувати відносну поширеність цих продуктів на вітчизняному ринку. Слід відмітити, що, незважаючи на певне переважання «престижності» продуктів від компанії Apple, завдяки кращій ціновій політиці, більшій лінійці доступних продуктів різних фірм (в першу чергу, такого гіганта, як Samsung, але і інших виробників мобільних телефонів: HTC, Huawei, Meizu, і т.д.) можна із впевненістю констатувати, що ОС від Google використовується на значно більшій кількості пристроїв в Україні, ніж Apple (різниця, ймовірно, становить сотні відсотків, тобто «в рази»). Таким чином, якщо обирати між цими двома альтернативами, то більш доцільним на даний момент є створення програмних продуктів (в т.ч. і з галузі захисту інформації) саме для Android.

Історично ця ОС виникла на базі відкритого ядра ОС Linux та спеціально розробленої віртуальної машини Java (програмний продукт для виконання програм, написаних однойменною мовою Java та скомпільованих у проміжний байт-код; такий підхід дозволяє робити програми одночасно і досить швидкими у виконанні, і повністю кросплатформенними – головне, щоби для цільової платформи існувала Java-машина). У 2005 р. корпорація Google купила компанію Android, яка на той час знаходилася в процесі

розробки першої версії ОС, і доробивши її, випустила у 2008 році першу версію. В цілому еволюція цих систем показана на рис. 1.1, з якого видно, що в середньому Google видає по одній новій версії ОС на рік.

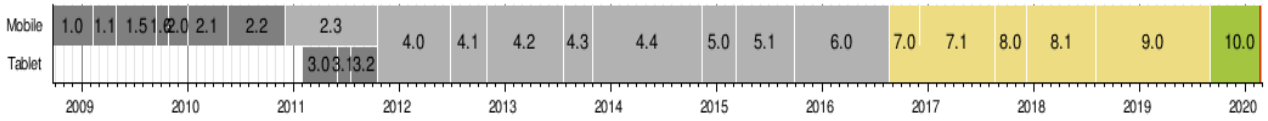

Рис. 1.1. Хронологія виходу версій ОС Android.

Використання віртуальної машини Java для виконання коду (фактично – у безпечному середовищі, вся небезпека якого обмежується можливостями, що надає саме це середовище), звичайно, значно підвищує загальну стійкість системи до зловмисних дій [6]. Так, значна частка IT-спеціалістів впевнені, що виконання коду у віртуальній машині (певній «пісочниці»), навіть за умови цілеспрямованої злочинності цього коду, не може завдати шкоди основній операційній системі та комп'ютеру в цілому. Зокрема, саме таким чином здійснюють перевірку підозрілих файлів на віруси та дослідження самих вірусів. При виході вірусу із під контролю фахівець просто витирає віртуальну машину та/або перезаписує поверх старої зараженої версії, нову чисту машину. Насправді, існують вразливості (та, відповідно, програми, що їх використовують – експлоїти), що дозволяють вийти за межі емульованого ПК і вразити реальну систему (як, наприклад, рішення від хакерської групи Vupen для віртуальної машини Xen, що дозволяє виконувати код на материнській ОС).

Однак, незважаючи на їх принципову наявність, такі випадки є поодинокими і все ж являються надто екзотичними, що говорить про підвищений рівень надійності систем, в яких виконання коду здійснюється на віртуальній машині (реалізація віртуальної машини Java для ОСAndroid називається Dalvik), а не безпосередньо на реальному процесорі у апаратному потоці команд (через регістр команд).

Також, із самого початку, розробники операційної системи Linux спиралися на загальний підвищений ступінь захисту, що втілилося, наприклад, у надзвичайно малій кількості вірусів, що за усю історію були написані для цієї системи (їх число не перевищує кількох десятків, на відміну від Windows, для якої кількість вірусів сягає десятків і сотень тисяч – залежно від того, чи вважати окремими вірусами модифікації одного і того ж самого движка, що виконувалися різними програмістами-хакерами). Розробники Android не зменшували загального рівня захисту системи, залучивши чимало нових, специфічних механізмів захисту саме мобільних платформ. Так, наприклад, за умовчанням у цій системі відключена можливість установки нових програм (додатків) із неперевірених джерел (наприклад, з карти пам'яті), а надійним вважається лише магазин додатків Google Play Market (та аналогічні онлайн «збірники» додатків типу Amazon Appstore (англ.), Uptodown, Opera Mobile Store, Яндекс.Store, GetUpps!, Mobogenie, F-Droid, 1Mobile Market, Meizu Appstore та ін.).

Уся архітектура ОС побудована таким чином, щоби унеможливити завантаження та запуск на Android небезпечних програм. Ще одним прикладом цього є те, що у посиланнях HTML ігнорується атрибут download. Як відомо, наявність цього атрибуту у тегу якоря <a> замість переходу за посиланням у браузері, викликає завантаження відповідного файлу, на який показує даний URL. Наприклад, тег

< a href="1.jpg" download>

при клацанні користувача по цьому посиланню повинен викликати завантаження файлу 1.jpg, а не його відкриття і перегляд у браузері. Однак така поведінка реалізується лише на настільних ПК, а у мобільних системах на Android даний атрибут просто ігнорується і файл просто відкривається браузером, що набагато безпечніше (оскільки ця дія лімітується можливостями самого браузера), ніж завантаження файлу на власний комп'ютер із подальшим майже стовідсотковим відкриттям (дійсно, більшість користувачів безтурботно відкривають усі файли підряд, що є

наявними у них на пристрої, а особливо, якщо виконуваний файл маскується під безпечний і має іконку якогось широко відомого типу (наприклад, у виконуваної EXE-програми зловмисником встановлена іконка як у стандартних JPG-файлів, або папок).

Постійне стимулювання користувачів встановлювати нове програмне забезпечення виключно із магазинів додатків типу Play Market (а точніше, виключно із цього магазину) обумовлене величезною увагою, що її приділяє компанія Google до сторонніх додатків, що включаються до Play Market. Кожен додаток проходить тестування спеціалістами Google на більше 20 різноманітних вразливостей і, якщо встановлюється можливість використання хоча б однієї з них (а тим більше, декількох), то компанія Google направляє розробникам відповідного листа з рекомендаціями, яким чином найпростіше закрити дану вразливість (вразливості).

Системність в роботі по забезпеченню безпеки Android є характерною рисою роботи компанії Google в цілому. Так, наприклад, пошук більш чи менш серйозних вразливостей у самій операційній системі (а не у сторонніх додатках) призводить до того, що відповідні оновлення випускаються як мінімум щомісяця, а часто – майже кожного тижня. Щоправда, тут ми стикаємося із першим суттєвим недоліком системи безпеки Android: якщо усі розроблені патчі негайно впроваджуються у операційні системи, що встановлюються на нові пристрої і продаються кінцевим користувачам, то системне програмне забезпечення на уже проданих смартфонах часто не оновлюється автоматично. Насправді, переважна більшість користувачів смартфонів не оновлює операційну систему, вважаючи таке оновлення складним процесом, який теоретично може вплинути на стабільність та якість роботи пристрою (що, в принципі, і правда може відповідати дійсності на дешевих пристроях маловідомих виробників). Звичайно, така ситуація становить суттєву небезпеку, не захищаючи існуючих користувачів не лише від вразливостей «нульового дня» (так називають вразливості, що тільки виявлені, і для них компанія-виробник ПЗ ще не встигла випустити патчів,

тобто фактично усі пристрої із даним вразливим ПЗ знаходяться у небезпеці), а й від тих експлоїтів, що були створені місяці і навіть роки тому назад (часто – з моменту покупки смартфону).

Відповідно, можна окреслити перший напрямок роботи щодо підвищення безпеки Android-систем: удосконалення системи оновлення ОС в частині обов'язковості встановлення таких оновлень. Зрозуміло, що існують ситуації, коли встановити оновлення «прямо зараз» користувач не може (дорогий доступ до Інтернету, наявність невідкладних поточних справ, що потребують залучення смартфону, наприклад, для здійснення телефонних дзвінків, і т.п.), однак система має слідкувати за активністю пристрою і завантажувати та встановлювати оновлення безпеки при першій доступній можливості (наприклад, при першому ліпшому підключенні Wi-Fi).

Як бачимо, розробники ОС Android приклали (і прикладають!) чимало зусиль для забезпечення достатнього рівня захищеності, однак, не зважаючи на всі ці заходи, як і для будь-якої іншої системи чи об'єкту, для ОС Android існують певні загрози несанкціонованого використання, а саме, традиційно:

- загрози цілісності даних;
- загрози конфіденційності;
- загрози доступності.

І, як традиційно прийнято при побудові систем захисту інформації, усі такі загрози слід конкретизувати під час виконання етапу аналізу загроз, що і виконується у наступному підрозділі.

1.2. Аналіз загроз, що існують в ОС Android

Отже, як уже зазначено вище, традиційно виділяють загрози трьом таким важливим властивостям інформації, як цілісність, конфіденційність та доступність інформації. Розглянемо їх окремо докладніше.

Цілісність інформації означає відсутність її несанкціонованої модифікації зареєстрованими користувачами або незареєстрованими

суб’єктами. При цьому зареєстрований в системі користувач може також виступати у ролі зловмисника (внутрішнього) при внесенні несанкціонованих змін у технічно доступну йому інформацію. Також зареєстровані користувачі можуть виступати і зовнішніми зловмисниками, якщо перезаписують інформацію, до якої у них доступу не повинно бути. Зовнішнім суб’єктом може виступати і активно працююча комп’ютерна програма-вірус, яка опосередковано діє в інтересах людини-зловмисника (хакера).

Загроза цілісності у ОС Android можуть бути виділені для наступних видів інформації:

- інформації з розділу «Контакти» (віруси-«хулігани» можуть змінювати деякі цифри у телефонних номерах, причому встановити сам факт такої модифікації та відновити вірні дані без залучення зовнішніх джерел інформації є надзвичайно складною задачею);

- відомостей про майбутні події, які прописані у додатках для планування персональної діяльності (органайзерах);

- текстових повідомлень, отриманих будь-яким із доступних каналів комунікації (по-перше, мова йде про звичайні SMS-повідомлення, а також, про повідомлення Інтернет-месенджерів типу Viber, WhatsApp, Telegram, ICQ, QIP, Facebook Messenger, і т.д.);

- будь-яких інших відомостей фактичного характеру, що зберігаються на пристрої під управлінням Android.

Логічний та статистичний аналіз вказаних загроз дозволяє зробити висновок, що вони є мало поширеними, а в деякій мірі видаються, навіть, надуманими.

Конфіденційність, на відміну від цілісності, є основною ціллю для несанкціонованих дій на системі Android. Тут перед зловмисником відкривається надзвичайно широке поле для діяльності, пов’язане, зокрема, із тим, що для мобільної системи (на відміну від настільної) є доступними значно більші масиви інформації про користувача. Це, наприклад:

- сукупність повідомлень SMS, які часто носять особистий характер, несуть безпосередню фінансову інформацію, або таку, що може вплинути на фінансовий стан особи, її репутацію, сімейні стосунки, робочі відносини, і т.п.;

- набір повідомлень із інформацією аналогічного характеру, але технічно іншого походження (не SMS), наприклад, таких, що отримані у різноманітних програмах месенджерах: Viber, WhatsApp, Telegram, Facebook Messenger, і т.д.;

- набір (і майже завжди досить обширний) персональних фотографій, причому актуальних у часі, і причому деякі з яких можуть мати строго конфіденційний характер (фотографії паролів, секретних слів, сторінок облікових записів, зображення інтимного характеру, або такі, що компрометують, і т.п.);

- історія місцеположень, причому з часовими мітками, які відображують цілком і повністю деталізовані переміщення особи протягом тривалих інтервалів часу, що за певних обставин може становити загрозу як особистій та професійній репутації (якщо, наприклад, у робочий час людина регулярно фіксується в інших місцях), сімейному життю, і, навіть, особистій безпеці (при переслідуванні особи історія її місцеположень дозволяє визначити її звички та досить точно спрогнозувати місцезнаходження у майбутні періоди);

- історія перегляду сторінок браузера (цей пункт є аналогічним і для ПК), разом із збереженими паролями, іменами користувача, файлами cookie, завантаженими файлами, і т.д.;

- історія пошуків контенту, причому така, що зберігається із різних сайтів (наприклад, історія пошуку на YouTube, у пошуковику Google, ВКонтакте, і т.д.);

- тощо.

Відмітимо, що перелічені загрози конфіденційності є досить суттєвими, оскільки можуть вплинути на різні сторони життя власника смартфона, включаючи фінанси, репутацію, особисте життя та фізичну безпеку.

Загрози доступності інформації менш актуальні для ОС Android (в першу чергу через те, що її програмні продукти практично не виступають серверами у системах типу «клієнт-сервер») і зводяться в цілому до можливостей блокування певних файлів або папок у файловій системі вірусами типу «шифрувальників», що вимагають гроші за розблокування інформації.

Слід відмітити, що для усіх згаданих вище загроз у системі Android існують вбудовані можливості, або цілі окремі програмні продукти, що забезпечують технічні міри захисту інформації. В цілому в рамках виконання кваліфікаційної роботи бакалавра важко конкурувати із всесвітньо відомим гігантом в частині створення технічних (програмних) систем захисту інформації. Однак, як відомо, крім технічних мір захисту актуальними також є правові та організаційні дії. Очевидно, що правові засоби захисту не можуть забезпечуватися шляхом розробки програмного забезпечення для Android, а ось значна робота по виконанню організаційних мір захисту інформації якраз може бути проведена за рахунок проектування та розробки програмного продукту для цієї операційної системи. Такий продукт, наприклад, може підвищувати швидкість та надійність інформованості працівників про події, пов’язані із забезпеченням захисту інформації на підприємстві чи у компанії. Відповідно, опишемо вимоги до такого продукту докладніше.

1.3. Аналіз науково-технічної літератури, присвяченої питанням захисту інформації на ОС Android

Галузь захисту інформації, або, як її тепер більше називають, кібербезпека, являє собою значний пласт знань, що знаходиться на стику

галузей ІТ, математики (численних її спеціальних розділів), фізики (в частині нейтралізації фізичних каналів витоку інформації) та споріднених наук. Захист інформації розглядається для усіх комп'ютерних пристроїв, а точніше – цифрових. Мобільні телефони, що працюють під управлінням ОС Android, звичайно не є виключенням і для них також у вільному доступі для всіх спеціалістів та просто зацікавлених осіб, існує чимало науково-технічних та чисто прикладних публікацій, присвячених питанням їх захисту. Розглянемо найбільш характерні із них докладніше.

У сучасних публікаціях розглядаються теоретичні питання зведення моделей безпеки систем захисту для ОС Android. Розглядаються можливості адаптації достатньо давно розроблених моделей управління доступом суб'єктів до об'єктів (що були початково створені для комп'ютерних систем), до нових умов, що відповідають мобільним пристроям під управлінням відповідної операційної системи, враховують їх особливості (у порівнянні зі звичайною комп'ютерною технікою). У статті розглянуто також розроблення нових моделей, які базуються на стандартному методі LSI, статистичному та категоріальному аналізі різних сервісів Google, що безперечно є актуальною темою для розвитку систем безпеки даної операційної системи. Найбільш слабким місцем на думку авторів цього дослідження є використання однократної авторизації у пристрої – зокрема, під час початку роботи з ним. Цілком слушно автори заявляють, що подолати цю проблему можливо шляхом розроблення (із подальшим впровадженням у програмних кодах та/або програмно-апаратних рішеннях – за умови наявності технічної можливості підключення додаткових датчиків чи інших девайсів) моделей безперервного захисту. У якості основного підходу до впровадження такого континуального захисту автори пропонують метод латентно-семантичної індексації, хоча чимала увага приділяється і методу аналізу привілеїв. В цілому робота несе більше теоретичний, аніж практичний характер, тобто на її основі мають далі розвинутися готові програмно-технічні рішення, що впроваджують запропоновані авторами методи та моделі.

Окремі роботи, безперечно, мають набагато більш прикладний характер, оскільки містить виключно дані фактичного характеру, без розбору чи зведення складних математичних моделей. Так, в процесі викладення матеріалу докладно проаналізовано наступні елементи аналізу загроз системи:

- вразливості самої операційної системи Android та основного (стандартного, системного) програмного забезпечення, що у ній використовується;

- аналізується така властивість, як відкритість Android (мається на увазі вихідних текстів її коду) і оцінюються ризики, до яких може приводити така відкритість;

- особлива увага приділяється фрагментації платформи, в якій чимала функціональність забезпечується використанням модулів та служб, які працюють більш-менш автономно, та виконується аналіз до яких наслідків може привести ця можливість;

- кінцево, автори розглядають як на основі даної ОС можуть реалізовуватися підходи із галузі соціальної інженерії (а саме, наскільки важко, чи навпаки просто зловмисникові скористатися соціальними методами для виманювання конфіденційної інформації та споріднених матеріалів у користувачів Android.

Окремі роботи є прикладом чисто технічного дослідження, яке може стати в нагоді практикуючим спеціалістам із захисту інформації, які в тому числі опікуються питаннями захисту для мобільних систем (смартфонів). Роботи, що включають набір в основному чисто прикладних відомостей. Відмітимо, що такі публікації можуть використовуватися у вигляді стартової інформації і для наукових досліджень, для введення дослідника у поточний стан речей, пов’язаних із захистом мобільних платформ на Android.

У дослідженнях також представлений аналіз, але не загроз для певної системи (як-то мобільний смартфон), а існуючих методів атаки, причому у досить широкому сенсі, зважаючи на тематику роботи. Незважаючи на це,

тут також присутні цікаві актуальні блоки інформації, присвяченої атакам саме на мобільні платформи, якою по суті і являється пристрій під управлінням ОС Android. Коротко кажучи, атаки на мобільні пристрої враховують такі їх особливості, як:

- постійний і безперервний зв'язок із зовнішніми мережами передачі даних (Інтернет через 3G/4G/5G технології та локальні мережі по Wi-Fi), що, при порівнянні з комп'ютерними системами, характерно більше для серверів;

- відносно слабкі обчислювальні ресурси та мала інформаційна ємність (як у простих настільних ПК чи ноутбуків);

- наявність можливості простого фізичного доступу до пристрою (телефони часто забувають на робочих столах колег, у помешканнях знайомих і т.п.) - як у планшетів, і т.д.

Відмітимо, що у публікаціях все ж більше уваги приділяється загрозам для секретної інформації (зокрема, військового характеру), але принципово, практично усі зроблені висновки та вироблені рекомендації майже без змін (або з мінімальними доробками) можуть бути адаптовані і для конфіденційної інформації персонального характеру, яка зазвичай і обробляється на смартфонах.

Цікавою особливістю ще однієї роботи [5] є спроба прогнозування шляхів розвитку проблем захисту інформації, що були наявними на час виходу публікації (ймовірно 2016-2017 р.), для даної ОС. Володіння такою інформацією дозволяє професіоналам галузі кібербезпеки працювати на випередження, зводячи вже сьогодні системи захисту від загроз, що реально можуть з'явитися тільки потім (наприклад, після виходу у вільний продаж певного апаратного рішення, або програмного продукту).

Приємно зауважити, що колеги-здобувачі вищої освіти також вносять суттєвий внесок у розвиток області інформаційної безпеки на мобільних платформах, зокрема таких, що працюють під управлінням ОС Android. Так магістерська дисертація [4] присвячена питанням захисту інформації для ОС Android, зокрема, дослідженню не зовсім самої системи, а в основному – її

додатків. При цьому автор-дослідник не бере інформацію із відкритих джерел, а отримує її власноруч, виконуючи дослідження Android-програм методами зворотного проектування (reverse engineering). При такому підході першим кроком є виконання дизасемблювання готової програми (що раніше уже була від трансльована із вихідного тексту у готовий код), після чого уся програма представляється набором величезної кількості дуже простих асемблерних команд (елементарні команди, які апаратно можуть виконуватися процесором цільового цифрового пристрою – комп'ютера чи смартфона і т.п.). Після дизасемблювання можна спробувати знайти цікаві місця у цьому величезному лістингу за допомогою програм відладчиків на зразок OllyDbg або аналогічних. За необхідності спеціаліст може змінювати двійковий код програмного забезпечення, вносячи невеликі корективи у його хід виконання (таким прийомом часто користуються зломщики пропрієтарних, тобто платних програмних продуктів, реалізуючи так звані патчі; пропатчені програми працюють за зміненими алгоритмами, і це дозволяє зловмиснику обходити алгоритми захисту програм, незаконно використовуючи їх без наявних прав та ліцензій).

Відмітимо, що корисну вхідну інформацію для проведення досліджень можна отримати не тільки у науково-технічних (чи популярних) вітчизняних публікаціях, а й розглядаючи наявні технічні рішення, присвячені вирішенню проблеми дослідження у зарубіжних публікаціях.

В першу чергу, самим очевидним класом програмних продуктів, що відноситься до галузі інформаційної безпеки є антивіруси. Із самого початку використання комп'ютерної техніки використовувалися програми-комп'ютерні антивіруси, але із широким поширенням мобільних пристроїв виникла нагальна потреба у використанні програм-мобільних антивірусів.

Зрозуміло, що лідируючі позиції на ринку мобільних антивірусів зайняли компанії, що давно і успішно займалися розробкою антивірусів комп'ютерних. Адаптувавши їх до вимог мобільних девайсів, вони отримали цілком робочі і ефективні програмні продукти, що зайвий раз підтверджує

спорідненість процесів захисту інформації на комп'ютерних та мобільних платформах.

Мобільні антивіруси, так само, як і комп'ютерні, мають резидентні версії, що здійснюють моніторинг усіх операцій, що здійснюються в системі, забезпечуючи безперервний захист – рис. 1.2.

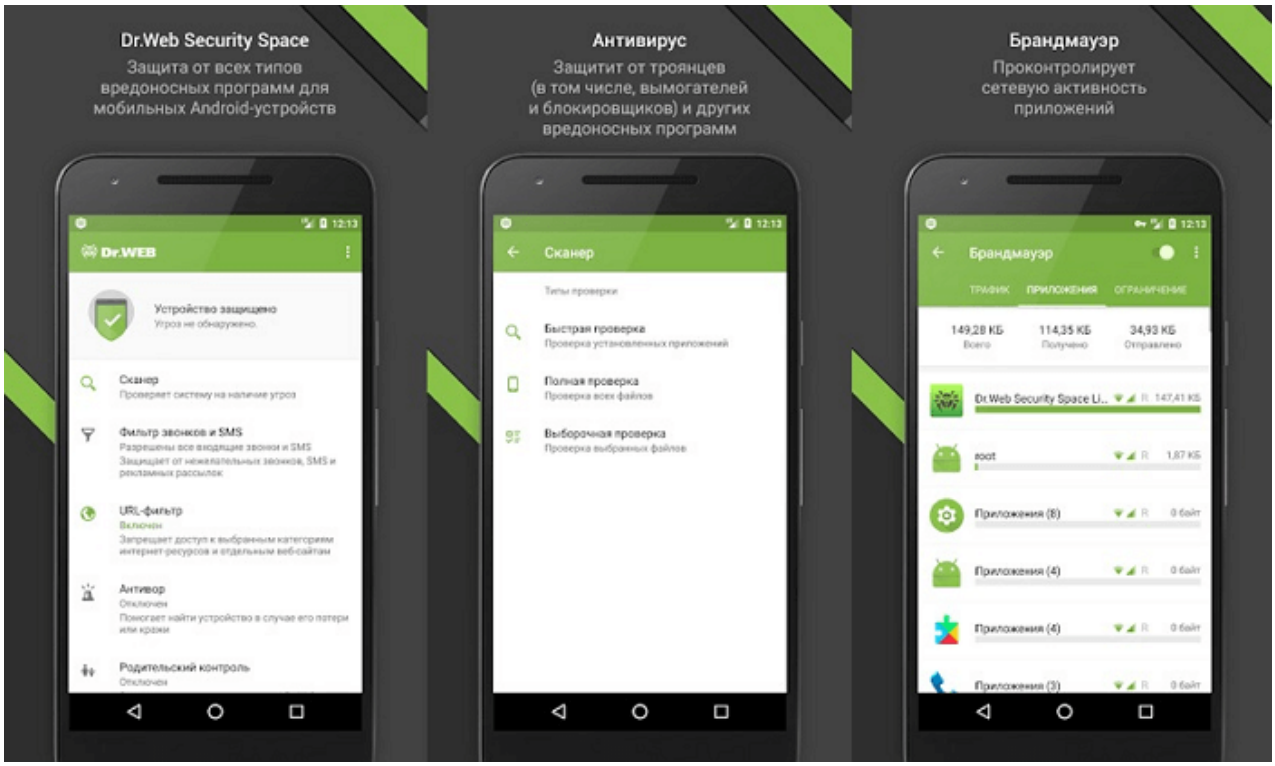

Рис. 1.2. Приклад поширеного мобільного антивірусу Dr. Web (показано екрани: загальний, сканер, брандмауер)

Відмітимо, що на рис. 1.2 у правому екрані показано брандмауер, який також є обов'язковим компонентом захисту мобільних пристроїв, і інтеграція цього продукту у антивірус (що часто зустрічається на мобільних платформах) є досить зручним рішенням, що обумовлене особливостями апаратної платформи, на якій запускається програмне забезпечення (в першу чергу, малий обсяг пам'яті для запуску кількох програм одночасно).

Серед найбільш популярних антивірусів дял мобільних платформ можна назвати:

– Kaspersky Mobile;
– Dr Web Security Space;
– 360 Security;
– Norton Security;
– McAfee;
– Malwarebytes Anti-Malware;
– Eset Mobile Security;
– Bitdefender Antivirus;
– AVG AntiVirus;
– CM Security;
– Avira Antivirus;
– AVAST Mobile;
– AndroHelm Mobile Security;
– TrustGo Mobile Security;
– Lookout;
– AVL.

Ще одним прикладом програми, призначеної для захисту мобільних пристроїв на Android, є додаток AppLock – рис. 1.3. Цим додатком досить просто користуватися: AppLock захищає окремі додатки від зломщиків, запитуючи ПІН-код або графічний ключ. Таким способом можна захистити SMS, контакти, Gmail, та й взагалі будь-який додаток.

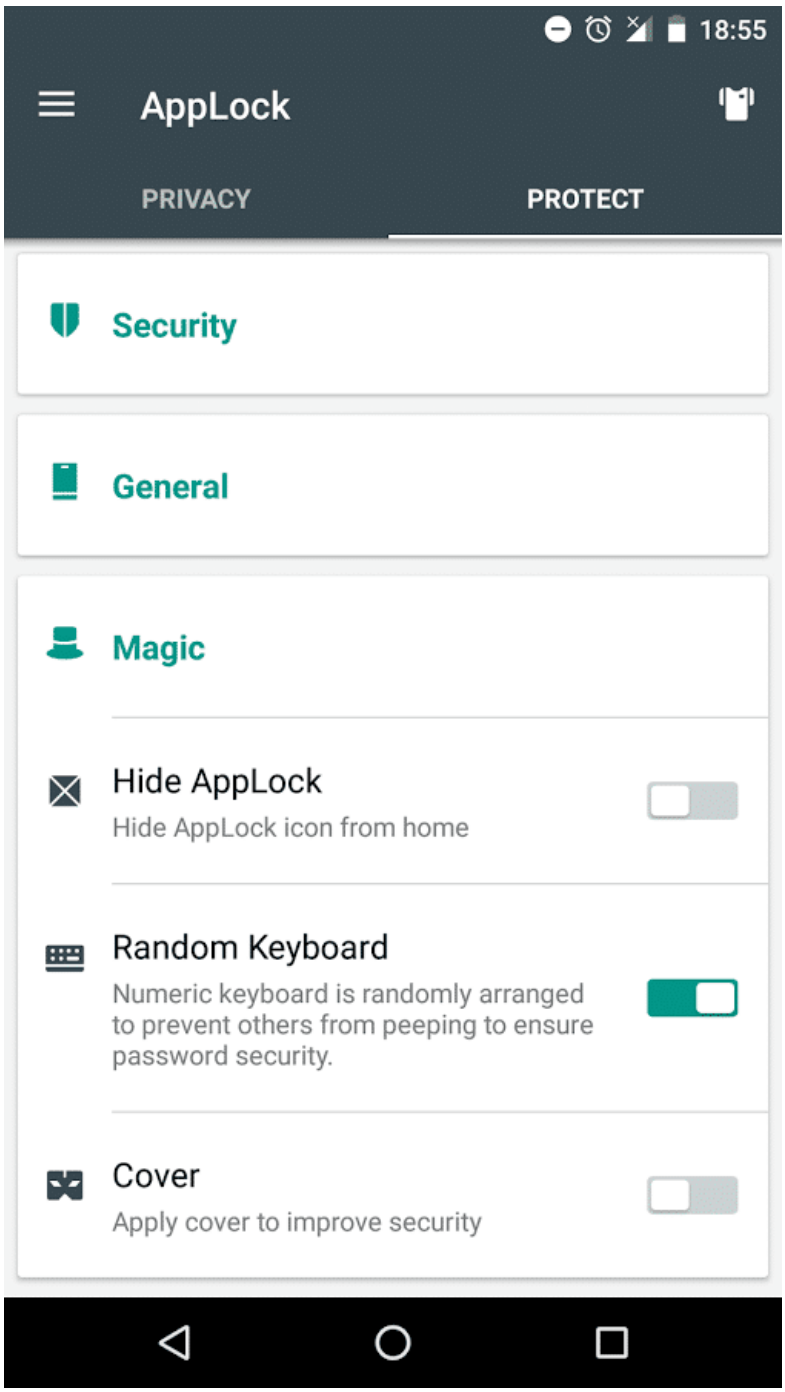

Рис. 1.3. Зовнішній вигляд вікна додатку AppLock, який дозволяє здійснювати захист на рівні окремих додатків ОС Android

Не слід плутати таке блокування додатків і вбудовану в телефон можливість блокування всього пристрою. Вбудоване блокування розповсюджується на весь телефон, при якому немає доступу у жодні програми. В свою чергу AppLock дозволяє заблокувати тільки обрані програми, не заважаючи роботі інших, що не потребують захисту.

Також при розгляді програмного забезпечення для захисту інформації на мобільних пристроях неможливо не згадати програму, призначену для безпечного обміну повідомленнями, адже значна частка усього часу роботи із мобільними пристроями присвячена саме обміну повідомленнями. Отже, розглянемо програму Signal Private Messenger докладніше – рис. 1.4.

Рис. 1.4 Вікно програми Signal Private Messenger

Існує безліч програм для безпечного обміну повідомленнями. Більшість із них працюють тільки, якщо обидва користувачі використовують один і той же додаток. Однак, Signal Private Messenger дозволяє додати додатковий рівень захисту до звичайних SMS, навіть якщо один з користувачів не користується Signal Private Messenger. Додаток розроблений Open Whisper System. Цей мобільний додаток має такі особливості:

- відкритий вихідний код;
- наскрізне шифрування. На сервері програми не зберігається нічого;
- шифрування можливо навіть якщо у одного з користувачів немає Signal Private Messenger.

Аналізуючи властивості даного додатку можна зробити висновок, що його використання для цілей захисту конфіденційної інформації на ОС Android є обов’язковим.

1.4. Уточнення постановки задачі дослідження

Повторімось, що організаційні міри по забезпеченню захисту інформації являють собою важливу складову усього комплексу заходів галузі кібербезпеки. Для забезпечення організаційного захисту інформації важливою умовою є надійне та швидке інформування усіх учасників виробничих процесів про ситуації, пов’язані власне із самим захистом. Якраз для цього доцільно використовувати мобільне програмне забезпечення власної розробки, що збудовуватиметься в рамках даного дослідження.

Програмне забезпечення має бути реалізоване по сучасній об’єктно-орієнтованій технології програмування, у вигляді мобільного додатку, написаного мовою Java.. Середовище розробки – Eclipse, яке до сих пір ще використовується в реальних задачах, поступаючись потроху, але стабільно середовищу Android Studio від Google, яке, нажаль, має набагато більші системні вимоги. Така ситуація обумовлена тим, що дане середовище є розвитком, адаптацією більш універсального продукту IntelliJ IDEA призначеного для програмування на Java в цілому, до особливостей розробки саме мобільних додатків. Очевидно, що Android Studio має значно кращі властивості по зведенню проекту додатку, але через завищені системні вимоги не підлягає вибору.

Проектований мобільний додаток повинен мати дві наступні підсистеми:

- оперативного отримання інформації від центрального джерела;

- ефективного відображення інформації користувачеві.

Програма повинна надавати можливості виконувати наступні налаштування:

- вибір режиму автоматичного оновлення або ручного (для особливо важливих подій на зразок засідання у директора, коли автоматичне оновлення є небажаним);

- регулювання інтервалу оновлення (збільшення цього значення економить трафік, а зменшення підвищує ефективність системи, тому не бажано виставляти це значення більше кількох хвилин);

- ранжування повідомлень по ступеню важливості;

- встановлення порогового значення ступеня важливості, для того, щоби відображувати лише повідомлення із ступенем важливості не менше встановленого.

Мобільний додаток має бути інтегрований із веб-службою інформаційної підтримки. Під веб-службою матимемо на увазі робочий веб-сервер, на якому у тому, чи іншому форматі (конкретний спосіб представлення інформації слід розробити в рамках даної роботи) викладаються інформаційні повідомлення, які слід відображувати усім клієнтським частинам, встановленим на смартфонах (які власне і являють собою мобільний додаток).

Інформація на веб-сервері має викладатися у стандартизованому форматі XML, або JSON. Для реальної розробки обираємо перший варіант через наявність великої кількості бібліотек, прикладів коду, комплексних наробок щодо опрацювання XML-документів. В той же час JSON хоча і набуває все більшої популярності, але поки що є порівняно новим продуктом, найбільш широке впровадження якого можна очікувати протягом кількох найближчих років.

Керуючись цією інформацією, можна переходити до проектування безпосередньо централізованої служби доставки повідомлень, що виконано у наступних розділах.

Висновки по розділу 1

Таким чином, у даному розділі проаналізовано особливості забезпечення захисту інформації в ОС Android. Встановлено, що компанією Google, яка на даний момент є виробником цієї операційної системи, ведеться активна і планомірна робота по забезпеченню захисту інформації у ній. Технічні заходи із забезпечення захисту інформації пророблені досить повно і недоцільно повторювати їхню функціональність в рамках даної роботи. Однак, з метою проведення організаційних мір захисту, доцільною є розробка програмного продукту для цієї ОС, за допомогою якого можна було би вести ефективний централізований обмін повідомленнями між працівниками компанії та особою, відповідальною за забезпечення захисту інформації (призначений працівник служби безпеки – у великих компаніях, або менеджер загального профілю – у менших). Створення такої централізованої служби (Service Centre), підвищує інформованість працівників про поточні питання, пов’язані із захистом інформації, а, отже, збільшує надійність усіх ланок захисту, де такі працівники задіяні. Відповідно, у наступних розділах буде здійснено проектування та реалізацію такої служби.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ АЛГОРИТМІВ ЗАХИСТУ ІНФОРМАЦІЇ В СИСТЕМІ ANDROID

2.1. Пошук слабких місць в існуючих алгоритмах захисту ОС Android

Як було зазначено вище, компанія Google, що є розробником операційної системи Android чималу увагу приділяє питанням захисту інформації. Із самого початку дана ОС була побудована на ядрі Linux, що мало на порядок кращі механізми захисту, ніж ОС сімейства Windows, а потім, в процесі розробки нових версій, Google не зменшувала рівень захисту, а навпаки – підвищувала, зокрема, шляхом застосування технічних мір захисту, багатократно досліджуючи висхідні коди системи та оперативно випускаючи патчі для нових вразливостей.

Якщо вдатися до аналізу статистики, то відкритою на сьогодні є наступна інформація. На даний момент Android є найбільш популярною мобільною ОС з часткою ринку більше 75%. Кількість додатків, завантажених з магазину Google Play Market, в 2016 році склало 65 мільярдів. Паралельно спостерігається і бурхливе зростання числа шкідливих додатків: у 2015 році їх було виявлено 2,3 мільйона (!!). Крім того, близько 60% Android-пристроїв працюють на версіях ОС з відомими уразливими, і вони потенційно можуть бути атаковані зловмисниками. Ці загрози, в свою чергу, призвели до розвитку засобів захисту. Офіційний магазин Google Play ввів фільтрацію додатків за допомогою пісочниці Google Bouncer. Антивірусні компанії стали випускати свої продукти під Android (хоча, багато хто з них самі по собі містять небезпечні уразливості). Число наукових публікацій з даної теми також зросла: одна з оглядових робіт 2015 року про рішеннях безпеки для Android налічує понад 40 посилань (проектів створення систем захисту) і згадує далеко не всі відомі на даний момент дослідження. У зв'язку з тим, що мобільні пристрої є джерелом і сховищем великої кількості

конфіденційних даних, проблема безпеки ОС Android і її додатків стоїть особливо гостро.

Платформа Android є вільним програмним забезпеченням, база її вихідних кодів повністю відкрита. Виробники пристроїв самостійно розвивають кодову базу, створюючи спеціалізовані прошивки з метою досягнення більшої функціональності і кращої продуктивності. Побічним результатом такої діяльності стають уразливості і слабкості в реалізації алгоритмів, відсутні в основній кодової базі, але існуючі на безлічі реальних пристроїв. Шкідливе ПЗ використовує ці уразливості для підвищення прав і подолання захисних механізмів. Виявлення вразливостей в прошивках при відсутності вихідних кодів вкрай утруднено. Першочерговою проблемою стає відсутність середовища контрольованого виконання, яка необхідна для проведення динамічного аналізу.

Як уже було зазначено вище, додатки для операційної системи Android розробляються на мові Java. Починаючи з версії Android 1.5 був представлений набір інструментів Android NDK, який дозволяє розробляти модулі додатків на мовах C і C++ і компілювати їх в машинний код. Додатки поставляються у вигляді файлів спеціального формату APK, який є ZIP-архівом з певною структурою каталогів і файлів. APK-файл програми містить:

- маніфест;
- модулі, скомпільовані в машинний код (колективні бібліотеки .so);
- різні ресурси програми;
- DEX-файл;
- інші необхідні файли.

Починаючи з версії Android 4.4 підтримуються два середовища виконання додатків: Dalvik VM і ART. Слід зазначити, що процес підготовки APK-файлу не змінився, змінився тільки процес установки додатків в новому середовищі виконання ART. Починаючи з версії 5.0 віртуальна машина ART стала використовуватися за умовчанням замість Dalvik VM.

Таким чином, повноцінний аналіз безпеки пристрою потребує вивчення властивостей системного і прикладного програмного забезпечення в сукупності. У даному підрозділі представлена класифікація проблем безпеки Android, а також список вимог до системи.

Середовище виконання Java-програм Dalvik VM на Android сильно відрізняється від «звичайної» Java VM. По-перше, при компіляції Java-програми вона компілюється в байт-код віртуальної машини Dalvik, який сильно відрізняється від байт-коду віртуальної машини HotSpot. Віртуальна машина Dalvik є регістровою, що робить її виконання на RISC-архітектурах, часто використовуваних в мобільних пристроях (як, наприклад, рішення на базі ARM), більш ефективним. Також при розробці Dalvik враховувалися обмеження пам'яті в мобільних пристроях. Починаючи з версії Android 2.2 середовище Dalvik містить JIT-компілятор, який компілює найбільш критичні до швидкості виконання ділянки коду Java-програм в машинний код.

Стандартні бібліотеки Java були або замінені на допрацьовані бібліотеки з пакету Apache Harmony або написані заново. При компіляції Java-програми виходить файл формату DEX (Dalvik Executable), який містить байт-код для віртуальної машини Dalvik. Формат файлу був розроблений з метою скорочення обсягу займаної пам'яті і суттєво відрізняється від стандартного формату JAR. Для виклику модулів, реалізованих в машинному коді, з Java-програм використовується інтерфейс JNI. Варто відзначити, що є можливість довантажувати машинні модулі динамічно по мережі за допомогою компонента DexClassLoader.

Батьківським процесом для всіх додатків в ОС Android є процес Zygote. Даний процес являє собою каркас Android-додатку, в якому вже завантажені всі необхідні бібліотеки оточення Android, але відсутній код самого додатку. Запуск програми Android з точки зору операційної системи відбувається наступним чином.

Спочатку відбувається системний виклик fork для створення нащадка від процесу Zygote. У цьому новому процесі відкривається файл програми яка запускається (системний виклик open). Далі відбувається читання інформації про файли класів (classes.dex) і ресурсів з файлу програми. Відбувається відкриття сокетів для IPC. Виконується системний виклик mmap для відображення файлів програми в пам'ять. Далі середовище виконання робить налаштування необхідного оточення і виконує додаток (інтерпретує байт-код Dalvik або передає управління функціям у виконуваному коді - у разі використання ART).

На рівні ядра ОС Android кожен додаток є окремим процесом з унікальними значеннями user / group ID, які даються йому при установці. Таким чином, кожна програма працює в своїй пісочниці. Починаючи з версії 4.3 на додаток до цієї політики безпеки додалося використання компонента мандатної контролю доступу SELinux.

За умовчанням додаток обмежений у своїх можливостях і не може зробити нічого, щоб негативно вплинути на іншу програму або користувача: ні прочитати призначені для користувача дані, ні модифікувати системні програми, не має доступу до мережі. Для отримання доступу до різних ресурсів додаток звертається до сервісів ОС Android. Дозволи на доступ задаються статично в файлі маніфесту додатку і видаються йому під час роботи по мірі необхідності. Система запитує у користувача згоду на видачу цих дозволів під час установки або під час оновлення програми. Відповідальність за видачу доступу додатку лежить на користувачеві, він самостійно вирішує, якому додатку давати дозвіл на певні дії, а яким не давати, - це виконується один раз, під час його установки. Використання такого підходу, при якому всі дозволи видаються разом при установці додатка, призвело до того, що користувачі стали роздавати повноваження додатків, не замислюючись про наслідки. У свою чергу, додатки стали запитувати все більше дозволів в міру розширення їх функціональності. В Android 6 Marshmallow дана система замінена на іншу: додаток запитує

доступ до ресурсів у користувача під час його роботи, а користувач може або дозволити доступ, або заборонити його.

Android-додаток складається з чотирьох видів компонентів і не містить функції main () або якоїсь іншої єдиної точки входу. Компоненти додатків взаємодіють один з одним за допомогою спеціальних повідомлень, званих Intent.

Компоненти під назвою Activity визначають призначений для користувача інтерфейс. Як правило, використовується один компонент Activity для опису одного екрану програми. Activity може запустити іншу Activity, передавши параметри за допомогою механізму Intent. Під час роботи тільки одна Activity може працювати і обробляти користувальницький інтерфейс, інші на цей час залишаються замороженими або знищуються, в залежності від обраного режиму роботи програми.

Компоненти під назвою Service виконують фонову обробку даних. Якщо якійсь Activity потрібно виконати певну операцію, наприклад завантаження файлу або програвання музики, і продовжити роботу, коли користувач перейшов в інший додаток або згорнув поточний, додаток запускає сервіс, мета якого - виконання цієї операції. Розробники можуть використовувати Service як додаток-демона, який стартує під час завантаження системи. Компонент Service, як правило, підтримує RPC (Remote Procedure Call), тому інші компоненти системи мають можливість вести з ним міжпроцесний обмін даними.

Компонент Content provider зберігає і обмінюється даними, використовуючи інтерфейс реляційних баз даних. Кожен Content provider містить унікальний URI для даних і обробляє запити до нього у вигляді черг SQL (використовуються традиційні запити типу Select, Insert, Delete).

Компоненти Broadcast receiver виступають в ролі ящиків для повідомлень від інших додатків.

Беручи до уваги ці, та вищенаведені особливості архітектури ОС Android, можна виділити наступні 8 класів проблем безпеки, що притаманні реальним системам на базі цієї ОС:

1) Уразливості ядра Linux і його модулів.

До даної категорії проблем відносяться уразливості, які притаманні всім ОС, заснованим на тій же версії ядра Linux, що і ОС Android. Експлоїти, що використовують уразливості в ядрі, можуть отримати дані користувача або права адміністратора системи. Підвищивши привілеї процесу до прав адміністратора системи, шкідлива програма може відключити систему безпеки Android, вставити в існуючі програми шкідливий код і встановити руткіт. До того ж виробники різних пристроїв додають в ядро модулі для своїх пристроїв; в цих модулях також можуть бути уразливості. Також варто відзначити, що зовсім недавно була виявлена уразливість в компоненті ashmem для взаємодії між процесами в Android.

2) Уразливості модифікацій і компонентів виробників пристроїв.

Останнім часом виробники різних мобільних пристроїв стали випускати свої модифіковані прошивки Android. Ці прошивки можуть містити різні додатки і сервіси, розроблені виробником пристрою, які найчастіше не можна видалити. Наприклад, в жовтні 2016 року було виявлено прихований бекдор в прошивках Foxconn. Аналіз цих сервісів показав, що в них міститься від 65 до 85% вразливостей, виявлених у всій системі. До того ж варто відзначити, що уразливості, які були виявлені і виправлені в основний гілці Android, можуть довгий час залишатися в гілках виробників пристроїв.

3) Уразливості модулів в машинних кодах.

Android-додатки підтримують запуск машинного коду через інтерфейс JNI. Це породжує ще одну категорію вразливостей, пов'язану з широко відомими уразливими витоками пам'яті в низькорівневих мовах (наприклад, в C і C ++). Оскільки на рівні процесів ОС Android немає ніяких обмежень, крім тих, що накладаються ядром Linux, сторонні бібліотеки в машинних

кодах можуть використовувати дозволи, видані всьому додатку, для здійснення шкідливої активності (див. також наступну категорію вразливостей, п. 4). Також модулі додатка в машинних кодах використовуються авторами шкідливих програм, щоб обійти інструменти аналізу і моніторингу рівня Android. Ці модулі також можуть використовувати техніки протидії аналізу, що використовуються в звичайних, настільних програмах.

4) Уразливості механізмів міжкомпонентної взаємодії.

До даної категорії відносяться уразливості, пов'язані із взаємодією між різними компонентами додатків. Так як на рівні операційної системи додаток обмежений пісочницею процесу, йому необхідний механізм доступу до компонентів ОС Android для взаємодії з ними. Це відбувається через пристрій / dev / Binder і різні інші сервіси ОС Android. Як вже говорилося вище, параметри цього доступу задаються у файлі маніфесту, у вигляді XML-файла з дозволами. Практика показує, що такий підхід має недоліки, завдяки яким існують реальні шляхи обходу відповідних механізмів. Так, наприклад, додаток може скористатися правами доступу іншої програми і отримати за допомогою нього дані через ICC. Також можуть існувати уразливості, пов'язані зі сторонніми бібліотеками. Сторонні бібліотеки, які використовуються в додатку, отримують той же набір обмежень, що і сам додаток. Тому сторонні бібліотеки можуть використовувати дозволи, видані всьому додатку, для здійснення своєї шкідливої активності. Додатки до того ж можуть перехоплювати системні події, що пересилаються через циркулярний запит, і зберігати інформацію про вхідні дзвінки й SMS.

5) Уразливості в самих додатках.

Кожна програма зберігає якісь дані про користувача. Ці дані повинні бути захищені належним чином, щоб до них не могли отримати доступ інші програми, - але такий захист передбачений не завжди. Наприклад, Skype в одній із версій додатка зберігав базу даних контактів у відкритому вигляді на диску. Таким чином, контакти можна було прочитати будь-яким іншим

додатком, у якого є доступ до диску. Також додатки можуть використовувати криптографічні бібліотеки з помилками, або ж якісь власні реалізації. До того ж не всі програми проводять хорошу автентифікацію і авторизацію користувача. Крім цього, додатки можуть дозволяти SQL-ін'єкції і схильні до атак XSS. Також варто відзначити, що більшість розроблюваних додатків написані на Java без використання будь-якого захисту для бінарного коду, а байт-код Java, як відомо, легко піддається дизасемблюванню та аналізу. Варто зазначити, що до цієї категорії вразливостей відноситься також відомий список Mobile OWASP-10.

6) Уразливості у вбудованих сервісах і бібліотеках.

Стандартний набір бібліотек і сервісів, що працюють в Android, також містить уразливості. Наприклад, недавно була виявлена уразливість Stagefright в бібліотеці для відображення відео в MMS-повідомленнях, якій були піддані всі версії Android, починаючи з 2.2. Пізніше була виявлена уразливість в компоненті MediaServer, якій піддаються всі версії Android з 2.3 до 5.1. Існують вразливості навіть у середовищі Dalvik: запустивши велику кількість процесів в системі, можна домогтися, що подальший процес запуститься з правами адміністратора.

7) Інтернет-джерела.

Android-додатки поширюються через широке кількість джерел крім офіційного магазину додатків. Оскільки Android-додатки написані в основному на Java, то вони легко піддаються зворотному інжинірингу та перепакуванню з використанням шкідливого коду. Крім того, пісочницю аналізу додатків Bouncer, яка використовується в офіційному каталозі, при певних обставинах можна легко обійти. Тому і в самому офіційному магазині міститься велика кількість шкідливих програм. Крім цього, Android підтримує віддалену установку додатків через Google Play на пристрої, пов'язаному з Google-аккаунтом. Таким чином, якщо зламати обліковий запис Google для пристрою, можна встановити з Google Play шкідливий додаток, який туди попередньо завантажив зловмисник. При цьому на екрані

мобільного телефону не потрібно буде робити жодних підтверджень цих дій, оскільки вони запитуються у вікні браузера і додаток встановлюється на телефон в фоновому режимі при отриманні доступу до Інтернету. Також до цієї категорії вразливостей відноситься використання соціальної інженерії, коли для продовження роботи пропонують встановити додаток з неавторизованого джерела.

8) Уразливості апаратури і пов'язаних з нею модулів і протоколів.

Мобільні пристрої, що працюють під управлінням ОС Android, мають широкий набір апаратури для взаємодії із зовнішнім світом. Відповідні уразливості можна експлуатувати при безпосередній близькості до пристрою або при наявності фізичного доступу до пристрою.

З того моменту, як були випущені перші Android-телефони, було написано велику кількість інструментів для аналізу Android-додатків. Ці інструменти можуть в першу чергу бути класифіковані за видами аналізу, який вони провадять: статичний, динамічний і їх об'єднання (змішаний). Розглянемо їх докладніше.

1) Статичний аналіз.

Інструменти статичного аналізу можна розділити на наступні категорії:

- інструменти, що дістають метаінформацію з маніфесту додатку та надають інформацію про запитувані дозволи, компоненти Activity, сервіси і зареєстровані Broadcast receivers. Метаінформація часто використовується пізніше в динамічному аналізі для запуску функцій програми;

- інструменти для модифікації існуючого байт-коду програми з використанням техніки інструментування. Це дозволяє, наприклад, додавати трасування функціональності в існуючі програми;

- інструменти, які реалізують декомпілятор або дизасемблер байт-коду Dalvik.

Одним з найбільш популярних інструментів зворотньої розробки є Apktool. Він має можливості декодування ресурсів програми приблизно в оригінальну форму, перепакування додатків назад в APK / JAR-файли,

налагодження байт-коду smali. Для декомпілювання і компілювання в apktool байт-коду Dalvik використовується інший широко відомий проект smali / backsmali. Також для дизасемблювання байт-коду Dalvik часто застосовується інструмент Dedexer.

Radare2 – це ще один цікавий інструмент з відкритим вихідним кодом для дизасемблювання, аналізу, налагодження та зміни бінарних файлів Android-додатків.

Один з найбільш різнобічних інструментів для статичного аналізу - Androguard. Він може аналізувати код і декомпілювати додаток назад у вихідний код Java. Отримавши два APK-файлу, він може порахувати коефіцієнт їх схожості для детектування перепакованих додатків або відомих шкідливих програм. Завдяки своїй гнучкості він використовується в деяких інструментах змішаного аналізу.

Прикладами таких атак є атака типу «відмова в обслуговуванні» на технологію Wi-Fi Direct, крадіжка даних кредитних карт за допомогою NFC, виконання віддаленого коду через Bluetooth, установка шкідливої програми без відома користувача через adb за допомогою механізму бекапів. Існують уразливості, за допомогою яких можна підвищити привілеї додатку, використовуючи помилки в реалізації протоколу adb.

Більш повний список інструментів статичного аналізу можна знайти зокрема у Всесвітній мережі на відповідних хакерських ресурсах та сайтах, присвячених кібербезпеці. Слід зазначити, що статичний аналіз має ряд істотних обмежень, пов'язаних з тим, що у зловмисника є лише абстрактне уявлення про програму. Наприклад, статичний аналіз стає набагато складніше, якщо до програми застосовані перетворення обфускації коду. Залежно від складності обфускациі деякі (а може, і все) статичні підходи можуть стати абсолютно марними. Якщо виклик кожного методу робиться тільки побічно, навряд чи вдасться побудувати граф викликів програми. В Android таке перетворення можна зробити, використовуючи Java Reflection API для виклику методів і створення об'єктів замість використання

звичайних викликів і інструкцій створення нового об'єкта. На ринку рішень щодо захисту вихідного коду вже представлено кілька продуктів для обфускації файлів Android-додатків, які можуть звести нанівець всі переваги статичного аналізу.

2) Динамічні і змішані інструменти аналізу.

Інструменти динамічного аналізу відстежують поведінку невідомого додатку під час виконання, при запуску його в контрольованій пісочниці для отримання поведінкового сліду. Для цього проводиться інструментування пісочниці на різних рівнях архітектури ділянками коду, які відстежують поведінку програми та ОС Android.

Архітектура пісочниці Android представляє собою емулятор (найчастіше QEMU), всередині якого працює ОС Android. Інструментування проводиться або на рівні емулятора, або на рівні ОС Android, або і там і там. Рівень ОС Android також ділиться на чотири підрівні:

- додатки;
- середовище розробки додатка;
- робоче оточення програми та бібліотеки;
- ядро і його модулі.

Техніки, що використовуються в динамічному аналізі програми є наступними:

- відстеження помічених даних. Такі інструменти використовуються, щоб відстежувати потенційні витоки конфіденційної інформації;
- моніторинг системних викликів. Такі інструменти можуть записувати системні виклики за допомогою інструментування віртуальних машин, strace або модуля ядра. Це дозволяє виробляти трасування машинного коду;
- трасування методів (функцій). Такі інструменти можуть відстежувати виклики Java-методів додатку у віртуальній машині Dalvik, виклики функцій в машинних кодах через JNI, системні виклики і переривання.

До інструментів змішаного аналізу відносяться інструменти, які поєднують в собі техніки динамічного і статичного аналізу.

В цілому ж можна констатувати, що існуючі інструменти динамічного аналізу мають ряд серйозних недоліків. Дані недоліки притаманні і стандартному інструменту аналізу додатків в магазині Google Play - Google Bouncer, внаслідок чого шкідливі програми можуть поширюватися через офіційний магазин, не будучи виявленими.

Зіставлення можливостей підходів і систем, описаних в літературі, дозволяє сформулювати вимоги до ідеальної системи аналізу платформи Android, яка здатна аналізувати в сукупності додатки і всі верстви системного ПО. Система запозичує деякі ефективні прийоми з розглянутих робіт, комбінуючи їх з метою подолання недоліків в існуючих рішеннях.

Зважаючи на вищесказане, реально ефективне покращення технічного захисту операційної системи Android в рамках бакалаврської роботи є малоймовірним. З іншого боку, усі мобільні пристрої знаходяться в експлуатації кінцевих користувачів (реальних людей), які пов’язані між собою комунікативними зв’язками і тут доцільною є розробка відповідного програмного забезпечення, що забезпечувало би середовище для передачі повідомлень. Причому, якщо звичайні повідомлення, пов’язані із рутинною роботою компанії або підприємства, можуть передаватися відомими програмами месенджерами (типу Viber та аналогічних), то для передачі повідомлень, пов’язаних із захистом інформації, слід використовувати окреме програмне забезпечення власної розробки, яке, відповідно, точно не має прихованих функцій по слідкуванню за користувачем та читанню його повідомлень.

Дійсно, із стабільною періодичністю у світі IT проходить інформація про викриття прихованої функціональності у тому, чи іншому програмному забезпеченні, особливо коли воно створюється невеликими або середніми компаніями. З іншого боку, не виключена ситуація і зловмисних дій окремих працівників великих компаній, що не відповідають їх загальній доброчинній політиці, але, нажаль, є реально існуючими фактами. Таким чином, незалежно від рівня компанії-виробника, слід констатувати потенційну

можливість вбудовування прихованої функціональності у програмні продукти будь-якого призначення, в т.ч. і у мобільні додатки.

Часто така прихована функціональність стосується саме стеження за користувачем, доступу до його конфіденційної інформації або інших дій, що були розглянуті у підрозділі 1.2 при аналізі загроз, що існують в ОС Android. Відповідно, якщо існує така можливість, для вирішення питань захисту інформації краще використовувати програмне забезпечення власної розробки, уся функціональність якого є видимою при аналізі висхідних текстів.

Таким чином, для пересилання повідомлень, що стосуються галузі кібербезпеки, доцільно використовувати мобільне програмне забезпечення власної розробки, до розробки алгоритмів якого слід перейти.

2.2. Удосконалення алгоритмів захисту ОС Android

Таким чином, слід реалізувати службу централізованого інформування про події, що стосуються галузі захисту інформації.

Консультації з працівниками, що пов’язані з питаннями захисту інформації, показали, що для них не є важливою наявність гарних продвинутих елементів управління, а для них головною є функціональність продукту, яка полягає у здійсненні пришвидшеного (у порівнянні зі звичайним інформуванням телефоном чи безпосередньою розмовою) інформаційного обміну між ними. Відповідно, інтерфейс розроблюваного програмного забезпечення може бути наближений до мінімалістичного.

Головним елементом управління буде елемент ListView – рис. 2.1, який буде заповнюватися повідомленнями, що лежатимуть у спеціально виділеному місці на сервері, у XML-файлі (не зважаючи на зростаючу популярність JSON, формат XML ще активно використовується для збереження структурованих даних, якими є якраз повідомлення, що слід

поширювати між працівниками). Уся організація системи, що проектується, наведена на рис. 2.2.

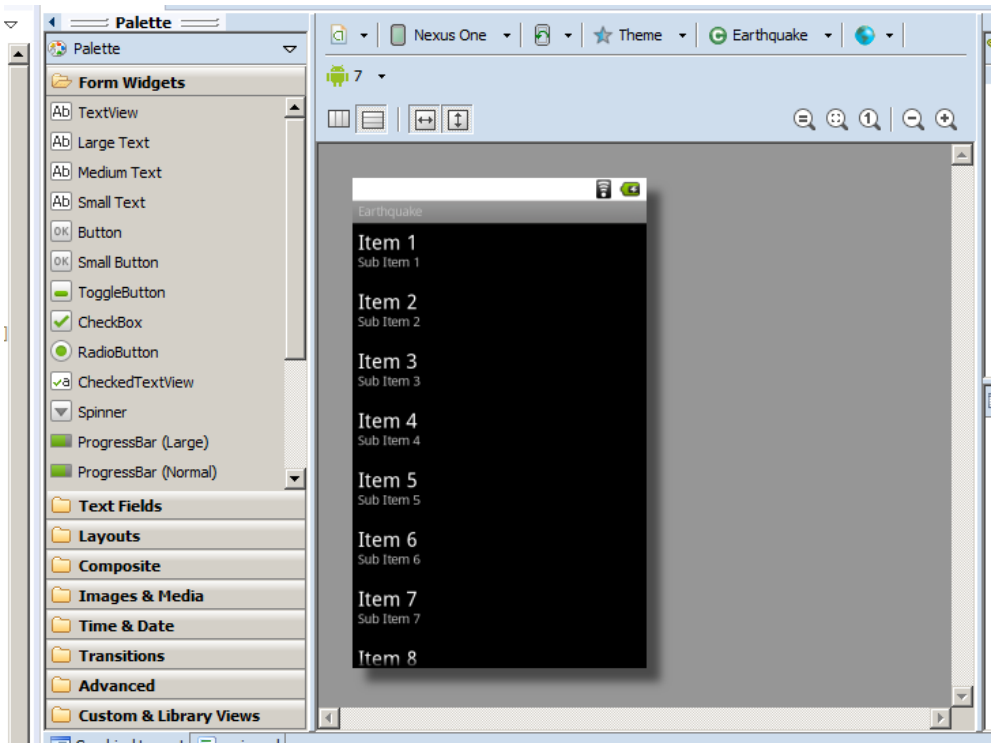

Рис. 2.1. Фрагмент середовища розробки Eclipse з елементом управління ListView.

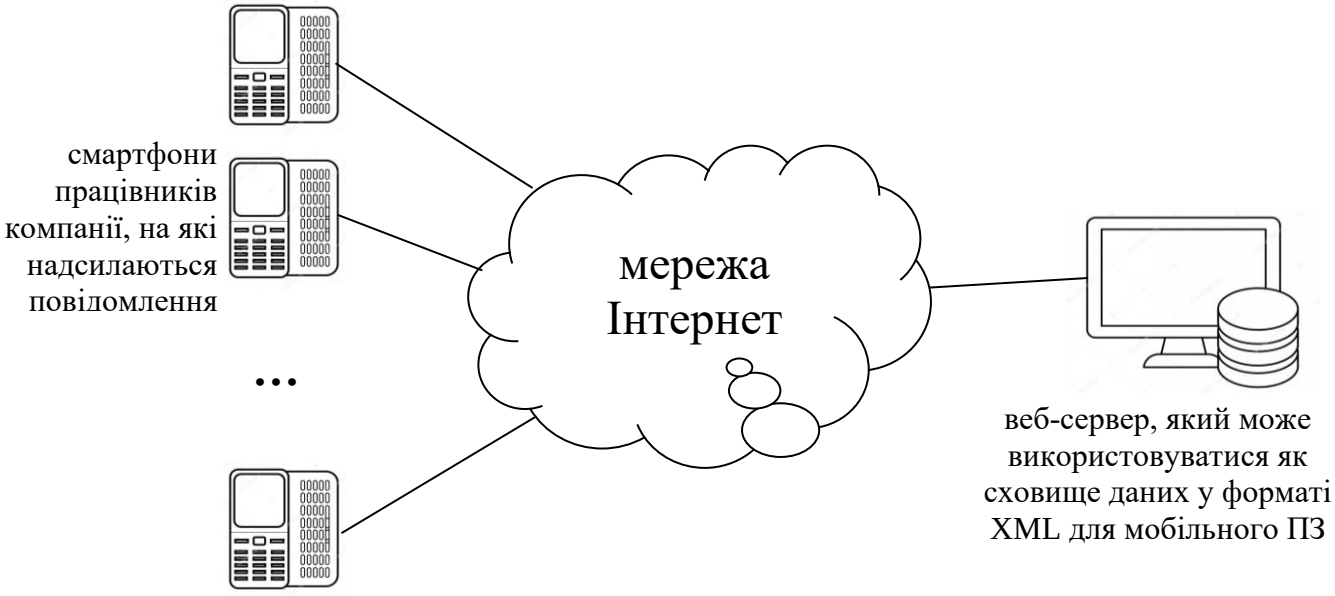

Рис. 2.2. Логічна структура програмного комплексу централізованої служби інформування про події з галузі захисту інформації.

Пояснюючи рис. 2.2, можна відтворити наступний алгоритм роботи усієї системи інформування про події галузі кібербезпеки. Адміністратор системи, що є одним відповідальним працівником служби безпеки компанії (якщо вона велика), або одним із менеджерів (для малих та середніх підприємств), отримує локальну інформацію про новини, пов'язані із захистом інформації, та вносить ці відомості до веб-серверу, який створюється чисто з метою підтримки роботи мобільних програм (на сьогоднішній день практично у будь-якої компанії вже є веб-сайт, тому достатньо додати на ньому лище один єдиний файл із необхідною інформацією; принципово – можна скористатися будь-яким безкоштовним вільним сервісом по розміщенню непрофесійних сайтів, адже повторімо, що на цьому сайті потрібно розмістити лише один XML-файл).

Самі мобільні додатки, працюючи як сервіси, через невеликі інтервали часу (порядку хвилини) перевіряють наявність оновлень на сайті підтримки. За умови появи нової об'яви вона відображується у мобільному додатку, і проінформований адресат може розпочинати реагувати на це повідомлення, в першу чергу, надіслати підтвердження, що він прочитав адресоване йому повідомлення, або ін.

Як уже зазначено вище, інформація зберігатиметься на сервері у форматі XML і вносити до неї зміни можна вручну, набираючи усі необхідні теги, відповідно до інструкції користувача системи, або шляхом використання спеціального веб-додатку, що збирає усю необхідну інформацію через форму (використовуючи зручні для людини елементи управління HTML), а потім автоматично формує необхідний XML-файл, розміщуючи введену змістову інформацію по тегам, які введено у даній роботі. Створення такого веб-додатку може бути розширенням даної роботи, яке виконуватиметься у перспективі (якщо розробка отримає реальне практичне впровадження).

Окрім основного екрану, де будуть списком видаватися усі актуальні повідомлення про особливості захисту інформації в системі, також реалізуємо екран налаштувань – рис. 2.3, що буде досить простим і міститиме два основних налаштування:

- інтервал часу, через який відбуватимуться оновлення інформації (рекомендується – 1 хвилина);

- граничний рівень важливості повідомлень, які будуть відображуватися.

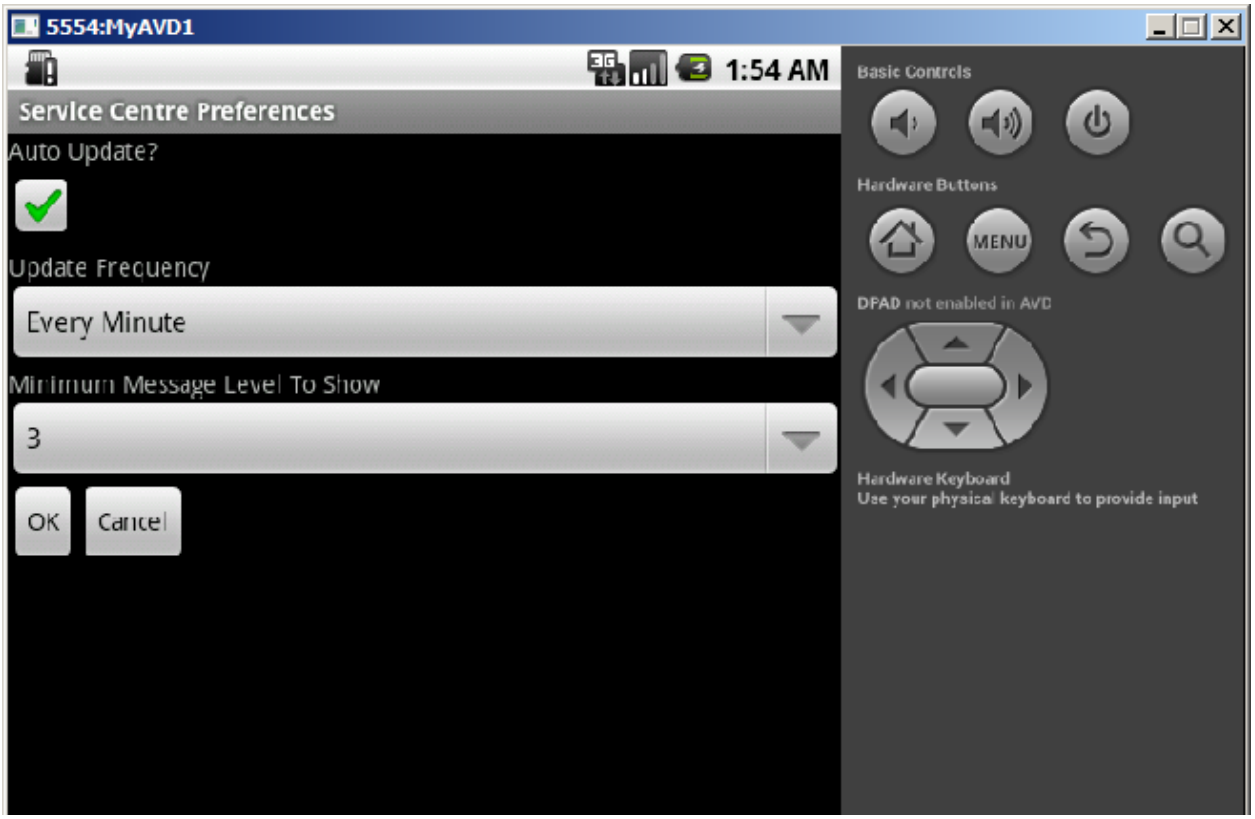

Рис. 2.3. Екран налаштувань мобільного додатку, що розробляється (нижня частина обрізана).

Останнє налаштування слід розглянути додатково. Теоретично можлива ситуація, що в окремі дні (на зразок святкових, наприклад, 31 грудня) система буде використовуватися не зовсім за призначенням, а в тому числі для розсилки другорядної інформації, що має низький пріоритет. Також, деякі оновлення безпеки є лише рекомендованими, а не

обов’язковими, тобто повідомлення про них також матимуть низький пріоритет. Відповідно, слід передбачити ранжування усіх повідомлень за деякою ознакою типу «Важливість» або «Рівень повідомлення» (англ. – Message Level). Це необхідно для того, щоби надати можливість працювати тим робітникам, що виконують якісь термінові роботи з захисту інформації і тимчасово не зважають на свято. Можна наводити і інші причини, коли не весь потік повідомлень має прослуховуватися. Отже, в налаштуваннях додатку має бути передбачена можливість фільтрації усіх повідомлень з рівнем, нижчим за заданий (вони не показуватимуться) – що і видно на рис. 2.3 (у вікні встановлено показувати тільки повідомлення з рівнем 3 і вище, а, відповідно, менш важливі повідомлення з рівнями 1 і 2 – не показуватимуться).

Таким чином, можна сказати, що на вимогу реальних працівників, що забезпечують безпеку організацій та підприємств, інтерфейс додатку виконано мінімалістичним, у вигляді простого списку повідомлень, але в налаштуваннях є опції, що дозволяють змінювати поведінку додатку під конкретного користувача, зокрема, чи проводити автоматичне оновлення інформації (або робити це виключно вручну), з яким інтервалом часу це робити, та якого рівня важливості повідомлення відображати.

Встановивши особливості інтерфейсу користувача, можна сформулювати більш точно вимоги до функціональності мобільного додатку для сервісу централізованого інформування про події в галузі кібербезпеки. Програма повинна вміти:

- завантажуватися на переважній більшості пристроїв з операційною системою Android (для чого вона компілюється під Android 2.1, отже автоматично працюватиме під більш нові версії ОС);

- з’єднуватися по мережі Інтернет з віддаленим сервером та зчитувати з нього поточну версію XML-файлу з повідомленнями;

- відображати повідомлення з рівнем важливості вище заданого;

- відображати більш докладну інформацію про повідомлення при щиглі на ньому;

- мати екран налаштувань, на якому виконуються базові настройки, які мають зберігатися локально;

- мати можливість зміни інтервалу оновлення блоку повідомлень;

- мати можливість зміни рівня важливості повідомлень, що відображаються;

- працювати як сервіс системи Android для того, щоби виводити повідомлення навіть тоді, коли додаток знаходиться не у фокусі.

Керуючись цією функціональністю, можна переходити до такого важливого питання, як вибір технології програмування та засобів розробки програмного забезпечення, що буде виконано у наступному розділі.

Висновки по розділу 2

Таким чином, у даному розділі докладно розглянуто механізми захисту, реалізовані в ОС Android, разом із описом їх недоліків, за рахунок чого можливими стають певні види атак на це програмне забезпечення та усю систему в цілому. Однак, незважаючи на певні можливі вразливості, в цілому рівень технічного захисту цієї ОС весь період її експлуатації і розвитку залишається надзвичайно високим. Через це доцільною є розробка програмного продукту для здійснення організаційного захисту інформації.

Встановлено алгоритмічні особливості побудови комплексного програмного продукту, який являє собою службу централізованої доставки повідомлень, пов’язаних із галуззю кібербезпеки. Запропоновано реалізувати систему у вигляді комплексу із безпосередньо мобільного додатку та веб-ресурсу, на якому відповідальний фахівець викладає повідомлення, що стосуються галузі кібербезпеки (в першу чергу це можуть бути повідомлення інформативного характеру про необхідність встановлення оновлень, оскільки, як встановлено у розділі 1, із цим процесом користувачі часто

мають проблеми, навіть не знаючи про випуск оновлень; повідомлення також може нести інформацію про дисциплінарну або фінансову відповідальність, яку буде нести працівник при ігноруванні необхідності встановлення оновлень, і т.д.). Також, беручи до уваги алгоритми роботи мобільного додатку у даному розділі запропоновано конкретні елементи його інтерфейсу, що будуть доповнені кодом у наступному розділі.

РОЗДІЛ 3. ОСОБЛИВОСТІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ УДОСКОНАЛЕНИХ АЛГОРИТМІВ ЗАХИСТУ ОС ANDROID

3.1. Обґрунтування вибору технологій та засобів розробки

Слід відмітити, що розробка програмного забезпечення для системи Android є дуже специфічним процесом, в першу чергу хоча б тому, що фактично для цього можна застосовувати лише одну єдину мову програмування Java (на відміну від розробки ПЗ для настільних систем, де можна використовувати будь-яку мову загального призначення, яких є сотні). Справедливості заради, відмітимо, що в документації зазначається можливість використання мови C/C++, однак в реальності відсоток усіх програм для Android, написаних цією мовою значно менше одного (< 1 %). Таким чином, мова програмування є визначеною.

Особливістю цієї мови, а отже, і усіх програм для Android є те, що вона фактично не допускає структурної (процедурної) технології програмування, а дозволяє лише об'єктно-орієнтовану розробку. Розглянемо особливості цієї технології докладніше.

Суть технології, або як ще кажуть, методології об'єктно-орієнтованого програмування - в тому, що система розглядається, як сукупність окремих сутностей - об'єктів, які мають набір якихось своїх внутрішніх параметрів - властивостей, а також можуть взаємодіяти між собою за допомогою деяких дій - викликів методів (або трохи більше непрямим чином - шляхом надсилання повідомлень, оброблюваних методами об'єктів; для цього необхідна присутність активної сутності, яка роздає повідомлення адресатам, як, наприклад, менеджер вікон у більшості багатозадачних ОС).

Якщо говорити про програмний код, то для того, щоб оперувати об'єктом, його спочатку потрібно створити. Об'єкти створюються як змінні, у яких типом виступає клас об'єкта. Клас - це просто опис, які властивості можуть мати об'єкти такого типу (тобто яку інформацію вони можуть

зберігати), і які у них є методи (тобто які дії вони можуть виконувати). Об’єкт - це набір значень, чому саме рівні властивості даного об’єкта (свої методи кожен об’єкт отримує від свого класу, тобто методи однакові у всіх об’єктів, що належать даному класу).

Для чого потрібен цей специфічний підхід, адже самі по собі об’єкти не додають нічого корисного (навпаки, введення об’єктів ускладнює програму, вносить в неї нові сутності). Виявляється, реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об’єктів, ми спрощуємо її розуміння для самих себе. Саме тому ОО-підхід рекомендується до застосування для великих проектів (більше десятків тисяч рядків коду), коли утримувати «в голові» всю систему цілком стає важко. Можна сказати, що розбиття (декомпозиція) програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення (в разі «великих» проектів). Людина мислить класами, об’єктами і зв’язками між ними.

Порівняльна складність проектів, що мають однакову функціональність, але побудованих по-різному (згідно структурному і об’єктно-орієнтованого підходів до програмування), як функція їх обсягу показана на рис. 3.1. З графіка рис. 3.1 слідує, що, якщо потрібно реалізувати продукт з невеликою функціональністю (тобто кількість рядків коду, що її реалізують буде очевидно невеликою), то це краще робити без застосування класів, так як вони будуть тільки ускладнювати всю справу. Якщо ж програма має більш-менш значну функціональність, а значить, реалізується хоча б кількома тисячами рядків коду, то вже є сенс замислюватися про застосування об’єктно-орієнтованого підходу. Однозначно будуватися за принципами ООП повинні програми, які мають 10000 рядків коду і більш. Такими є критерії вибору технології програмування, якщо такий вибір взагалі є. В нашому ж випадку доступною в мові Java є тільки ОО-технологія, тому для розробки нашого додатку обираємо її.

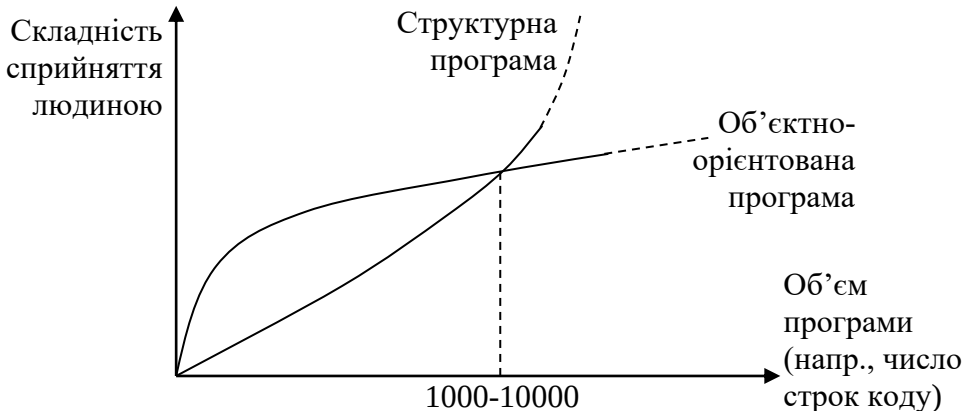

Рис. 3.1. Порівняльна складність висхідного тексту двох програм, що мають однакову функціональність, але реалізованих по-різному: згідно об'єктно-орієнтованому та структурному підходам.

Відзначимо, що часто крім розглянутих міркувань, також на вибір методики програмування впливають інші чинники, наприклад, можливість майбутнього розширення функціональності, створення якомога більш зрозумілого коду (для роботи над проектом цілої команди, а не одного програміста), або просто побажання замовника застосувати найбільш сучасний підхід до програмування.

Крім розбиття (декомпозиції) всієї предметної області на об'єкти (класи) і співвідношення між ними, також ОО-підхід має на увазі дотримання трьох основних його принципів: інкапсуляція, наслідування, поліморфізм.

Під інкапсуляцією мається на увазі об'єднання даних (значення властивостей класу у деякого конкретного об'єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені сутності - класи, які самі обробляють свої дані. Звернення до об'єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

Наслідування дуже корисно, тому що дозволяє сильно скоротити обсяги повторюваного коду (до чого потрібно завжди прагнути при розробці будь-якого програмного забезпечення). Згідно з цим принципом виділяється

клас, який має загальний набір властивостей і методів для декількох більш розширених класів. Цей клас оголошується батьком, базовим класом для декількох похідних від нього (нащадків, спадкоємців). Всі класи-нащадки успадковують від базового всі його властивості та методи, але до цього ще мають свої власні оригінальні властивості і / або методи.

Наприклад, клас Студент є похідним від класу Людина, тому що кожна людина має властивість Ім’я, Прізвище, метод Відпочити(). Однак у Студента є свої специфічні властивості і / або методи, які як раз і відрізняють його від просто Людини: СереднійБал, НомерЗаліковки, ЗдатиЕкзамен(), і т.д.

При наслідуванні іноді методи батьківського і похідного класу мають однакове призначення, але реалізовуються по-різному. Такі методи називаються перевантаженими. Наприклад, метод Відпочити() у класу Людина реалізується як відпочинок на дивані, а у класу Студент - як похід в клуб. При цьому ще раз підкреслимо, що призначення методу в обох випадках одне і той саме.

Поліморфізм є можливістю деякої функції приймати об’єкти як батьківського, так і похідних класів, і вміти викликати перевантажені методи саме того класу, об’єкт якого був переданий в функцію. Слід сказати, що це досить специфічна можливість і в загальному багато програмістів використовують ОО-підхід і без звернення до поліморфізму.

Нехай, наприклад, в нашій програмі є функція ПровестиВихідні(), припустимо яка не належить якомусь класу (хоча це не принципово). Нехай аргументом цієї функції є об’єкт класу Людина. Тоді в неї можна передавати об’єкти всіх похідних від Людини класів: Студент, Службовець, Пенсіонер, і т.д., тому що всі вони є Людиною (спадкоємці цього класу). Ясно, що ця функція повинна включати різні дії: ПрибратиКвартиру(), ПітиНаРинок(), і в тому числі Відпочити(). Так ось поліморфізм дозволяє всередині цієї функції просто вказати назву методу Відпочити(), не вказуючи якого саме класу він повинен бути викликаний, а вже в процесі виконання програми, якщо в

функцію переданий об’єкт класу Студент, то викликається саме його метод Відпочити(), а якщо переданий об’єкт класу Пенсіонер, то автоматично викликається саме його метод Відпочити(), і т.д. Кажуть, що функція ПровестиВихідні() - поліморфна, і вона є такою завдяки тому, що реалізує принцип поліморфізму.

Важливими поняттями в ООП також є: статичні члени класу, абстрактні методи і класи, дружба функцій і класів, і т.д.

Таким чином, будемо використовувати об’єктно-орієнтований підхід як більш сучасний (тобто такий, що краще відповідає віянням у світі програмування) і такий, що підтримується майже всіма сучасними мовами програмування, зокрема, мовою Java, що є де-факто єдиною мовою програмування у системі Android.

3.2. Особливості програмної реалізації удосконалених алгоритмів захисту ОС Android

Відповідно до описаного вище завдання проектування програмна реалізація не міститиме якихось складних алгоритмів по інтелектуальній обробці даних. Суть роботи проектованого додатку полягає у виконанні комунікаційної функції: з’єднанні з сервером, отриманні нової інформації, виведенні її на екран мобільного пристрою. Тому коротко опишемо відповідні прості алгоритми та відповідні програмні складові (реалізацію цих алгоритмів), на основі яких працює додаток.

Отже, розглянемо функції-члени (методи), що реалізують алгоритми (функціональність) проекту. У файлі (класі) `Message`, який уособлює основну базову одиницю запису інформації в проекті - повідомлення, і є найменшим по об’єму є:

1-4) Функції «геттери» для усіх чотирьох властивостей класу (дати повідомлення, детальної інформації про нього, величини важливості повідомлення та посилання, пов’язаного із ним):

```
  public Date getDate() { return date; }
  public String getDetails() { return details; }
  public double getMagnitude() { return magnitude; }
  public String getLink() { return link; }
```

5) Конструктор класу, який є стандартним:

```
public Message(Date _d, String _det, double _mag, String _link) {
    date = _d;
    details = _det;
    magnitude = _mag;
    link = _link;
}
```

6) Функція, необхідна для серіалізації об’єктів класу, тобто така, що зафіксовує особливості переведення об’єкту у текстовий рядок:

```
public String toString() {
    SimpleDateFormat sdf = new SimpleDateFormat("HH.mm");
    String dateString = sdf.format(date);
    return dateString + ": " + magnitude + " " + details;
}
```

Наступним, більш складним по структурі й функціональності класом є клас `Preferences`. Усі поля цього класу мають суто системний характер, і необхідні для посилань (тобто для організації можливості оперування з) елементами управління активності `Preferences`. Методи цього класу мають наступний зміст:

1) Функція-обробник події створення активності:

```
public void onCreate(Bundle icicle) {
```

Тут виконується налаштування активності Preferences перед тим, як її показати та почати використовувати. Тут програмно за допомогою функції `setOnClickListener` встановлюються два обробники для натискань на кнопки ОК та Cancel (зважаючи на динамічний характер створення цих обробників у них навіть немає імені).

2) Окремий метод, який заповнює усі елементи управління типу Spinner доступними значеннями, які задаються окремо у ресурсах (через змінну R):

```
private void populateSpinners() {
```

3) Метод для заповнення елементів управління, що містяться на активності Preferences:

```
private void updateUIFromPreferences() {
```

4) Метод для збереження налаштувань:

```
private void savePreferences() {
```

Найскладнішим і основним класом програмного продукту є клас `ServiceCentre`, який задає головну активність додатку. Він містить наступні властивості:

```
ListView messageListView;
ArrayAdapter<Message> aa;
ArrayList<Message> messages = new ArrayList<Message>();

Message selectedMessage;

int minimumMagnitude = 0;
boolean autoUpdate = false;
int updateFreq = 0;
```

Із них:

- `messageListView` – елемент управління для списку усіх повідомлень;

- `selectedMessage` - об’єкт для збереження одного обраного повідомлення;

- три останніх поля – значення налаштувань проекту за умовчанням (без автоматичного оновлення, показувати усі повідомлення, оновлювати кожної хвилини).

Методи цього класу та їх алгоритмічна суть:

1) Функція-обробник події створення активності:

```
public void onCreate(Bundle icicle) {
```

Тут виконується налаштування активності `ServiceCentre` перед тим, як її показати та почати використовувати. Тут програмно за допомогою функції `setOnClickListener` встановлюється обробник для натискання на одне із повідомлень і видачі детальної інформації про нього (зважаючи на динамічний характер створення цього обробника у нього немає імені).

Тут також завантажуються раніше збережені налаштування додатку – функцією `updateFromPreferences();`.

Тут також присутній виклик функції `refreshMessages();`, яка відображує список усіх повідомлень, що відповідають заданому рівню значимості.

2) Функція, яка безпосередньо зчитує вхідний XML-файл з повідомленнями і відображує їх у списку:

```
private void refreshMessages() {
```

3) Функція додавання нового повідомлення у загальний список:

```
private void addNewMessage(Message _message) {
```

4) Функція-обробник події створення меню налаштувань:

```
public boolean onCreateOptionsMenu(Menu menu) {
```

5) Функція-обробник вибору однієї із доступних опцій програми:

```
public boolean onOptionsItemSelected(MenuItem item) {
```

6) Функція створення діалогового вікна із детальною інформацією про вибране повідомлення:

```
public Dialog onCreateDialog(int id) {
```

7) Функція підготовки вікна відповідного діалогу та заповнення його текстових та інших властивостей:

```
public void onPrepareDialog(int id, Dialog dialog) {
```

8) Функція, що здійснює оновлення роботи програми (зокрема, списку повідомлень) після виходу із активності налаштувань, відповідно до нових налаштувань:

```
private void updateFromPreferences() {
```

9) Функція-обробник події повернення із меню налаштувань:

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
```

Таким чином, розглянуто основні алгоритмічні дії, що виконуються у даному програмному забезпеченні. Ще раз відмітимо, що якоїсь складної серйозної обробки даних технічне завдання не передбачає, тому всі алгоритмічні конструкції зводяться до викликів розглянутих функцій-методів та виконання їх ділянок коду.

Спираючись на основні алгоритмічні конструкції та їх програмні реалізації, розглянуті вище, можна переходити до розгляду конкретної структури проекту мобільного додатку для служби централізованого інформування працівників про події галузі кібербезпеки – рис. 3.2.

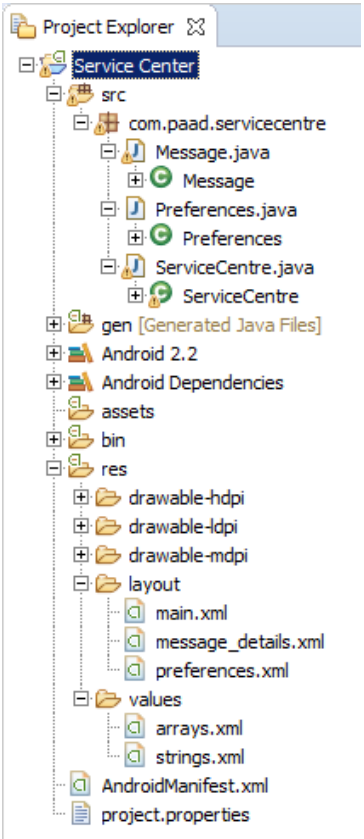

Рис. 3.2. Структура проекту мобільного додатку для служби централізованого інформування працівників про події галузі кібербезпеки.

На рисунку видно, що у складі проекту присутні наступні програмні складові:

а) Пакет `com.paad.servicecentre`, у якому виділено три класи:

- `ServiceCentre`, який відповідає головній активності розроблюваного додатку;
- `Message`, який містить інформацію про одну базову одиницю інформації, з якою працює програма – повідомлення для передачі до співробітників і публікації;
- `Preferences`, що відповідає окремій активності, яка з'являється, коли користувач обирає пункт Preferences у головній активності, і яка відображує усі доступні для налаштувань опції програми.

б) Далі у складі проекту слідує розділ `gen`, який генерується автоматично самим середовищем Eclipse на основі відомостей про ресурси проекту. Цей розділ не слід змінювати вручну, оскільки будь-які виправлення будуть перезаписані.

в) Наступним пунктом іде системна папка Android 2.2 – відповідно до версії SDK, на якій збирається код додатку (в процесі проектування вирішено використати досить стару версію Software Development Kit – сьому – для того, щоби програмний продукт можна було поширювати навіть на дуже старих смартфонах з операційною системою версії не нижче Android 2.2; також це обумовлене тим, що програмний продукт не використовує і не потребує якоїсь складної функціональності, що з'являлася у нових версіях Android, тому із міркувань підвищення переносимості продукту обрано досить стару версію ОС). Цей пункт також змінювати не потрібно.

г) Ще одна системна папка Android Dependencies містить інформацію про програмні залежності даного продукту від інших системних компонентів Android.

д) Порожня папка assets, у якій мають розміщуватися ті файли ресурсів, які не включаються до системної змінної R (в нашому проекті таких файлів немає).

е) Папка bin, що містить двійкові файли проекту. Тут інтерес представляє сам виконуваний файл проекту ServiceCentre.apk, який треба

поширювати і встановлювати на мобільні пристрої працівників компанії, що мають отримувати повідомлення з галузі кібербезпеки.

є) Папка з ресурсами, які включаються до складу змінної R. Тут містяться XML-файли з текстовими рядками проекту.

Таким чином, розглянуто структуру проекту розробленого мобільного додатку і можна переходити до опису методики роботи з ним.

3.3. Тестування та аналіз результатів роботи розробленої системи

Розглянемо покроково методику роботи зі створеним мобільним програмним продуктом для служби централізованого інформування працівників про події галузі кібербезпеки.

1) Для початку роботи слід завантажити програмний продукт Service Centre Messaging, після чого відбувається автоматичне підключення до серверу, вказаного у рядку message_feed файлу strings.xml у розділі res ресурсів програми.

2) Із серверу завантажуються наявні на даний момент повідомлення і відображуються у головному вікні програми.

3) Якщо заголовок якогось повідомлення зацікавив користувача, він може клацнути на ньому, в результаті чого відкриється невелике діалогове вікно з більш докладною інформацією про повідомлення.

4) При натисненні на апаратну кнопку меню смартфону, виникає дві кнопки, що пропонують оновити стрічку повідомлень або увійти до вікна налаштувань системи.

5) Якщо користувач клацне кнопку «Refresh Messages» то відбудеться оновлення стрічки повідомлень, виконане вручну.

6) Якщо користувач клацне кнопку «Preferences» то відкриється нова активність з налаштуваннями.

7) Якщо користувач хоче, щоби виконувалися автоматичні оновлення стрічки повідомлень (логічне побажання за умови, що трафік не є дуже

сильно обмеженим), то він має встановити прапорець «Auto Update» у вікні налаштувань системи.

8) Якщо раніше користувач обрав пункт автоматичного оновлення, то логічним буде завдання інтервалу такого оновлення (менший інтервал підвищує якість роботи додатку, але більший інтервал сприяє економії трафіку смартфону). Для цього слід клацнути пункт «Update Frequency» і у списку задати необхідний інтервал.

9) Якщо користувач хоче, щоби йому показувалися не усі повідомлення, а тільки важливі, то він може клацнути пункт «Minimum Message Level to Show» і обрати рівень важливості повідомлень, які буде відображати йому система (це налаштування актуальне, якщо працівник зайнятий важливою роботою і не хоче відволікатися на другорядні повідомлення, не ігноруючи в той же час критичних повідомлень про найважливіші події в галузі кібербезпеки).

10) Якщо користувач хоче зберегти налаштування додатку, то він має натиснути кнопку «ОК», а якщо він не хоче зберігати зміни – натискає кнопку «Cancel». Після цього вікно з налаштуваннями закривається і здійснюється перехід до основної активності зі списком усіх повідомлень (що мають рівень важливості не нижчий за той, що обрано в налаштуваннях додатку даного конкретного працівника).

11) Якщо користувач хоче закрити додаток, йому слід скористатися стандартною кнопкою «Назад» свого смартфону, або використати інший стандартний метод, за допомогою якого він закриває усі інші додатки на своєму смартфоні (ці методи можуть відрізнятися на різних моделях телефонів).

Так виглядає базова методика роботи з розробленим програмним продуктом.

Слід відмітити ще, що програма для зчитування даних звертається до файлу 1.xml, що розміщений на доступному через мережні з’єднання для даного мобільного пристрою сервері. Під час розробки у якості такого

серверу використовувався звичайний сервер Apache, у якому у кореневому каталозі і було розміщено даний файл з даними. Ім’я серверу test1.ru, що використовувалося при тестуванні системи, прописане у ресурсній константі message_feed, і має бути змінено на ім’я реального серверу, що доступний для розміщення інформації даній компанії.

Розробляє та оновлює цей XML-файл призначений відповідальний працівник служби безпеки (у випадку, якщо компанія має досить великий штат), або звичайний менеджер (призначений відповідальним за вирішення питань захисту інформації) - відповідно до інформаційних новин із галузі кібербезпеки на поточний момент, які відносяться до апаратного чи програмного забезпечення, яке використовується працівниками компанії.

Відповідно до розробленої методики використання створеного програмного комплексу, було проведено дослідження стійкості та адекватності його роботи, іншими словами – тестування продукту.

Робота додатку, традиційно при розробці програмного забезпечення для Android, тестувалася в рамках емуляторів, що є доступними для середовища розробки Eclipse. Для виконання емуляції мобільного пристрою доступною є досить велика кількість опцій повний перелік яких наведено на рис. 3.3.

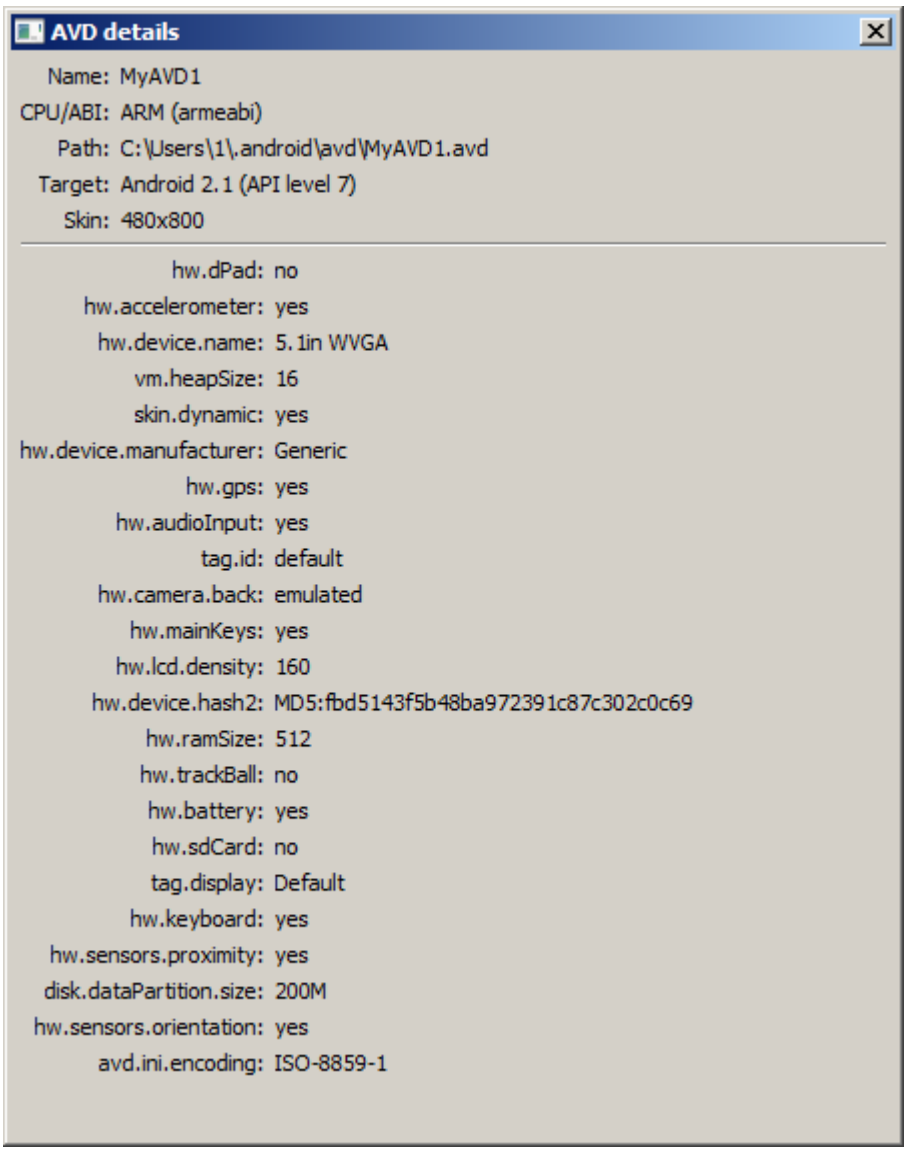

Рис. 3.3. Перелік опцій, що є доступними при створенні емулятора системи Android.

За необхідності властивості віртуального пристрою можуть бути відредаговані – рис. 3.4. Те ж саме вікно викликається при створенні нового віртуального пристрою.

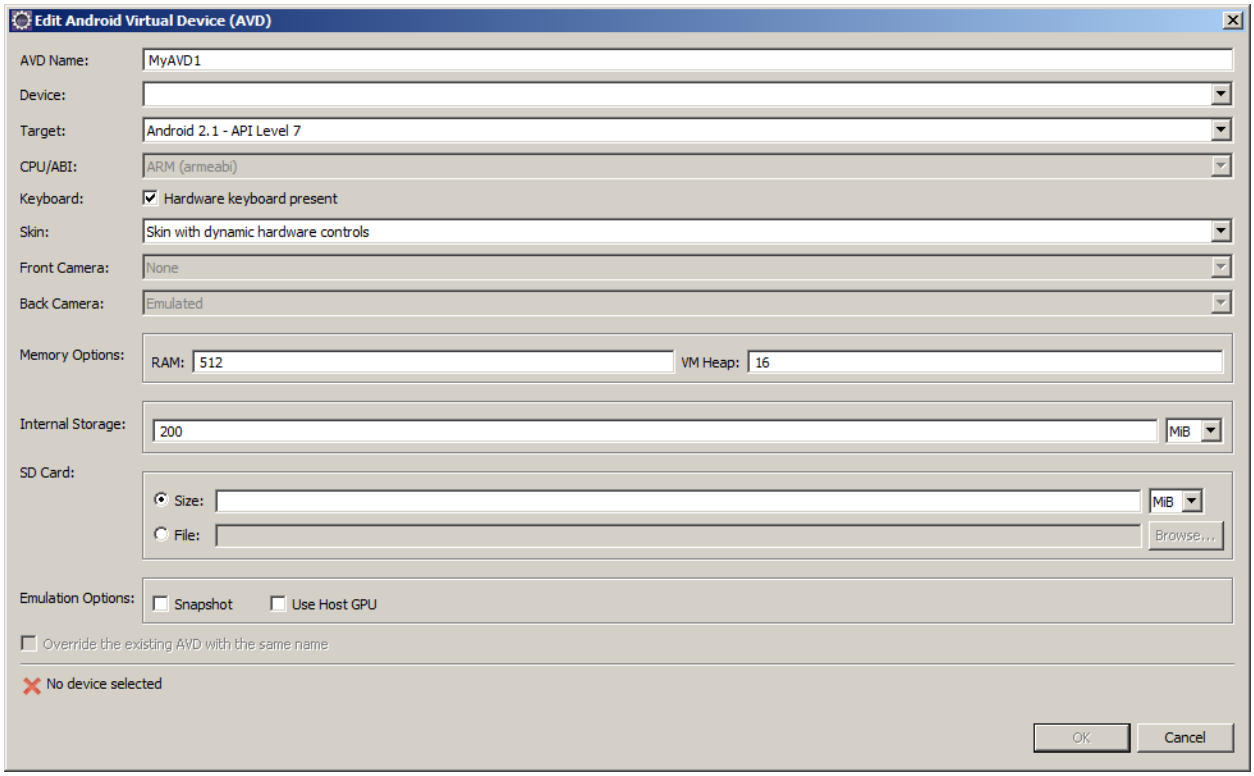

Рис. 3.4. Вікно створення та редагування властивостей віртуального пристрою Android.

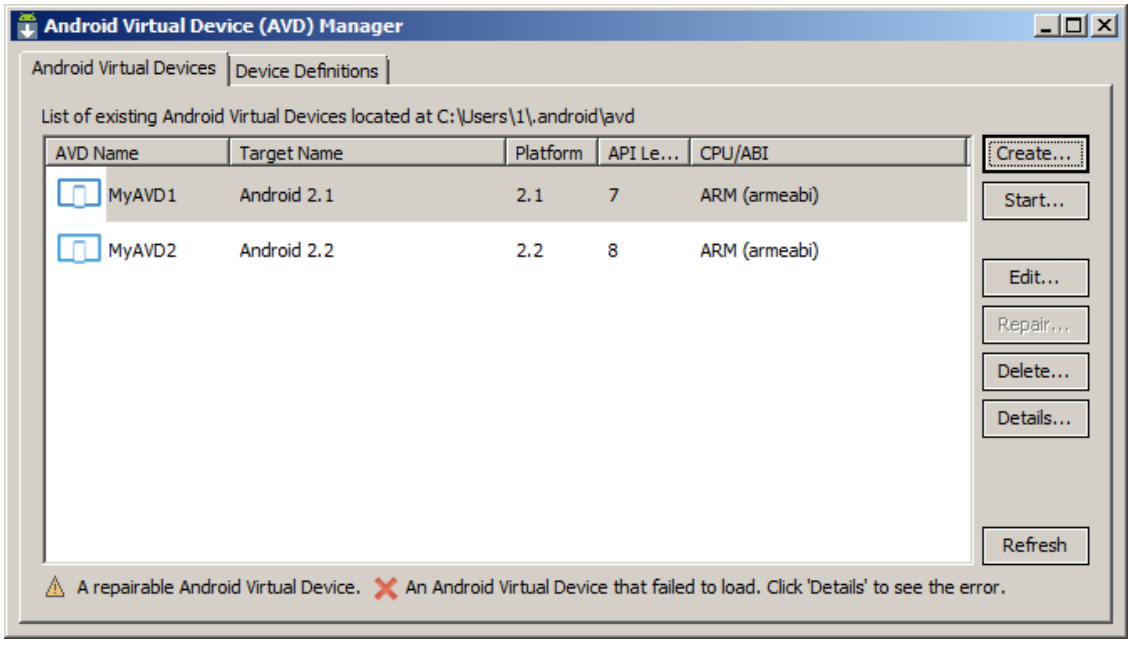

Рис. 3.5. Наявні в системі віртуальні пристрої Android, на яких виконувалося тестування.

Тестування на наявних в системі віртуальних машинах (рис. 3.5) показало стабільну роботу додатку при різних вхідних умовах.

Спостерігалося швидке оновлення інформації при змінах у висхідному XML-файлі. Таким чином, робота розробленого програмного продукту відповідає необхідним вимогам до мобільного програмного забезпечення.

Висновки по розділу 3

Таким чином, у розділі здійснено реалізацію мобільного додатку для служби централізованого інформування працівників компанії про події з галузі кібербезпеки. Програмний продукт створений для сучасної і надзвичайно поширеної операційної системи Android (причому через спеціальним чином виконані налаштування проекту, він може бути завантажений навіть на дуже стару версію ОС 2.2, яка все ще є поширеною серед користувачів мобільних пристроїв – на відміну від версії 1.x, яких практично уже не залишилося). У розділі наводиться аналіз алгоритмів, що покладені в основу роботи додатку, розглядається структура проекту програмного продукту, а також складові його класів: властивості та, в основному, методи. Також тут розроблена методика роботи з системою та проведено тестування, що показало стабільність роботи комплексу та адекватність поставленій задачі.

ВИСНОВКИ

Таким чином, у дипломній роботі виконано проектування та реалізацію мобільного програмного забезпечення на базі поширеної ОС Android, призначене для служби централізованого інформування працівників про події галузі кібербезпеки. Такий вибір обумовлений тим, що розробник операційної системи компанія Google приділяє дуже багато уваги питанням технічного захисту інформації та покращити її виріб саме в даній частині в рамках бакалаврської роботи уявляється малоймовірним. В той же час гарні технічні рішення часто не доходять до кінцевого користувача, оскільки порівняно мало уваги приділяється механізму забезпечення встановлення оновлень операційної системи (зокрема оновлень безпеки). Така ситуація є проблемою організаційного, а не технічного характеру і має вирішуватися розробкою та впровадженням відповідних організаційних заходів захисту. Крім того, існують і інші проблеми організаційного плану, зокрема, мала інформованість користувачів про виявлені нові вразливості та хакерські підходи, причому не тільки технічної частини, а й у галузі соціальної інженерії. Усе це вирішено в рамках даної роботи шляхом впровадження служби централізованого інформування працівників про події галузі кібербезпеки, що реалізована у вигляді програмного комплексу, основною частиною якого є створений мобільний додаток. Встановлено, що для використання саме в галузі захисту інформації доцільно використовувати програмне забезпечення власної розробки, щоби убезпечитися від потенціально наявних «люків» (недокументованих можливостей) у програмах сторонньої розробки.

Програма виконана традиційною для мобільної розробки, повністю об’єктно-орієнтованою мовою програмування Java, у середовищі Eclipse, що раніше використовувалося надзвичайно активно для таких цілей, а на даний час ще продовжує використовуватися, але трохи менше, у зв’язку із просуванням компанією Google власної системи Android Studio (що є нажаль

значно вибагливішою до ресурсів ПК). І однією із переваг використаного підходу якраз є можливість порівняно простої міграції розробленого проекту на цю платформу у майбутньому, за умови, якщо актуальною буде необхідність розвитку даного програмного продукту, а підтримка Eclipse взагалі припиниться.

У роботі виконано проектування програмних складових додатку, реалізовано 3 класи, докладно описуються методи цих класів. Наведено методику роботи з програмою.

В цілому програма може застосовуватися в реальних компаніях та підприємствах для здійснення інформування про критичні події кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Вишня В. Б. Інформаційні системи та технології: підруч. Запоріжжя: ЗНУ, 2021.
2. Гуржій А. М., Возненко Л. І., Поворознюк Н. І., Самсонов В. В. Основи інформаційних технологій: навч. посіб. Київ: Літера ЛТД, 2023.
3. Кравченко І. В., Микитенко В. І. Інформаційні технології: підруч. Київ: КПІ ім. Ігоря Сікорського, 2022.
4. Різник В.О. Виявлення вразливостей Android-застосунків із використанням підходів реверсної інженерії: магіст. дис. К.: НТУУ «КПІ», 2018. 113 с.
5. Сєвєрінов О.В., Федорченко В.М., Перепадя В.І.. Аналіз загроз персональним даним в мобільному пристрої під час використання різноманітних додатків. Системи озброєння та військова техніка, 2016. №4 (48). С. 42-45.
6. Сєвєрінов О.В., Хрєнов А.Г., Поляков А.О. Аналіз сучасних методів атак на автоматизовані системи управління військами та інформаційні мережі. Системи обробки інформації. 2015. № 9. С. 101-104.
7. Aydos M., Kaya K., Gonen S. Security testing of web applications: A systematic mapping study. Journal of King Saud University – Computer and Information Sciences. 2022. 34(10):7652-7668.
8. Ehichoya O., et al. Evaluating security vulnerabilities in web-based applications: a qualitative assessment of scanners. arXiv. 2022. arXiv:2212.12308.
9. Robbins J. N. Learning Web Design: A Beginner’s Guide to HTML, CSS, JavaScript, and Web Graphics. 5th ed. Sebastopol: O’Reilly Media, 2018.
10. Meyer E. A., Weyl E. CSS: The Definitive Guide: Web Layout and Presentation. 5th ed. Sebastopol: O’Reilly Media, 2023.
11. Flanagan D. JavaScript: The Definitive Guide: Master the World’s Most-Used Programming Language. 7th ed. Sebastopol: O’Reilly Media, 2020.

12. Stenberg D. HTTP/3 explained. 2nd rev. ed. 2020. URL: https://http3-explained.haxx.se/en
13. Biørn-Hansen A., Grønli T.-M., Majchrzak T. A Survey and Taxonomy of Core Concepts and Research Directions in Progressive Web Apps. ACM Computing Surveys. 2018. 51(5):108. DOI:10.1145/3241739.
14. Fauzan R., et al. A Systematic Literature Review on Progressive Web Application (PWA) Method Practices. Journal of Theoretical and Applied Information Technology. 2022. 100(19):6674–6697.
15. OWASP Foundation. OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks. 2021.
16. Mohamed A. K. Y. S., et al. A systematic literature review for authorization and access control models in cloud. International Journal of Web Information Systems. 2022. 18(2/3):156-192.
17. Shukla A., et al. System security assurance: A systematic literature review. Journal of Systems Architecture. 2022. 125:102420.
18. 30. Stack Overflow. Developer Survey Results 2019: Web Frameworks Usage. 2019.
19. Jin B., Sahni S., Shevat A. Designing Web APIs: Building APIs That Developers Love. Sebastopol: O'Reilly Media, 2019.
20. McDonald M. Web Security for Developers: Real Threats, Practical Defense. San Francisco: No Starch Press, 2020.
21. Nixon R. Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites. 6th ed. Sebastopol: O'Reilly Media, 2021.
22. Schwalbe K. Information Technology Project Management. 9th ed. Boston: Cengage Learning, 2018.
23. Bărcanescu E. D. Artificial Intelligence and Big Data Analytics in Information Systems. Cham: Springer, 2022.
24. Popović A., Hackney R., Coelho P. S. The impact of big data analytics on firms' high performance. Information Systems Frontiers. 2020. 22(2):337-353.

25. European Commission. European Data Strategy. Brussels: European Union Publications, 2020.

ДОДАТОК 1. ВИСХІДНИЙ ТЕКСТ ОСНОВНОГО ФАЙЛУ SERVICECENTRE.JAVA

```
package com.paad.servicecentre;

import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.GregorianCalendar;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.paad.servicecentre.R;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Resources;
import android.location.Location;
```

```
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.MenuItem;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class ServiceCentre extends Activity {

  static final private int MENU_UPDATE = Menu.FIRST;
  static final private int MENU_PREFERENCES = Menu.FIRST+1;
  static final private int MESSAGE_DIALOG = 1;
  private static final int SHOW_PREFERENCES = 1;

  ListView messageListView;
  ArrayAdapter<Message> aa;
  ArrayList<Message> messages = new ArrayList<Message>();

  Message selectedMessage;

  int minimumMagnitude = 0;
  boolean autoUpdate = false;
  int updateFreq = 0;

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
```

```
    messageListView =
(ListView)this.findViewById(R.id.messageListView);

    messageListView.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView _av, View _v, int _index,
long arg3) {
          selectedMessage = messages.get(_index);
          showDialog(MESSAGE_DIALOG);
        }
      });

    int layoutID = android.R.layout.simple_list_item_1;
    aa = new ArrayAdapter<Message>(this, layoutID , messages);
    messageListView.setAdapter(aa);

    updateFromPreferences();

    refreshMessages();
  }

  private void refreshMessages() {
        // Get the XML
        URL url;

        try {

          String messageFeed = getString(R.string.message_feed);
          url = new URL(messageFeed);
          Toast toast = Toast.makeText(getApplicationContext(),
url.toString(), Toast.LENGTH_SHORT);
            // String.valueOf(responseCode), Toast.LENGTH_SHORT);
     toast.show();
          URLConnection connection;
```

```
          connection = url.openConnection();

          HttpURLConnection httpConnection =
(HttpURLConnection)connection;
          int responseCode = httpConnection.getResponseCode();

          if (responseCode == HttpURLConnection.HTTP_OK) {
            InputStream in = httpConnection.getInputStream();

            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();

            // Parse the messages feed.
            Document dom = db.parse(in);
            Element docEle = dom.getDocumentElement();

            // Clear the old messages
            messages.clear();

            // Get a list of each message entry.
            NodeList nl = docEle.getElementsByTagName("entry");

            if (nl != null && nl.getLength() > 0) {
              for (int i = 0 ; i < nl.getLength(); i++) {

                 Element entry = (Element)nl.item(i);
                Element title =
(Element)entry.getElementsByTagName("title").item(0);
                Element when =
(Element)entry.getElementsByTagName("updated").item(0);
                Element link =
(Element)entry.getElementsByTagName("link").item(0);
```

```
                String details = title.getFirstChild().getNodeValue();
                String hostname = "http://test1.ru";
                String linkString = hostname +
link.getAttribute("href");

                String dt = when.getFirstChild().getNodeValue();
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-
dd'T'hh:mm:ss'Z'");
                Date qdate = new GregorianCalendar(0,0,0).getTime();
                try {
                  qdate = sdf.parse(dt);
                } catch (ParseException e) {
                  e.printStackTrace();
                }

                String magnitudeString = details.split(" ")[1];
                int end =  magnitudeString.length()-1;
                double magnitude =
Double.parseDouble(magnitudeString.substring(0, end));

                details = details.split(",")[1].trim();

                Message message = new Message(qdate, "details", 2,
"linkString");

                // Process a newly found message
                addNewMessage(message);
              }
            }
          }
        } catch (MalformedURLException e) {
          e.printStackTrace();
        } catch (IOException e) {
          e.printStackTrace();
```

```
        } catch (ParserConfigurationException e) {
          e.printStackTrace();
        } catch (SAXException e) {
          e.printStackTrace();
        }
        finally {
        }
      }

    private void addNewMessage(Message _message) {
        if (_message.getMagnitude() > minimumMagnitude) {
          // Add the new message to our list of messages.
          messages.add(_message);

          // Notify the array adapter of a change.
          aa.notifyDataSetChanged();
        }
      }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      super.onCreateOptionsMenu(menu);

      menu.add(0, MENU_UPDATE, Menu.NONE, R.string.menu_update);
      menu.add(0, MENU_PREFERENCES, Menu.NONE,
R.string.menu_preferences);

      return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
      super.onOptionsItemSelected(item);

      switch (item.getItemId()) {
```

```
        case (MENU_UPDATE): {
          refreshMessages();
          return true;
        }
        case (MENU_PREFERENCES): {
          Intent i = new Intent(this, Preferences.class);
          startActivityForResult(i, SHOW_PREFERENCES);
          return true;
        }
      }
      return false;
    }

    @Override
    public Dialog onCreateDialog(int id) {
      switch(id) {
        case (MESSAGE_DIALOG) :
          LayoutInflater li = LayoutInflater.from(this);
          View messageDetailsView =
li.inflate(R.layout.message_details, null);

          AlertDialog.Builder messageDialog = new
AlertDialog.Builder(this);
          messageDialog.setTitle("Message Time");
          messageDialog.setView(messageDetailsView);
          return messageDialog.create();
      }
      return null;
    }

    @Override
    public void onPrepareDialog(int id, Dialog dialog) {
      switch(id) {
        case (MESSAGE_DIALOG) :
```

```
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
            String dateString = sdf.format(selectedMessage.getDate());
            String messageText = "Magnitude " +
selectedMessage.getMagnitude() +
                                 "\n" + selectedMessage.getDetails()  +
"\n" +
                                 selectedMessage.getLink();

            AlertDialog messageDialog = (AlertDialog)dialog;
            messageDialog.setTitle(dateString);
            TextView tv =
(TextView)messageDialog.findViewById(R.id.messageDetailsTextView);
            tv.setText(messageText);

            break;
        }
    }

    private void updateFromPreferences() {
      Context context = getApplicationContext();
      SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(context);

      int minMagIndex = prefs.getInt(Preferences.PREF_MIN_MAG, 0);
      if (minMagIndex < 0)
        minMagIndex = 0;

      int freqIndex = prefs.getInt(Preferences.PREF_UPDATE_FREQ, 0);
      if (freqIndex < 0)
        freqIndex = 0;

      autoUpdate = prefs.getBoolean(Preferences.PREF_AUTO_UPDATE,
false);
```

```
      Resources r = getResources();
      // Get the option values from the arrays.
      int[] minMagValues = r.getIntArray(R.array.magnitude);
      int[] freqValues = r.getIntArray(R.array.update_freq_values);

      // Convert the values to ints.
      minimumMagnitude = minMagValues[minMagIndex];
      updateFreq = freqValues[freqIndex];
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode,
Intent data) {
      super.onActivityResult(requestCode, resultCode, data);

      if (requestCode == SHOW_PREFERENCES)
        if (resultCode == Activity.RESULT_OK) {
          updateFromPreferences();
          refreshMessages();
        }
    }
}
```

ДОДАТОК 2. ВИСХІДНИЙ ТЕКСТ ФАЙЛУ PREFERENCES.JAVA

```
package com.paad.servicecentre;

import com.paad.servicecentre.R;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Spinner;

public class Preferences extends Activity {

  public static final String PREF_AUTO_UPDATE = "PREF_AUTO_UPDATE";
  public static final String PREF_MIN_MAG = "PREF_MIN_MAG";
  public static final String PREF_UPDATE_FREQ = "PREF_UPDATE_FREQ";

  CheckBox autoUpdate;
  Spinner updateFreqSpinner;
  Spinner magnitudeSpinner;

  SharedPreferences prefs;

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.preferences);
```

```
    updateFreqSpinner =
(Spinner)findViewById(R.id.spinner_update_freq);
    magnitudeSpinner =
(Spinner)findViewById(R.id.spinner_message_mag);
    autoUpdate = (CheckBox)findViewById(R.id.checkbox_auto_update);

    populateSpinners();

    Context context = getApplicationContext();
    prefs = PreferenceManager.getDefaultSharedPreferences(context);
    updateUIFromPreferences();

    Button okButton = (Button) findViewById(R.id.okButton);
    okButton.setOnClickListener(new View.OnClickListener() {

      public void onClick(View view) {
        savePreferences();
        Preferences.this.setResult(RESULT_OK);
        finish();
      }
    });

    Button cancelButton = (Button) findViewById(R.id.cancelButton);
    cancelButton.setOnClickListener(new View.OnClickListener() {

      public void onClick(View view) {
        Preferences.this.setResult(RESULT_CANCELED);
        finish();
      }
    });
  }

  private void populateSpinners() {
```

```
    // Populate the update frequency spinner
    ArrayAdapter<CharSequence> fAdapter;
    fAdapter = ArrayAdapter.createFromResource(this,
R.array.update_freq_options,

android.R.layout.simple_spinner_item);
    int spinner_dd_item =
android.R.layout.simple_spinner_dropdown_item;
    fAdapter.setDropDownViewResource(spinner_dd_item);
    updateFreqSpinner.setAdapter(fAdapter);

    // Populate the minimum importance spinner
    ArrayAdapter<CharSequence> mAdapter;
    mAdapter = ArrayAdapter.createFromResource(this,
R.array.magnitude_options, android.R.layout.simple_spinner_item);
    mAdapter.setDropDownViewResource(spinner_dd_item);
    magnitudeSpinner.setAdapter(mAdapter);
  }

  private void updateUIFromPreferences() {
    boolean autoUpChecked = prefs.getBoolean(PREF_AUTO_UPDATE, false);
    int updateFreqIndex = prefs.getInt(PREF_UPDATE_FREQ, 2);
    int minMagIndex = prefs.getInt(PREF_MIN_MAG, 0);
    updateFreqSpinner.setSelection(updateFreqIndex);
    magnitudeSpinner.setSelection(minMagIndex);
    autoUpdate.setChecked(autoUpChecked);
  }

  private void savePreferences() {
    int updateIndex = updateFreqSpinner.getSelectedItemPosition();
    int minMagIndex = magnitudeSpinner.getSelectedItemPosition();
    boolean autoUpdateChecked = autoUpdate.isChecked();

    Editor editor = prefs.edit();
```

```
        editor.putBoolean(PREF_AUTO_UPDATE, autoUpdateChecked);
        editor.putInt(PREF_UPDATE_FREQ, updateIndex);
        editor.putInt(PREF_MIN_MAG, minMagIndex);
        editor.commit();
    }
}
```

ДОДАТОК 3. ВИСХІДНИЙ ТЕКСТ ФАЙЛУ MESSAGE.JAVA

```
package com.paad.servicecentre;

import java.util.Date;
import java.text.SimpleDateFormat;
import android.location.Location;

public class Message {
  private Date date;
  private String details;
  private double magnitude;
  private String link;

  public Date getDate() { return date; }
  public String getDetails() { return details; }
  public double getMagnitude() { return magnitude; }
  public String getLink() { return link; }

  public Message(Date _d, String _det, double _mag, String _link) {
    date = _d;
    details = _det;
    magnitude = _mag;
    link = _link;
  }

  @Override
  public String toString() {
    SimpleDateFormat sdf = new SimpleDateFormat("HH.mm");
    String dateString = sdf.format(date);
    return dateString + ": " + magnitude + " " + details;
  }
}
```