УДК 004.056.57

Володимир Казимир, Ігор Карпачев

ЗАБЕЗПЕЧЕННЯ КОНТРОЛЮ ДОСТУПУ ЗАСТОСУВАНЬ МЕТОДОМ СИСТЕМНОЇ БЕЗПЕКИ НА ОСНОВІ ВБУДОВАНИХ МОДЕЛЕЙ ЗАГРОЗ

Владимир Казимир, Игорь Карпачев

ОБЕСПЕЧЕНИЕ КОНТРОЛЯ ДОСТУПА ПРИЛОЖЕНИЙ МЕТОДОМ СИСТЕМНОЙ БЕЗОПАСНОСТИ НА ОСНОВЕ ВСТРОЕННЫХ МОДЕЛЕЙ УГРОЗ

Volodymyr Kazymyr, Ihor Karpachev

APPLICATION'S CONTROL SUPPLY TO SYSTEM SECURITY METHOD BASED ON EMBEDDED THREAT MODELS

Наведено аналіз існуючих програмних платформ з погляду управління правами доступу до системних ресурсів, сформульовано основні підходи до моделювання загроз з боку застосувань та вирішено питання удосконалення існуючих методів контролю доступу до системних ресурсів за допомогою розробленого методу системної безпеки на основі вбудованих моделей загроз.

Ключові слова: *безпека, безпечні застосування, методи контролю прав доступу, Android, операційні системи, безпечне використання застосувань.*

Рис.: 2. Бібл.: 19.

Приведен анализ существующих программных платформ с точки зрения управления правами доступа к системным ресурсам, сформулированы основные подходы к моделированию угроз со стороны приложений и решены вопрос усовершенствования существующих методов контроля доступа к системным ресурсам с помощью разработанного метода системной безопасности на основании встроенных моделей угроз.

Ключевые слова: *безопасность, безопасные приложения, методы контроля прав доступа, Android, операционные системы, безопасное использование приложений.*

Рис.: 2. Библ.: 19.

In this article there is an analysis provided of existing platforms in terms of their influence on functioning of operating systems, there are main approaches are forming to modeling threats from applications and solving task of improving existing methods of the control access to system resources by using developed method of system security based on embedded threats model.

Key words*: safety, safe application methods to control access rights, Android, operating systems, secure use of applications.*

Fig.: 2. Bibl.: 19.

Постановка проблеми. Для найперших операційних систем з притаманною їм архітектурою існувала достатньо примітивна система контролю прав та рівнів доступу. З роками функціональність програм розширювалася і необхідність у розподіленому контролі дозволів зростала. Сучасні системи стали підтримувати різні методи контролю прав та рівнів доступу застосувань. Незважаючи на це, велика кількість застосувань для сучасних операційних систем підвищує ризик компрометування приватних даних користувачів.

У 1980-х роках більшість комп'ютерів спільно використовувалися кількома людьми. Користувачі писали багато програм, які працювали на загальних комп'ютерах, та іноді шпигували або робили шкоду один одному. Механізми безпеки операційної системи були розроблені для захисту користувачів один від одного, а програми працювали від імені користувача, що їх запустив. Головною метою безпеки було ізолювати користувачів, забезпечуючи контроль обміну між користувачами [1].

У наш час, виходячи з наявності існуючих сервісів розміщення застосувань, користувачі мають спокусу їх легкого завантаження, щоб доповнити роботу своїх смартфонів, комп'ютерів, веб-браузерів та сайтів соціальних мереж. Наприклад, станом на 2015 рік, Google Play налічував більш ніж 10 000,000 застосувань Android, платформа Facebook підтримує більше ніж дев'ять мільйонів, і Apple App Store налічує понад 600,000 ОС IOS застосувань [2]. При цьому застосування, що пропонуються широким колом розробників, можуть створювати небезпеку для користувача. Так, деякі з застосувань активно збирають особисту інформацію про своїх користувачів. Інші шкідливі

програми використовують тактику соціальної інженерії, щоб переконати користувачів встановити їх. Застосування можуть також поміщати користувачів під загрозу зовнішніх (наприклад, мережних) атак, що задають вразливості, оскільки сторонні виробники застосувань, як правило, не є експертами в галузі безпеки [3].

Виникає питання, як допомогти розробникам застосувань уникнути цих загроз, підтримуючи при цьому широкий спектр різноманітних функцій? Традиційні механізми захисту користувачів у цьому випадку більше не підходять. Тому сучасні платформи перейшли до нової моделі безпеки, в якій кожне застосування має свій набір дозволів, заснованих на своїх вимогах. За цією моделлю програми, що керуються дозволами, мають доступ до відповідних системних ресурсів, так що користувачі можуть самі вирішити, чи слід надавати окремим програмам доступ до цих важливих ресурсів [4].

Аналіз останніх досліджень і публікацій. UNIX, VAX/VMS і Multics – три приклади операційних систем розподілу часу. Вони були побудовані з передбаченням загрози зловмисників. UNIX пов'язує біти захисту з кожним файлом, що визначають, які користувачі можуть читати, писати і виконувати цей файл. Коли програма виконується, вона, як правило, може отримати доступ тільки до тих файлів, до яких користувач може отримати доступ. У VAX/VMS користувачі відносяться до одного з семи рівнів привілеїв, і процеси виконуються з рівнями привілеїв, посилаючись на користувачів, якщо інше не вказано системним адміністратором. Multics процеси так само працюють від імені користувача, але співробітники ВВС США виявили, що у Multics був потенціал для забезпечення ізоляції користувача достатньо, щоб зберігати секретні файли військових. Сучасні настільні операційні системи (наприклад, Windows, Mac OS X) успадкували модель управління доступом схожим на UNIX. У результаті, ці платформи як і раніше переважно надають всі права доступу до застосувань користувача [2].

Враховуючи наявність зловмисних користувачів, Сальтцер і Шредер [5] визначили сім принципів проектування систем безпеки: економіка механізму, безвідмовність за замовчуванням, повне посередництво, відкрита конструкція, поділ привілеїв, найменший рівень привілеїв, найменший спільний механізм та психологічний фактор. Ці принципи забезпечили основу сучасної комп'ютерної безпеки. Для прикладу, найменший привілей означає, що кожна програма і кожен користувач системи повинен працювати, використовуючи найменший набір привілеїв, необхідних для виконання завдання. Отже, цей принцип, в першу чергу, обмежує шкоду, яка може виникнути в результаті нещасного випадку або помилки. У свою чергу, психологічний фактор вимагає, щоб інтерфейс був розроблений для простоти використання, надаючи користувачам можливість правильно, регулярно й автоматично застосовували механізми захисту. Хоча ці принципи механізмів безпеки, які захищають користувачів один від одного, були викладенні давно, вони досі актуальні в контексті сучасних шкідливих застосувань [6].

Але основним у формулюванні невирішеної загальної проблеми є статична природа перегляду дозволів при початковому встановленні застосування. При подальшому використанні немає гарантії, що застосування не буде використовувати доступ до будь-яких ресурсів пристрою з шкідливою метою. Подібний недолік дає підґрунтя для маскування намірів застосування під широкий набір дозволів, як, наприклад, це зроблено у файлі маніфесту в ОС Android.

Метою цієї статті є порівняння методів контролю доступу застосувань в сучасних операційних та програмних середовищах, формулювання основних підходів до моделювання загроз з боку застосувань та обґрунтування методу системної безпеки ОС Андроїд.

Порівняння системних платформ управління доступом.

Java. Java-аплети були одними з перших екземплярів мобільного коду. Вони, зазвичай, поширюються як частина веб-сайтів. За замовчуванням аплети непривілейовані.

Для аплетів існувало кілька способів, щоб отримати доступ до додаткових привілеїв. Точний механізм залежав від браузера та версії Java. У деяких випадках користувачі були залучені у процес надання дозволів [7].

CapDesk. Робочий стіл із системою безпеки на основі мандатних посилань. За замовчуванням програми не мають ніяких привілеїв. У CapDesk деякі дозволи (можливості) вимагають схвалення користувача або взаємодії. Застосування на цій платформі, зазвичай, запитують дозволу, коли вони запускаються. Та застосування можуть запросити додаткові дозволи і під час виконання. Розробники повинні запросити можливості з центрального «PowerBox», який розподіляє дозволи у частині застосування, яка вимагає їх [8].

Symbian. Це операційна система для смартфонів. Вона була однією з перших мобільних операційних систем з підтримкою встановлення довільних програм сторонніх розробників. Symbian пропонує послугу підпису застосування, і тільки підписані застосування можуть отримати доступ до критичних дозволів Symbian. Підписані застосування автоматично отримують всі необхідні дозволи без згоди користувача. Проте Symbian пропонує користувачеві надати дозволи під час встановлення непідписаних застосувань. Непідписані застосування можуть запросити тільки кілька дозволів, таких як підключення до мережі. Інші можливості недоступні для непідписаних застосувань. Розробники повинні визначити права доступу, які вимагають їх застосування і в разі необхідності застосовувати процес підписання застосування Symbian [9].

Blackberry. Операційна система для смартфонів, яка також підтримує сторонні застосування. Коли користувач встановлює застосування Blackberry, користувачеві пропонується надати йому «статус довірених застосувань». Довірене застосування може запитувати й отримувати дозволи без запиту користувача, хоча користувач може змінити права доступу, надані в установках пристрою BlackBerry. Застосування, які не належать до Blackberry market, повинні запитувати у користувача отримання певних дозволів. Застосування на цій платформі можуть попросити дозволу в будь-який час, хоча більшість запитують при першому запуску. Дозволи Blackberry охоплюють налаштування телефону, призначені для користувача, Bluetooth, WiFi та ін. Розробники повинні ініціювати запити дозволу, перш ніж намагатися використовувати виклики API [10].

Windows UAC. У Windows Vista та Windows 7 користувачам рекомендується запускати застосування з облікового запису користувача з низьким рівнем привілеїв. Програми запускаються з повними привілеями користувача, але з низьким рівнем привілеїв облікового запису, що не надає їм повного списку привілеїв. Будь-яке застосування може запитувати нові привілеї в службі захисту користувачів (UAC), якщо це необхідно. За допомогою контролю облікових записів діалоговий дозвіл з'являється щоразу, коли програма намагається виконати привілейовану дію. Для того, щоб використовувати контроль облікових записів, розробник повинен перерахувати ресурси контролю облікових записів, які потрібні застосуванню у файлі маніфесту. Діалог буде показано для надання дозволу, коли програма намагається отримати доступ до ресурсу [11].

Facebook. Підтримує сервіс застосувань сторонніх виробників. Зі схвалення користувачів застосування можуть отримувати доступ до даних профілю, друзів та списку активності. Застосування можуть попросити дозволу в будь-який час, хоча більшість просять дозволу під час інсталяції [12].

IOS. Операційна система IOS використовується смартфонами і планшетами. Користувачі ОС IOS можуть запускати сторонні застосування, які вони завантажують з Apple App Store. Apple дуже критично тестує всі застосування перед тим, як вони попадуть до офіційного сервісу App Store. Після цього огляду застосування можуть отримати доступ до більшості привілеїв без схвалення користувачем. Тим не менш, деякі привілеї

вимагають явного дозволу користувача. У версіях ОС IOS 5 і більш ранніх версіях дозволи користувачів управління регулюють доступ до GPS та повідомлень. У IOS 6 права доступу також необхідні для контактів, календаря та бібліотеки фотографій. Система показує діалог запиту привілеїв у перший раз, коли розробник намагається отримати доступ до ресурсів [13].

Bitfrost. One Laptop Per Child (проект OLPC) поширює недорогі ноутбуки для дітей в країнах, що розвиваються. Bitfrost - це модель безпеки для OLPC операційної системи. У Bitfrost застосування є непривілейованими за замовчуванням. Вони повинні запитувати привілеї від користувача. Дозволи можуть бути запитані як при запуску застосування, так і під час виконання [14].

Браузери. Веб-сайти пісочниці в браузері за замовчуванням не можуть отримати доступ до локальних файлів або інших призначених для користувача даних. Однак сучасні браузери починають пропонувати додаткові ресурси для веб-сайтів.

Розширення Google Chrome. Платформа розширень Google Chrome дозволяє створити сторонні розширення для додавання функціональності в браузер та зміни вигляду веб-сайтів. Для того, щоб отримати доступ до веб-сайтів або призначених для користувача даних, розширення Google Chrome повинно запросити дозволу. Розробники перераховують дозволи у файлі маніфесту, а користувачі схвалюють запити дозволів під час інсталяції.

Платформа розширень Google Chrome була першою платформою розширень для браузера, яка мала певну систему дозволів. Firefox та Internet Explorer надають всі розширення з повними привілеями [15].

Android. Операційна система для смартфонів, яка підтримує сторонні застосування. За замовчуванням, Android програми не можуть отримати доступ до важливих призначених для користувача даних або налаштувань пристрою. Якщо застосуванню потрібні такі привілеї, розробник повинен вказати список дозволів, які вимагає застосування. Android інформує користувачів про необхідні дозволи застосуванню під час інсталяції [16].

Windows Phone. Починаючи з Windows Phone 7, користувачам пропонується надавати дозволи під час інсталяції застосувань. Розробники повинні вказати дозволи, що необхідні застосуванню. Під час інсталяції користувачеві показано список дозволів як частина процесу [17]. Застосування не можуть запросити додаткові дозволи під час виконання. Windows Phone 7 має 16 дозволів, які охоплюють апаратні засоби (наприклад, мікрофон) і персональні дані (наприклад, ідентичність телефону). Застосування Windows 8 мають дуже аналогічні обмеження: вони є в пісочниці за замовчуванням, розробникам необхідно вказати будь-які додаткові дозволи, а користувачі просять надати дозвіл [18].

Моделі загроз застосувань.

Моделювання процесу аналізу системної безпеки при використанні застосувань доцільно здійснювати відповідно до логічної послідовності, показаної на рис. 1.

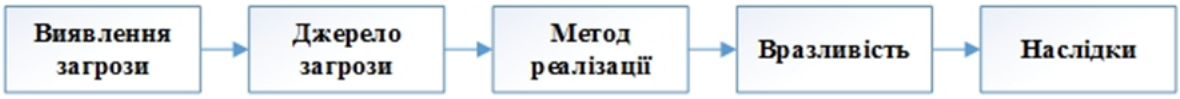

Рис. 1. Процес аналізу системної безпеки

У ході аналізу негативних наслідків загроз необхідно переконатися, що всі можливі джерела загроз ідентифіковані, всі можливі вразливості ідентифіковані та зіставлені з ідентифікованими джерелами загроз, всім ідентифікованим джерелам загроз і відповідним вразливостям зіставлені методи реалізації. При цьому важливо мати можливість, у разі необхідності, не змінюючи загальної моделі, вводити нові види джерел загроз, вразливостей та методів реалізації, які стануть відомі в результаті розвитку знань у цій га-

лузі. Необхідно також враховувати, що джерела загроз системної безпеки можуть знаходитися як усередині системи (внутрішні джерела), так і поза нею (зовнішні джерела). Такий поділ виправдано тому, що для тієї ж самої загрози (наприклад, крадіжка) методи захисту для зовнішніх та внутрішніх джерел можуть бути принципово різними.

У випадку виявлення загроз системній безпеці було використано два підходи. Перший – це аналіз вже існуючих шкідливих мобільних застосувань. Такий підхід дозволяє побачити деякі специфічні особливості застосувань з погляду безпеки. Наприклад, шкідливі програми були лідерами з використання методів обфускації, або заплутування коду [19]. Нині ця ознака вже не така актуальна, оскільки розробники популярних застосувань у процесі побудови використовують різні обфускатори залежно від операційної системи (наприклад, для операційної системи Android включають ProGuard – офіційний обфускатор від Google). Іншою виявленою особливістю є використання дозволів. На відміну від попереднього випадку, ця особливість не втратила своєї актуальності, так як навіть самі прості шкідливі додатки просять дозволу на відправку SMS.

Другий підхід до виявлення загроз базується на вхідному файлі, його розмірі, наборі статичних дозволів, кількості URL у змінних та ресурсах, викликах API.

Пріоритет системної безпеки буває двох видів, базуючись на джерелі інсталяції: високий та низький. Якщо застосування було встановлене з відкритого джерела, то пріоритет буде високий, адже саме тут можливість ураження дуже висока, на відміну від інсталяції з офіційного сайта.

Винятковими називаються загрози, у яких ланцюги викликів повністю збігаються з ланцюгом з бази знать. Такі застосування підлягають негайній деінсталяції. Хоча кінцеве рішення все одно залишається за користувачем. Відповідно, актуальними називаються загрози, які частково збігаються з загрозами з бази знань.

Метод системної безпеки ОС Андроїд. Оскільки не всі системи мають можливість запитувати користувача про привілеї застосування під час його роботи, для ОС Андроїд можна запропонувати метод системної безпеки, який базується на вбудованих моделях загроз у вигляді ланцюгів небезпечних викликів та включає такі етапи:

1. Під час виконання програми та використання певних викликів API Android Application Framework аналізується, чи необхідні цьому виклику додаткові привілеї.

2. Якщо необхідні додаткові привілеї, сканується та записується зліпок стека подальшого ланцюга викликів, що мають відношення до функцій пристрою.

3. Після збирання даних відбувається порівняння ланцюгів викликів з відомими моделями загроз у вигляді небезпечних ланцюгів. База знань потенційно небезпечних послідовностей викликів може знаходитися як локально, так і у хмарі, що допоможе зробити існуючі моделі загроз актуальними для всіх застосувань одразу.

4. Після порівняння ланцюгів викликів розраховується коефіцієнт схожості з відомими небезпечними послідовностями. Коефіцієнт схожості розраховується як відсотковий показник збігів послідовних API викликів. Тобто якщо с ланцюжка-шаблону розміром 10 API викликів 5 збігаються (не обов'язково послідовно), то коефіцієнт схожості дорівнює 50 %.

5. Користувач отримує повідомлення про найбільшу відсоткову схожість з одним із небезпечних застосувань з бази знань та інформацію про можливе неправомірне використання привілеїв. Кінцеве рішення про подальше використання залежить від користувача.

Схема описаного вище методу системної безпеки показана на рис. 2.

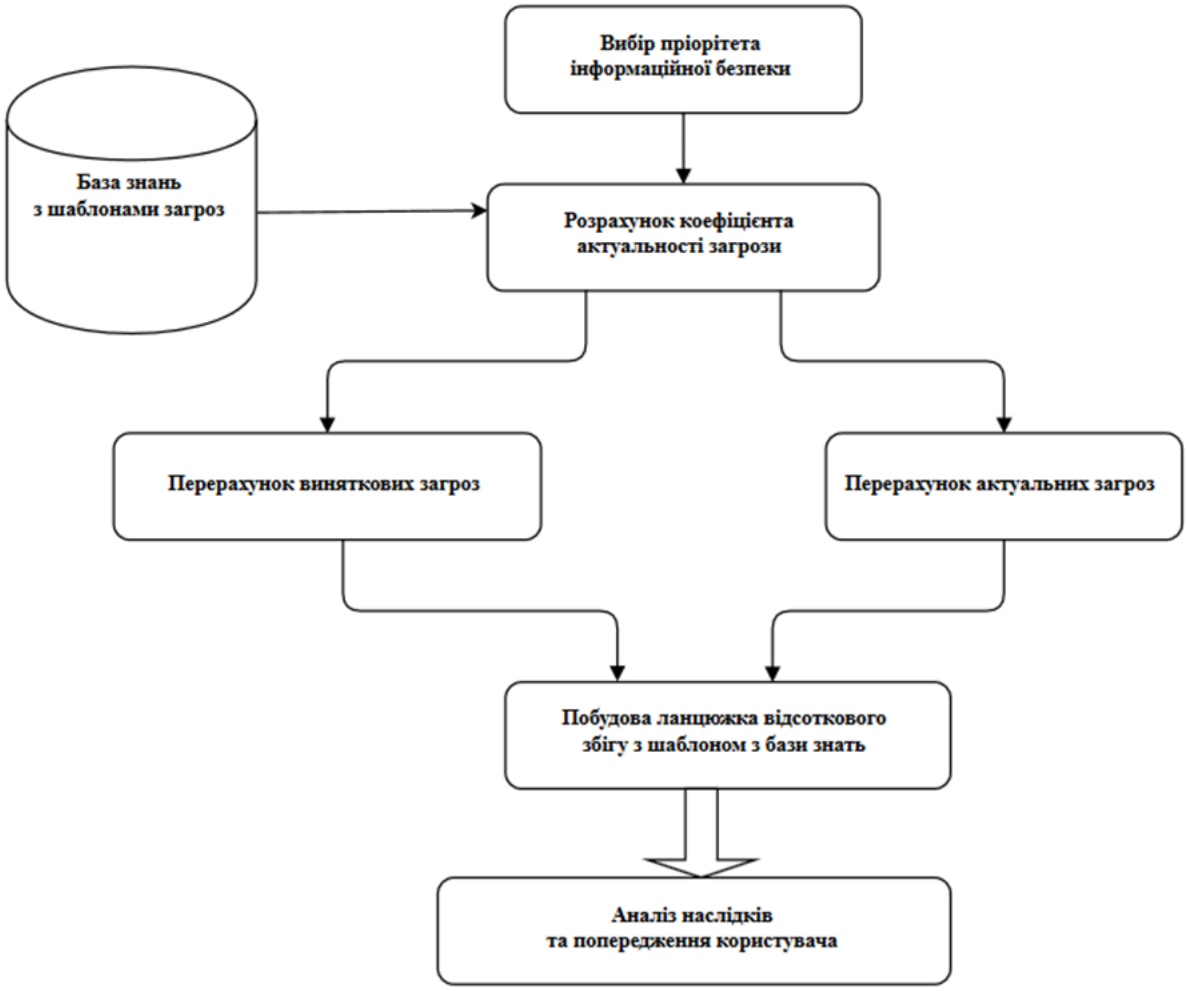

Рис. 2. Схема методу системної безпеки

Запропонований метод уможливлює вдосконалення старих версій операційних систем лише за допомогою встановлення відповідного пакета, який буде слідкувати за роботою потенційно небезпечних застосувань та повідомляти користувача у разі підвищення загрози.

Висновки. Якщо не змінити статичний підхід системи дозволів, то користувач залишається відкритим до зовнішніх та внутрішніх атак. Вирішенням проблеми може бути часткове перенесення рішення про рівень шкідливості застосування на віддалений сервер та давати користувачу прийняти кінцеве рішення на основі результатів отриманого аналізу.

Список використаних джерел

1. Cross-Origin XMLHttpRequest [Electronic resource]. – Access mode: http://code.google-.com/chrome/extensions/xhr.html.

2. Content Security Policy (CSP) [Electronic resource]. – Access mode: http://code.google.com/chrome/extensions/trunk/contentSecurityPolicy.html.

3. Cross-Platform Application Development on Symbian [Electronic resource]. – Access mode: http://www.theseus.fi/bitstream/handle/10024/14566/Thesis_John_Mathew.pdf.

4. Intents and Intent Filters [Electronic resource]. – Access mode: http://developer.android.-com/guide/components/intents-filters.html.

5. Manifest.permission [Electronic resource]. – Access mode: http://developer.-ndroid.com/reference-/android/Manifest.permission.html.

6. Npapi plugins [Electronic resource]. – Access mode: http://code.google.com/chrome/-extensions/npapi.html.

7. Permissions reference [Electronic resource]. – Access mode: https://developers.facebook.-com/docs/authentication/permissions/.

8. SELinux Project Wiki [Electronic resource]. – Access mode: http://selinuxproject.org/page/Main Page.

9. Fundamentals of Symbian OS [Electronic resource]. – Access mode: http://neo.dmcs.pl/symos/wyklady/01a-Introduction.pdf.

10. Tabs [Electronic resource]. – Access mode: http://code.google.com/chrome/extensions/tabs.html.

11. The Add-on Review Process and You [Electronic resource]. – Access mode: http://blog.mozilla.com/addons/2010/02/15/the-add-on-review-process-and-you.

12. Введение в API-интерфейсы Facebook [Електронний ресурс]. – Режим доступу: http://www.ibm.com/developerworks/ru/library/x-androidfacebookapi/.

13. Analysis and Comparison with Android and iPhone Operating System [Electronic resource]. – Access mode: http://www.eecs.ucf.edu/~dcm/Teaching/COP5611Spring2010/Project/AmberChang-Project.pdf.

14. Trusted computer system evaluation criteria (orange book) [Text] // Department of Defense, Tech. Rep. DOD 5200.28-STD, December 1985. – P. 23–31.

15. Device APIs Requirements: W3C Working Group Note 15 October 2009 [Electronic resource]. – Access mode: http://www.w3.org/TR/2009/NOTE-dap-api-reqs-20091015/.

16. How Consumers Interact with Mobile App Advertising [Electronic resource]. – Access mode: http://www.pontiflex.com/download/harrisinteractive.Pdf.

17. US Smartphone Owners by Age [Electronic resource]. – Access mode:http://www.comscoredatamine.com/2011/06/us-smartphone-owners-by-age.

18. *Ackerman, M.* Privacy in e-commerce: examining user scenarios and privacy preferences [Text] / M. Ackerman, L. Cranor, J. Reagle // in Proceedings of the ACM Conference on Electronic Commerce. - 1999. – P. 72–86.

19. Obfuscation of Abstract Data-Types [Electronic resource]. – Access mode:http://www.cs.ox.ac.uk/stephen.drape/papers/thesis.pdf.

References

1. Cross-Platform Application Development on Symbian (2016). Retrieved from http://www.theseus.fi/bitstream/handle/10024/14566/Thesis_John_Mathew.pdf.

2. Content Security Policy (CSP) (2016). Retrieved from http://code.google.com/chrome/extensions/trunk/-contentSecurityPolicy.html.

3. Cross-Origin XMLHttpRequest (2016). Retrieved from http://code.google.com/chrome/extensions/xhr.html.

4. Intents and Intent Filters (2016). Retrieved from http://developer.android.com/guide/components/intents-filters.html.

5. Manifest.permission (2016). Retrieved from http://developer.android.com/reference/android/-Manifest.permission.html.

6. Npapi plugins. (2016). Retrieved from http://code.google.com/chrome/extensions/npapi.html.

7. Permissions reference (2016). Retrieved from https://developers.facebook.com/docs/authentication/permissions/.

8. SELinux Project Wiki. (2016). Retrieved from http://selinuxproject.org/page/Main Page.

9. Fundamentals of Symbian OS (2016). Retrieved from http://neo.dmcs.pl/symos/-wyklady/01aIntroduction.pdf .

10. Tabs (2016). Retrieved from http://code.google.com/chrome/extensions/tabs.html.

11. The Add-on Review Process and You (2016). Retrieved from http://blog.mozilla.com/addons/2010/02/15/the-add-on-review-process-and-you.

12. Vvedenie v API-interfejsy Facebook [Introduction to the Facebook API interfaces] (2016). Retrieved from http://www.ibm.com/developerworks/ru/library/x-androidfacebookapi/.

13. Analysis and Comparison with Android and iPhone Operating System (2016). Retrieved from http://www.eecs.ucf.edu/~dcm/Teaching/COP5611Spring2010/Project/AmberChang-Project.pdf.

14. Trusted computer system evaluation criteria (orange book) (1985) Department of Defense, Tech. Rep. DOD 5200.28-STD, December, pp. 23–31.

15. Device APIs Requirements: W3C Working Group Note 15 October 2009 (2016). Retrieved from http://www.w3.org/TR/2009/NOTE-dap-api-reqs-20091015/.

16. How Consumers Interact with Mobile App Advertising (2016). Retrieved from http://www.pontiflex.com/download/harrisinteractive.Pdf.

17. US Smartphone Owners by Age (2016). Retrieved from http://www.comscoredatamine.com/2011/06/us-smartphone-owners-by-age.

18. Ackerman, M., Cranor, L., Reagle, J. (1999). Privacy in e-commerce: examining user scenarios and privacy preferences, in Proceedings of the ACM Conference on Electronic Commerce, pp. 72-86.

19. Obfuscation of Abstract Data-Types (2016). Retrieved from http://www.cs.ox.ac.uk/stephen.drape/papers/thesis.pdf.

Казимир Володимир Вікторович – доктор технічних наук, професор, Чернігівський національний технологічний університет (вул. Шевченка, 95, м. Чернігів, 14027, Україна).
Казимир Владимир Викторович – доктор технических наук, профессор, Черниговский национальный технологический университет (ул. Шевченко, 95, г. Чернигов, 14027, Украина).
Kazymyr Volodymyr – Doctor of Technical Sciences, Professor, Chernihiv National University of Technology (95 Shevchenka Str., 14027 Chernihiv, Ukraine).
E-mail: vvkazymyr@gmail.com
ORCID: http://orcid.org/0000-0001-8163-1119
Scopus Author ID: 56644727300

Карпачев Ігор Ігорович – аспірант кафедри інформаційних та комп'ютерних систем. Чернігівський Національний Технологічний Університет. (вул. Шевченка, 95, м. Чернігів, 14000, Україна).
Карпачев Игорь Игоревич – аспирант кафедры информационных и компьютерных систем, Черниговский национальный технологический университет (ул. Шевченко, 95, г. Чернигов, 14000, Украина).
Karpachev Igor – PhD student of Department of Informational and Computer Systems, Chernihiv National Technological University (95 Shevchenka Str., 14027 Chernihiv, Ukraine).
E-mail: benchakalaka@gmail.com
ORCID ID: http://orcid.org/0000-0003-1910-3264
ResearcherID: R-3626-2016

Казимир В. Забезпечення контролю доступу застосувань методом системної безпеки на основі вбудованих моделей загроз / В. Казимир, І. Карпачев // Технічні науки та технології. – 2016. – № 4 (6). – С. 131-138.