4. Недоліками алгоритму IDEA є те, що він запатентований, тобто не може вільно поширюватися, а також у ньому не передбачена можливість збільшення ключа.

5. Апаратна реалізація асиметричної криптосистеми ≈ в 1000 разів повільніше за апаратну реалізацію симетричного криптоалгоритму.

6. Програмна реалізація RSA ≈ в 100 разів повільніше DES.

З розвитком комп'ютерних технологій ці оцінки можуть дещо змінюватися, але асиметрична криптосистема ніколи не досягне швидкодії симетричних криптосистем [6].

Список використаних джерел

1. *Зегжда Д. П.* Основы безопасности информационных систем / Д. П. Зегжда, А. М. Ивашко. – М. : Горячая линия–Телеком, 2000. – 452 с.

2. *Диффи У.* Защищенность и имитостойкость. Введение в криптографию / У. Диффи, М. Э. Хэллмэн // ТИИЭР. – 1979. – Т. 67, № 3. – С. 71–109.

3. *Домашев А. В.* Программирование алгоритмов защиты информации / А. В. Домашев, В. О. Попов, Д. И. Правиков. – М. : Нолидж, 2000. – 279 с.

4. *Мельников В. В.* Защита информации в компьютерных системах / В. В. Мельников. – М. : Финансы и статистика, 1997. – 368 с.

5. *Романец Ю. В.* Защита информации в компьютерных системах и сетях / Ю. В. Романец, П. А. Тимофеев, В. Ф. Шаньгин. – М. : Радио и связь, 1999. – 328 с.

6. *Столингс В.* Криптография и защита сетей / В. Столингс. – М. : Вильямс, 2001. – 672 с.

7. *Ященко В. В.* Введение в криптографию / В. В. Ященко. – СПб. : Питер, 2001. – 288 с.

UDC 004.451.9

Volodymyr Kazymyr, Doctor of Technical Sciences
Ihor Karpachev, PhD student
Chernihiv National University of Technology, Chernihiv, Ukraine

FUNCTIONAL SECURITY IN AN ANDROID MOBILE ARCHITECTURE

В.В. Казимир, д-р техн. наук
І.І. Карпачев, аспірант
Чернігівський національний технологічний університет, м. Чернігів, Україна

ФУНКЦІОНАЛЬНА БЕЗПЕКА АРХІТЕКТУРИ МОБІЛЬНОЇ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID

В.В. Казимир, д-р техн. наук
И.И. Карпачев, аспирант
Черниговский национальный технологический университет, г. Чернигов, Украина

ФУНКЦИОНАЛЬНАЯ БЕЗОПАСНОСТЬ АРХИТЕКТУРЫ МОБИЛЬНОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID

Android is currently the most popular operating system used on smartphones. However, users feel their private information is at threat, facing a rapidly increasing number of malware for Android, which significantly exceeds that of other platforms. Antivirus software promises effective protection against malware on mobile devices and many products are available for free or at reasonable prices. Their effectiveness is supported by various reports, attesting very high detection rates. Neither do the exceedingly high numbers of different malware variants reflect the real threat in comparison to other platforms, nor do the results of testing antivirus software against a set of already known malware samples (retrospective tests) provide a clear picture of the capabilities and limitations of antivirus software on the Android platform.

***Key words**: android security, malware protection, viruses, protection software.*

Досліджено проблеми функціональної безпеки, вразливостей мобільних пристроїв не тільки з боку операційної системи, але також з боку стороннього програмного забезпечення.

***Ключові слова:** функціональна безпека, операційна система андроїд, функціональний захист.*

Исследованы проблемы функциональной защиты, уязвимостей мобильных устройств не только со стороны операционной системы, но так же со стороны постороннего программного обеспечения.

***Ключевые слова**: функциональная безопасность, операционная система андроид, функциональная защита.*

Introduction to the topic of the study. The general perception of Android security has been largely shaped by two classes of reports: first, antivirus vendors – as they have access to

the biggest set of malware samples – regularly release threat reports on the state of malware threats for various platforms, including mobile platforms. Second, magazines, specialized journals, companies, and institutes publish test reports of malware product tests. Both reports contribute to the perceived risk level on the Android platform. Risk, in the following case, is a purpose of both the threat and the protection level. However, several issues that have not been addressed in both publication groups that were identified. These issues mainly concern the practical implications and conclusions of these reports for users and their devices' security.

Purpose. The aim of the current research is to conduct an investigation in existing systems of providing functional security and antivirus's specifications. The main focus of the research is to examine as much as possible ways of infection of mobile device. However, a more detailed investigation is required for understanding the real risk level arising from malware for the Android platform.

Methodology, Limiting Factors and Evaluation. Currently AV- Test is one of the most complete and well-known Android antivirus software tests [5]. It deploys the same techniques used for antivirus tests on other platforms such as Microsoft Windows. Most significantly, this is the benchmarking of the retrospective detection rate of all products. While this approach already does not accurately depict the protection level offered by each product at the time of a new malware threat's emergence – as it is retrospective – there are more issues on the Android platform, which make these test methodologies less appropriate for the assessment of the protection level offered by the examined products. The Android platform has fundamentally different approaches for controlling soft- ware access to operating system, device, and other program's resources. On desktop platforms, most software is a priori considered as very trusted and has, once installed, unlimited access to the system's, the users' and other software's data on that system. On Android, however, a file system ensures that each installed application may only have the access to its own data and not any other user or other applications' data, unless explicitly permitted by the user (well-known permission system [4]). Even other directories' contents on an Android device cannot be listed by Android antivirus software.

Furthermore, one of the abilities of the antivirus software on Windows is monitoring file system operations. This way, if a software which has not been visible before suddenly downloads malicious code to the hard-drive, this code will be detected nevertheless, as the download to the hard-drive will be spotted by the antivirus software's on-access scanner. On Android, however, file-system monitoring is not possible as well, as required techniques such as hooking are not allowed to user-installed applications.

Thus, neither full file system scans, nor file-system monitoring can be used by antivirus software on Android. This has severe implications. Otherwise harmless applications may start downloading executables to their working directories and run them – a technique that is not controlled or prohibited by Android. Android antivirus software cannot detect that. Hence, these attacks cannot be covered by retrospective detection rate tests of Android antivirus at all. Such attacks are feasible and can be easily deployed. High detection rates reported by antivirus tests, however, suggest a near-perfect protection of devices with detection rates of 90% and above. In the case of a widely spread malware family with many development iterations, it did not get detected by antivirus software at all at the time of its release, leading other researchers to criticize antivirus software on Android for low effectiveness.

Also, like all retrospective tests, Android antivirus tests fail to reflect how quickly the examined products detect new malware threats. On Android, this is even of bigger importance than on desktop platforms such as Microsoft Windows, as users cannot notice positive infection easily. Removal of malware may even be completely impossible for almost all users, as it might need the reinstallation of the device's software image. This would be necessary, for example, if malware gains higher privileges and installs itself to the system partition of the device, which is

only readable by all other software. Thus, timely reaction to new threats is of absolute importance, as device reinstallation is often not possible, and malware can often remain unnoticed because of antivirus' limitations on the Android platform. Ultimately, Android antivirus software has to resort to two primary sources of information for malware detection:

1. Package database.
2. Package files (APK files) of installed apps.

Package names, and also package file locations are stored in the package database. As Android antivirus software cannot list the contents of the directory with installed packages, it has to rely on the package database to provide it with the locations of installed packages. These package files can then be read to be checked using typical antivirus detection techniques. All files added after installation remains invisible to antivirus software on the Android operation system.

Investigation of existing way of infection. App Markets. Some paid content on the Android market is particularly popular with huge portions of Android users. Thus, a way often taken by malware authors to spread their malicious applications is the provision of free prohibited copies of paid content that are infected with the malware author's malicious code [7]. Users unwilling to pay for such content turn to these pirated copies and in turn get infected, accounting for the majority of all malware infections on Android devices by far.

Websites. Device owners can configure their devices to allow website sources for application installation. Those underlie no restriction or monitoring at all, increasing the risk of installing Trojanized apps. Also, when this option is activated, users can be redirected to fake websites supposedly supplying a "critical update". Based on the user agent identification string of a device's browser, targeted attacks against vulnerable smartphones can be conducted. Rogue networks or attackers may even be able to rewrite web traffic to swap legitimate applications with malicious ones.

Infection via personal computer. Due to a lack of remote exploits for the Android operating system and its security model that prevents vulnerable, compromised apps from modifying any operating system components, device-to-device infections are virtually impossible. This applies for all Android versions prior to Version 3.1, which features USB host mode. USB host mode can be used to infect other Android-based smartphones with USB debugging enabled. Versions prior to Version 3.1 account for around 90% of all Android devices as of May 2014 [1].

Rooting. To date, vulnerabilities are mainly used by users to root their phones, meaning to grant the user full administrative access to their smartphone. Such access is needed for various actions. This includes installation of apps in conflict with the Android security architecture or removing carrier branding, circumventing app or usage limitations (e.g., tethering), or even uninstalling provider-installed spyware. An example of such spyware is Carrier IQ, which is deployed by carriers to retrieve detailed data on customer device usage behavior. It uses rootkit technology to keep its activities unnoticed by users [2]. Furthermore, some users may wish to install modified operating systems on their devices, which is also only possible with privileged access.

Device-to-device infection. Autonomous device-to-device broadcast is currently not possible. With Android versions 3.1 and 4.0, two main changes have been introduced which may serve for device-to-device broadcast: USB host mode (Android 3.1) Android Beam (Android 4.0), an NFC (Near Field Communication) based file and data transmission system with a range of around 10 cm USB host mode is very likely to be usable for malware propagation. Similar to desktop computer propagation, an Android smartphone may use the Android Debug Bridge to push and install malicious applications to other devices with USB debugging enabled. This may happen both deliberately or accidentally: Targeted attacks may be conducted against single individuals. A device owner only has to leave their device out of sight for a short period of time, and an attacker close by may

infect it through plugging a USB cable into it. This requires only few seconds. Accidental device-to-device infections may occur as well. Given an already infected device, propagation code may run invisibly in background, waiting for other Android devices to be plugged in. Once two devices are connected together, the host mode-capable device will imitate the Android Debug Bridge's protocol and infect the other device. Android Beam is of limited utility, as it requires user interaction for installation. For example, a web link to a malicious application can be sent to another Android 4 device via Android Beam, but the user still has to click on the link and confirm it. The limited physical distance reduces malware infection risks even further. A USB host mode has only recently become available, device-to-device propagation has not yet been reported. However, Android 4 comes shipped with an adb-server, which allows remote access via shell on other connected Android devices. As a result, malware can use the resupplied adb program to install applications on other devices. Any difficulties in implementing the adb protocol are thus eliminated. Facilities for device-to-device infections are provided by the Android operating system.

Infection via Rogue Wireless Networks. Open wireless access points are very attractive for smartphone users, as the monthly amount of data, which can be transmitted in current plans, is very limited. Rogue wireless access points have several options to manipulate data traffic sent from or to a user's handheld device. For example, download and installation requests for apps distributed by single websites instead of the official vendor app market can be easily redirected to malicious APK files. During transmission even legitimate apps may be replaced. Alternatively, users logging into the rogue wireless network may be presented with a fake website displaying a "critical update" to an app installed on nearly all devices such as Google Search. Furthermore, the Google Market protocol is capable of forcing devices to install or remove apps through the INSTALL_ASSET and REMOVE_ASSET commands. If it is possible to impersonate Google servers, manipulate transmitted traffic or successfully hijack a session and redirect it to own servers, an attacker might be able to force smartphones into installation of malicious apps. However, further research into this matter has still to be done. Figure below depicts significant increasing of malware software for last few years. Till the beginning of 2012 online banking and other finance apps/actions were not so widespread. But afterword's, users started to store private finance information on mobile devices. At the same time (Q1 2012) intruders started to develop much more applications in order to hack user's information.

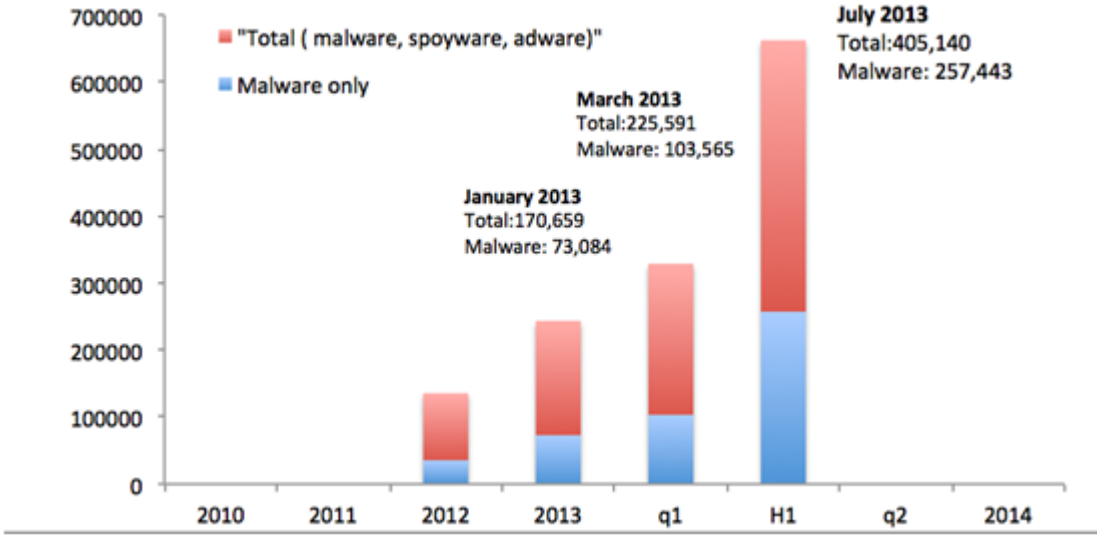

Fig. 1. Increasing of malware software from Q1 2011 to H1 2014

Mobile malware, current propagation scenarios significantly differ from those of desktop malware. Direct self-spreading mechanisms over primary communication networks known from desktop environments are very unlikely. However, different approaches exist, which uti-

lize existing infrastructure such as the Android Market and websites. Also, novel approaches such, as PC-to-device and device-to-device infections will be discussed.

Application sandboxing. Lets consider the process of an Android application installation in details. Android apps are distributed in the form of Android Package (.apk) files. A package consists of Dalvik executable, resources, native libraries and a manifest file, and a developer signature also signs package. There are three main mediators that may install a package on a device in the stock Android operating system:

– Google Play;
– Package Installer;
– ADB install.

Google Play is a special application that provides all the Android smartphones users to look for and download the applications that are uploaded on the market by third-party developers. Although it is also a third-party application, Google Play app (because of being signed with the same signature as the operating system) has access to protected components of Android, which other third-party applications lack for. In case if the user installs applications from other sources she usually implicitly uses Package Installer app. This system application provides an interface that is used to start package installation process. The utility adb install, which is provided by Android, is mainly used by third-party app developers. While the former two mediators require a user to agree with the list of permissions during the installation process, the latter installs an application quietly.

The process of provisioning Application Sandbox at the Linux kernel level is the following. During the installation, each package is assigned with a unique user identifier (UID) and a group identifier (GID) that are not changed during app life on a device. Thus, in Android each application has a corresponding Linux user. User name follows the format app x, and UID of that user is equal to Process.FIRST_APPLICATION_UID + x, where Process.FIRST_APPLICATION_UID constant corresponds to 10000. In Linux, all files in memory are subject for Linux Discretionary Access Control (DAC). A creator or an owner of a file for three types of users sets access permissions: the owner of the file, the users who are in the same group with the owner and all other users. For each type of users, a tuple of read, write and execute (r-w-x) permissions are assigned. Hence, so as each application has its own UID and GID, Linux kernel enforces the app execution within its own isolated address space. Beside that, the app unique UIDs and GIDs are used by Linux kernel to enforce fair separation of device resources (memory, CPU, etc.) between different applications. These architectural decisions set up effective and efficient Application Sandbox on the Linux Kernel level. This type of sandbox is simple and based on the verified Linux Discretionary Access Control model. Luckily, so as the sandbox is enforced on the Linux Kernel level, native code and operating system applications are also subject to these constraints [3].

Threat scenario. Though conventional desktop computer and mobile device malware share many threats for affected users, some are exclusive to mobile platforms due to their use cases and usage environments. This chapter aims to give an overview of threats for smartphones devices. The implications of mobile botnets and the weakness of cellular network environments will be discussed. There is also a consideration and illustration the impact of compromised smartphones on private and corporate targets, e.g., financial fraud and espionage. Due to usage scenario integration on mobile devices – most prominently con- ducting online banking transfers as well as receiving mTANs with the same de- vice, among others – smartphones have become personal communication centers, electronic wallets and even workstations. Use cases slowly evolve towards typical desktop computer fields of application, and even go beyond. Due to the centralization of potentially critical data, platform openness and limited administrative control over a device, as well as resupplied channels for malware distribution, smartphones be-

come highly valuable targets to attack. In the following, attractiveness and threat scenarios will be assessed for private and corporate targets in particular. Figure 2 depicts whole deployment process, vulnerable code could be incapsulated at any time during these steps.

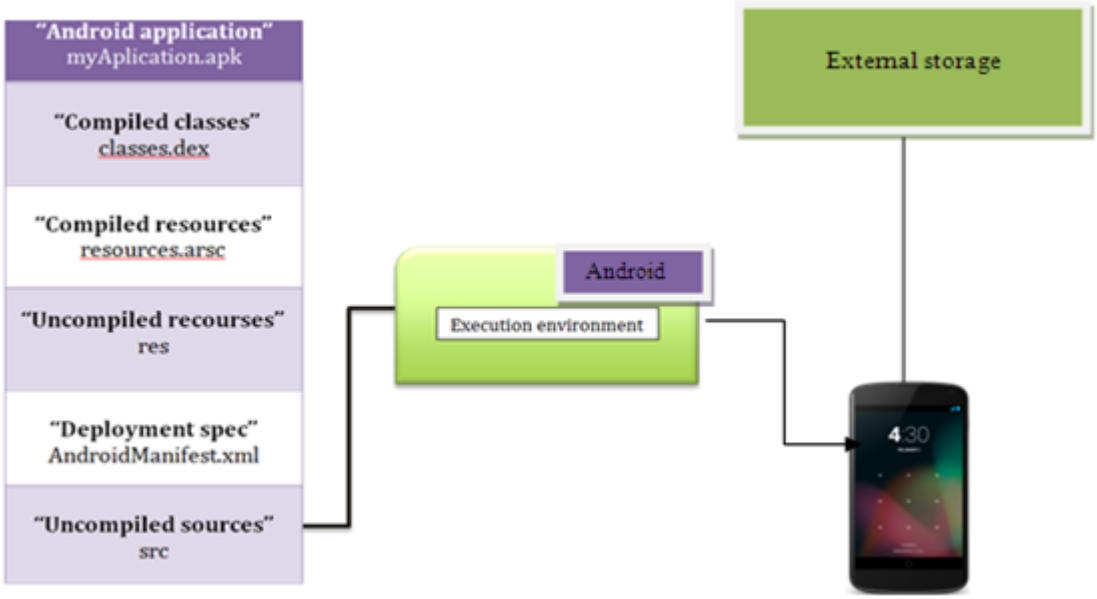

Fig. 2. Android application deployment process

The central Android logging service has proven to be a very rich resource for different personal information. Many apps tend to write status messages to the logging service containing parameters, which disclose personal details of their device owners. For example, several GPS-based apps were found to write the de- vice's geo-coordinates to the logging service in regular intervals, thus providing full profiles on the device owner's movements to other apps installed [6]. Some apps log web requests or other network communication. Thus, by only reading log files, much sensitive information can be gathered, depending on the apps installed and their behavior regarding logging.

Online Banking. Transactions are often confirmed and authorized via the mTAN method. Since the abandonment of printed iTAN lists by many banks, popularity of this method is increasing rapidly, as it is considered easier by customers than TAN generators. In both cases – when a smartphone is only used to receive mTANs or when used to issue transactions as well – the user's bank account is put at high risks, if his device is compromised. On infected phones, login credentials for online banking portals entered by the device owner can be recorded and forwarded easily. This applies when a com- promised smartphone is used for receiving mTANs and for issuing transactions, or even when used only to log in to the banking account to check its balance. Only logging in once is sufficient for an attacker to withdraw all money from an mTAN-protected bank account, as ordering a transaction and intercepting the mTAN text message is trivial. Under the Android operating system, an application can register to receive SMS messages before the phone's own messaging application through the RECEIVE_SMS permission. An attacker only would have to react quickly enough to confirm the transaction after parsing the respective SMS message via the READ_SMS permission, even without preemptive interception.

Modules of generation of hidden events. Contact Information, Location Data, Credentials and Private Details. Espionage, communication eavesdropping, blackmailing, botnet formation, gathering valid email addresses for spam mailing and recording of sensitive information such as login credentials or credit card data are just some of the classical applications of Trojan software. All of these apply for mobile platforms as well. In some cases, they may even be more dangerous on mobile devices: Users are less cautious and store a lot of in-

formation and communication together. For convenience reasons, application credentials are reserved unencrypted or only obfuscated [7]. When encrypted, the corresponding key is usually saved in plain text and easily available. This way, users are not forced to enter passwords on a regular basis. As a result, obtaining credentials for any app is trivial, given root privileges. Multiple real-world spying cases have gained some attention recently and similar scenarios will become more common, due to thriving smartphone numbers and increasingly sophisticated attacks. Best known is the News of the World eavesdropping case in which the Murdoch-owned newspaper spied on celebrities and politicians as well as on abduction or even murders victims. Although these targeted privacy breaks were not technically sophisticated, they demonstrate very well what can be done with mobile spyware. Through the possibility of targeted infection of end-user devices, single persons can be attacked successfully in public places such as cafés during short moments of inattentiveness. When targeting persons of political, military or corporate power or interest these attacks can be highly effective.

Corporate Targets. One of the biggest threats for medium to large sized enterprises and even to whole economies is industrial espionage. Employee and management workstations usually underlie close monitoring and are supplied with security patches centrally and quickly. Corporate security and data policy, as well as technology, such as firewalls, intrusion, detection systems and content filters, account for usually very high security standards. They aim to prohibit data breaches and compromisation of employee workstations.

However, corporate-supplied smartphones are often less well monitored. Central administration without removing all end-user privileges on their devices is an issue of very high difficulty. Private smartphones usually are not monitored at all. However, they are often used to enter a company's networks, store sensitive work-related documents, carry out work-related communication or are plugged into workstations' USB ports for battery charging. Such usage may not only result in data breaches where an attacker is able to copy contracts, product designs or other mission-critical documents through a compromised smartphone. It can also lead to the infiltration of the corporate network. Attackers may use an infected mobile device, as a base of operations for mapping the company's structure, when logged into the company's network, for intercepting inside data traffic or for other forms of attacks and eavesdropping. Company-provided smartphones often contain VPN login identifications, giving an attacker corporate network access at will. Furthermore, operating system security vulnerabilities in USB device management allow for the infection of computers with malware when a compromised smartphone is plugged in, e.g., for charging its battery.

Conclusions. Like on desktop and server systems, private and corporate users should not rely on antivirus products for perfect protection of their Android devices. While this is a well-known fact, it is even more important on Android due to the limited capabilities of antivirus apps on this platform. In practice, antivirus software on Android can only offer more limited security than on desktop and server platforms. Smartphone security begins with education designed for the smartphone user about existing security controls that can provide especially protection when conducting financial transactions. Android has an operating system that is very secure, it has several levels of protection to tackle a broad range of security threats and requires users permission to do almost anything that could lead to users data or the system to be compromised. Moreover, the SELinux solution, which protects against weaknesses on the Linux-kernel layer, have been implemented in Android 4.3 However, in a helpless area such as the one mobile devices represent, a system can never be considered perfect. This is why we pointed out some weak parts in Android security and exposed several countermeasures along with their corresponding mitigation level and required implementation effort. Definitely, the platform needs to improve its permission mechanism and its capacity of detecting misuse of granted permissions. So, a good idea would be to assign the highest priority to SELinux along with additional countermeasures such as fire- wall, IDS

and CAAC. At lower priority we could add spam filtering, data encryption, and selective Android permission mechanisms. Finally, remote management, VPN and login solutions would be a significant advantage when targeting corporate customers.

References

1. *Android* developers. Dasboard & Platform versions [Online]. – Available : https://developer.android.com/about/dashboards/index.html.

2. *Understanding* security on Android [Online]. – Available : http://www.ibm.com/developerworks/library/x-androidsecurity.

3. *Android* security (And not) internals / Y. Zhauniarovich ; Toronto University. – June 2012. – P. 11–14, 32–37.

4. *Effectivness* of malware protection on android and evaluation of android antivitus apps / R. Fedler, J. Schutte, M. Kulicke // Applied and integrated security. – 2014. – P. 7–13, 26–32.

5. *Enhanced* android security to prevent privilege escalation / J. Majer // Munich University. – September 16. – 2013. – P. 4–5, 22–31.

6. *Android* OS security: risks and limitations / R. Fedler, C. Banse, C. Kraub, V. Fuesing. – 2012. – P. 19–27.

7. *Mobile* application security on android, context on android security / J. Burns // Black Hat. – 2009. – 27 p.

UDC 004.5:004.89:004.9

Vitalii Lytvynov, Doctor of Technical Sciences
Chernihiv National University of Technology, Chernihiv, Ukraine
Olena Skakalina, PhD in Technical Sciences
Poltava National Technical Yuri Kondratyuk University, Poltava, Ukraine

METHODS OF FUZZY LOGIC TO MINIMIZE THE RISKS OF IT-PROJECTS

В.В. Литвинов, д-р техн. наук
Чернігівський національний технологічний університет, м. Чернігів, Україна
О.В. Скакаліна, канд. техн. наук
Полтавський національний технічний університет, м. Полтава, Україна

МЕТОДИ НЕЧІТКОЇ ЛОГІКИ ПРИ МІНІМІЗАЦІЇ РИЗИКІВ ІТ-ПРОЕКТІВ

В.В. Литвинов, д-р техн. наук
Черниговский национальный технологический университет, Чернигов, Украина
Е.В. Скакалина, канд. техн. наук
Полтавский национальный технический университет, Полтава, Украина

МЕТОДЫ НЕЧЕТКОЙ ЛОГИКИ ПРИ МИНИМИЗАЦИИ РИСКОВ ИТ-ПРОЕКТОВ

This article describes the selection of an IT project using fuzzy sets and project risk management. Proposed for determining the most effective use of fuzzy logic project «worst-case method», which is based on the principle of fuzzy intersection Bellman criteria-Zadeh and 9-point scale linguistic evaluations Saaty.

Key words: *fuzzy set, fuzzy logic, fuzzy variable, linguistic variable, IT project risks IT-project management standards IT-projects.*

Розглянуто вибір ІТ-проекту за допомогою нечітких множин та управління ризиками проекту. Запропоновано для визначення найбільш ефективного проекту використання нечіткої логіки «метод найгіршого випадку», основу якого складають принцип перетинання нечітких критеріїв Белмана-Заде і 9-бальна шкала лінгвістичних оцінок Сааті.

Ключові слова: *нечітка множина, нечітка логіка, нечітка змінна, лінгвістична змінна, ІТ-проект, ризики ІТ-проектів, стандарти управління ІТ-проектів.*

Рассмотрен выбор ИТ-проекта с помощью нечетких множеств и управления рисками проекта. Предложено для определения наиболее эффективного проекта использования нечеткой логики «метод наихудшего случая», основу которого составляют принцип пересечения нечетких критериев Беллмана-Заде и 9-балльная шкала лингвистических оценок Саати.

Ключевые слова: *нечеткое множество, нечеткая логика, нечеткая переменная, лингвистическая переменная, ИТ-проект, риски ИТ-проектов, стандарты управления ИТ-проектов.*

Statement of Problem. At the present stage of information technology (IT) is considered as one of the main tools for the implementation of the strategic objectives in various fields.