Технічні науки

УДК 004.451

Григор'єв Василь Ілліч
архітектор рішень
"EPAM systems" (Київ, Україна)
Григорьев Василий Ильич
архитектор решений
"EPAM systems" (Киев, Украина)
Grygoryev Vasyl
Solution Architect
"EPAM systems" (Kyiv, Ukraine)

СИСТЕМИ КЕРУВАННЯ ПАКУНКАМИ ДЛЯ ДИСТРИБУТИВА DEBIAN: БАЗОВА АРХІТЕКТУРА І ВИСОКОРІВНЕВІ УТИЛІТИ
СИСТЕМЫ УПРАВЛЕНИЯ ПАКЕТАМИ ДЛЯ ДИСТРИБУТИВА DEBIAN: БАЗОВАЯ АРХИТЕКТУРА И ВЫСОКОУРОВНЕВЫЕ УТИЛИТЫ
PACKAGE MANAGEMENT SYSTEMS FOR THE DEBIAN: BASIC ARCHITECTURE AND HIGH-LEVEL UTILITIES

Анотація. *Розглянуто базову архітектуру системи керування пакунками та менеджерами пакунків дистрибутиву Debian. Зазначено типові правила безпеки роботи з пакувальником, вказано на ризики не дотримання даних правил, зазначено підходи, що дозволяють зменшити їх вплив. Проведено аналіз рівнів залежності між пакунками, за рахунок чого можна визначити характер взаємодії встановлених програмних додатків. Запропонована базова модель життєвого циклу пакувальника, визначено спеціалізовані команди, що можуть бути не включені до даної моделі. Зазначено відповідності рівнів команда, пакунок та утиліта. Розроблено*

схема керування пакунками та менеджерами пакувальників, що дозволяє визначити вимоги для безпечної роботи з пакунками.

Ключові слова: *дистрибутив Debian, пакувальник, менеджер пакунків, рівні залежності, apt, dpkg, життєвий цикл, високорівневі утиліти.*

Аннотация. *Рассмотрена базовая архитектура системы управления пакетами и менеджерами пакетов дистрибутива Debian. Указаны характерные правила безопасности работы с упаковщиком, показаны риски несоблюдения данных правил, предложены подходы, позволяющие уменьшить их влияние. Проведен анализ уровней зависимости между пакетами, за счет чего можно определить характер взаимодействия установленных приложений. Предложена базовая модель жизненного цикла упаковщика, определены специализированные команды, которые могут быть не включены в данную модель. Указаны соответствия уровней команда, пакет и утилита. Разработана схема управления пакетами и менеджерами пакетов, которая позволяет определить требования для безопасной работы с пакетами.*

Ключевые слова: *дистрибутив Debian, упаковщик, менеджер пакетов, уровни зависимости, apt, dpkg, жизненный цикл, высокоуровневые утилиты.*

Summary. *The basic architecture of the package management system and package managers of the Debian is considered. Specific safety rules for working with a package are indicated, the risks of non-compliance with these rules are shown, and approaches to reduce their impact have been proposed. The analysis of the levels of dependencies between packages was carried out, so to determine interaction of installed applications. A basic model of the life cycle of the package utilities has been proposed, specialized operations, which may not be included in this model have been identified. The command, package, and utility*

levels relations were indicated. A package and package managers organizing scheme has been developed, it allows defining requirements for safe work with packages.

Key words: *Debian, packaging, package manager, dependency levels, apt, dpkg, life cycle, high-level utilities.*

Вступ. Debian є розповсюдженою UNIX-подібною операційною системою (ОС), основний дистрибутив якої складається з вільного програмного забезпечення (ПЗ), що надає можливість користувачу модифікувати початковий програмний код та вільно розповсюджувати модифікації [1-2]. При цьому слід зазначити, що ОС Debian працює з широким спектром ПЗ, частину якого не можна віднести до вільного ПЗ, зокрема невільне ПЗ (Non-Free), що має обмеження у розповсюдженні та доступі до програмного коду і додаткове ПЗ (Contrib), що залежать від невільного ПЗ. Це зазначається у «Соціальному Контракті Debian» [3], при цьому користувач попереджається про відсутність свободи у редагуванні цих програмних пакетів та відсутність їх підтримки з боку служб Debian, що також не можуть аналізувати відповідний програмний код і корегувати його у разі виявлення помилок. «Debian GNU/Linux» можна розглядати як багатоцільовий дистрибутив, що є основою для ОС робочих станцій та мобільний пристроїв, наприклад ядер «Debian GNU/Hurd», «Debian GNU/kFreeBSD», «Debian GNU/kNetBSD» та ін.

Типовий пакет інсталяція для робочої станції включає графічне середовище, додатками для роботи з периферійним обладнанням апаратної платформи, офісний пакет, браузер, графічний редактор та ПЗ для роботи з документами і медіа файлами. Окремим функціональним блоком дистрибутиву «Debian GNU/Linux» є система керування пакунками [4-5], що включає у себе набір ПЗ для керування процесом установки, вилучення, налаштування і оновлення компонентів програмних додатків. Згідно

основної парадигми UNIX-подібних ОС ПЗ представляється у вигляді пакунків, що містять програмний код та набір метаданих (повне ім'я пакунку, номер версії, опис пакунку, ім'я розробника, контрольну суму, взаємодії з іншими пакунками, та ін.). Система керування пакунками є досить низькорівневою утилітою, що супроводжується високорівневими утилітами, так званими менеджерами пакунків (package manager), що відстежують зв'язки між пакунками та представляють результати аналізу кінцевому споживачу ПЗ.

Система керування пакунками ПЗ ОС Debian з одного боку характеризується жорсткими правилами роботи з пакунками, а з іншого позиціонується як ефективний та зручний засіб установки, налаштування та оновлення програмних компонент через забезпечення прозорості процесів розробки і тестування [6-7]. Таким чином для забезпечення безпечної та ефективної роботи з даною системою достатньо знати її базову архітектуру та особливості застосування високорівневих утиліт.

Аналіз останніх досліджень і публікацій у цій галузі вказав на те, що система керування пакунками та менеджерами пакунків ОС Debian є ефективним засобом виділення даних і ПЗ пакунків в автономні одиниці, з їх подальшим встановленням, корегування і видаленням, а також контролю системних ресурсів, цілісності програмних додатків і конфліктів між пакунками [8-10]. Було показано, що забезпечення ефективної роботи даної універсальної системи базується на обізнаності користувача у її архітектурі та досвіді роботи з високорівневими утилітами [1-3; 6-10].

Проведений аналіз вказав на актуальні підходи до організації роботи системного адміністратора з системою керування пакунками та показав необхідність побудови цілісної методології визначення типових факторів ризику, що у рамках цієї роботи **виділяється як невирішена частина загальної проблеми**.

Метою дослідження, таким чином, є визначення оптимального підходу при побудові та застосуванні наявних алгоритмів керування пакунками, через аналіз обмежень, що накладають виконання правил безпеки багатоцільового дистрибутиву «Debian GNU/Linux».

Базова архітектура система керування пакунками Debian. Система керування пакунками Debian при належному рівні обізнаності користувача дозволяє ефективно та безпечно встановити ПЗ у ОС з бінарних пакунків, через використання фронтенд- та бекенд-утиліт. Базовою утилітою програмно-апаратної частини сервісу керування пакунками Debian є «dpkg», по відношенню до якої набір «apt» є універсальною фронтенд-утилітою (рис. 1).

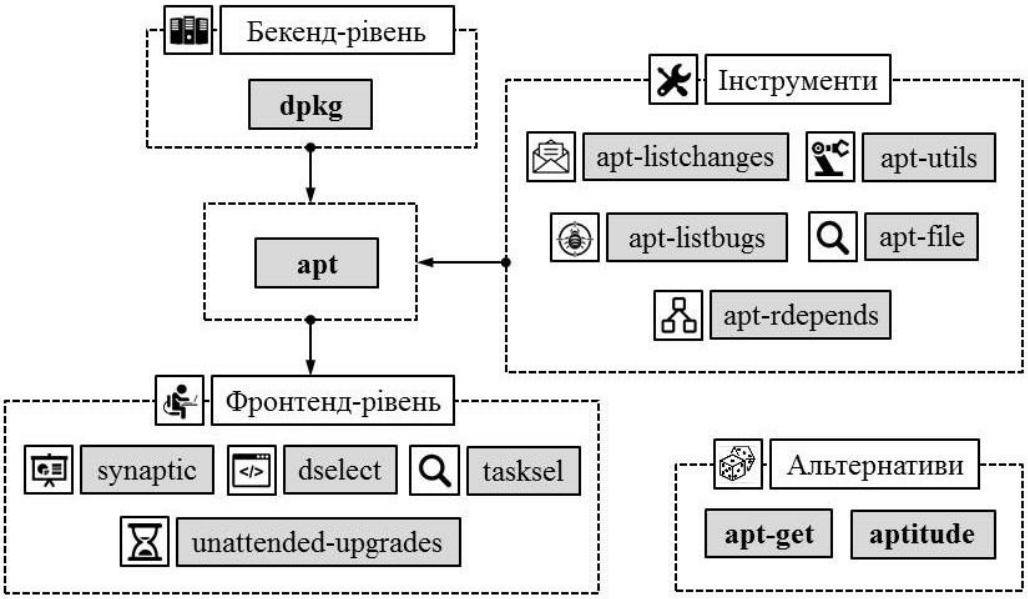

Рис. 1. Співвіднесення типових фронтенд-утиліт ОС Debian

Утиліта «apt» працює з набором «apt-utils», інструментом «apt-listchanges», що відслідковує історію пакунку, блоком «apt-listbugs», що повідомляє про критичні помилки пере інсталяцією пакунку і додатками «apt-file», що здійснює пошук пакунків та «apt-rdepends», що відслідковує зв'язки між пакунками. У свою чергу, фронтенд-утилітами по відношенню до «apt» є менеджер пакунків «synaptic», що надає графічний інтерфейс, «dselect» з тектовим інтерфейсом, інструмент пошуку пакунків для

інсталяції «tasksel» і додаток для автоматичного оновлення систем захисту ПЗ ОС Debian «unattended-upgrades». При цьому серед найбільш типових серед альтернативних по відношенню до «apt» утиліт можна зазначити скриптовий менеджер «apt-get» та інтерактивний менеджер «aptitude» [8; 11-12].

Серед ключових моментів конфігурації пакунків у системі Debian можна зазначити наступні чотири пункти [8]:

1) відсутність вбудованої системи конфігурації, застосування дистрибутиву «Debian GNU/Linux» зумовлює необхідність ручного налаштування конфігурації пакунків системним адміністратором;
2) наявність у кожного пакунку індивідуального сценарію налаштування (configuration script) зі стандартизованим інтерфейсом користувача debconf [13], що застосовується на початковому етапі встановлення пакунку і при оновленні програмного додатку;
3) доступ до повного набору системного пакету ПЗ (при цьому елементи, що характеризуються певними ризиками, можуть бути відключені за замовчанням, при їх включенні користувач несе індивідуальну відповідальність;
4) можливість включити у конфігурацію пакунку приховані блоки програмного коду (не використовується у популярних програмних додатках);

Даний перелік вказує на широкі можливості для роботи з пакунками у рамках системи Debian, що дозволяє як забезпечити високий рівень безпеки, так і зашкодити ОС у разі низького рівня обізнаності кінцевого користувача та ретельного тестування з конкретною конфігурацією Debian в безпечних умовах. Практика робота з даним дистрибутивом вказала на типовий перелік дій, яких треба уникати при роботі з пакунками [8-10]:

- включення тестових або нестабільних блоків у «/etc/apt/sources.list»;

- об'єднання стандартних Debian-архівів з архівами інших ОС (як-от архівами ОС Ubuntu) у «/etc/apt/sources.list»;
- налаштування поведінки інструментів керування пакунками через файли конфігурації без повного усвідомлення впливу застосування відповідних дій;
- встановлення невідомих пакунків через «dpkg -i <random_package>« або «dpkg --force-all -i <random_package>«;
- видалення або внесення змін у «/ var / lib / dpkg /»;
- перезапис системних файлів при встановленні ПЗ з невідомого джерела.

Наведений перелік застережень часто порушується користувачами, що запускають тестові та нестабільні пакунки, що характеризуються принципово новим функціоналом. Це пов'язано з тим, що пакети віднесені службою Debian до нестабільних можуть ефективно працювати протягом достатньо тривалого часу виконуючі основні функції, а наявні помилки виправляються оновленням [8-9; 14]. Застосування нестабільних пакетів популяризує парадигму «функціонування у режимі постійних оновлень» (Life with eternal upgrades), в цих умовах необхідно мати надійні системи захисту і відслідковування помилок і пакети аварійної інсталяції системи (rescue boot).

Система Debian також включає у себе декларації рівня залежності між пакунками та їх версіями, що визначають характер взаємодії ПЗ (рис. 2).

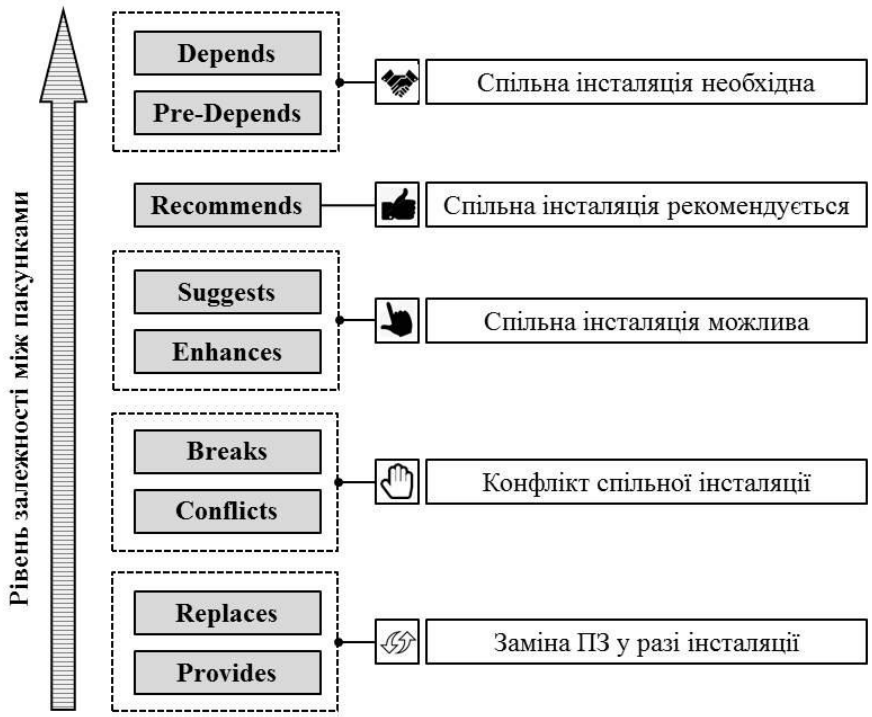

Рис. 2. Рівні залежності між пакунками ОС Debian

Таким чином, повний перелік рівнів залежності (dependency declaration) включає у себе наступні категорії [8; 14]:

- рівень «Depends» визначає повну залежність, пакунки зазначені у списку мають бути встановлені одночасно з основним пакунком або заздалегідь;
- рівень «Pre-Depends» визначає повну залежність, пакунки зазначені у списку мають бути встановлені перед основним пакунком;
- рівень «Recommends» вказує на значну залежність, але не вказує на принципову необхідність інсталяції зазначених пакунків;
- рівні «Suggests» та «Enhances» вказують на слабку залежність, якщо перелічені пакунки не будуть встановлені, ПЗ може, тим не менш, працювати ефективно;
- рівень «Breaks» вказує на несумісність з певними версіями пакунків;
- рівень «Conflicts»: вказує на повну несумісність, пакунки вказані у переліку мають бути попередньо деінстальовані;

- рівень «Replaces» вказує на те, що зазначені пакунки будуть заміщені після інсталяції нового ПЗ;
- рівень «Provides» вказує на те, що пакунок функціонально замістить зазначені пакунки і надалі виконуватиме відповідні функції.

Усвідомлення рівня залежності між пакунками є ключовим етапом забезпечення ефективної роботи пакувальника.

Моделювання процесу роботи пакувальника Debian. Життєвий цикл пакувальника складається з наступних етапів (рис. 3):

1) підготовка до оновлення пакунку (update);
2) оновлення пакунку (upgrade);
3) інсталяція пакунку (install);
4) видалення (remove) або остаточного видалення пакунку (purge).

Підготовка до оновлення пакунку виконується за допомогою команд «apt update», «aptitude update» або «apt-get update» і включає у себе отримання архівних метаданих та оновлення локальних метаданих, що використовуються утилітою «apt». У свою чергу етап оновлення пакунку виконується за допомогою команд «apt upgrade» і «apt full-upgrade», «aptitude safe-upgrade» і «aptitude full-upgrade», або «apt-get upgrade» і «apt-get dist-upgrade». Він включає у себе вибір версії для оновлення (зазвичай останньої серед перевірених), визначення залежностей пакунку, отримання пакунку, розпакування, запуск сценарію preinst, встановлення файлів прямого доступу (двійкових файлів) та запуск сценарію postinst. Інсталяція пакунку виконується за допомогою команд «apt install», «aptitude install» або «apt-get install». При цьому обираються пакунки, вказані у командному рядку, відбувається аналіз залежностей пакунку, а також отримання і розпакування двійкових пакетів з архіву, запускається сценарій preinst, встановлюються двійкові файли і запускається сценарій postinst.

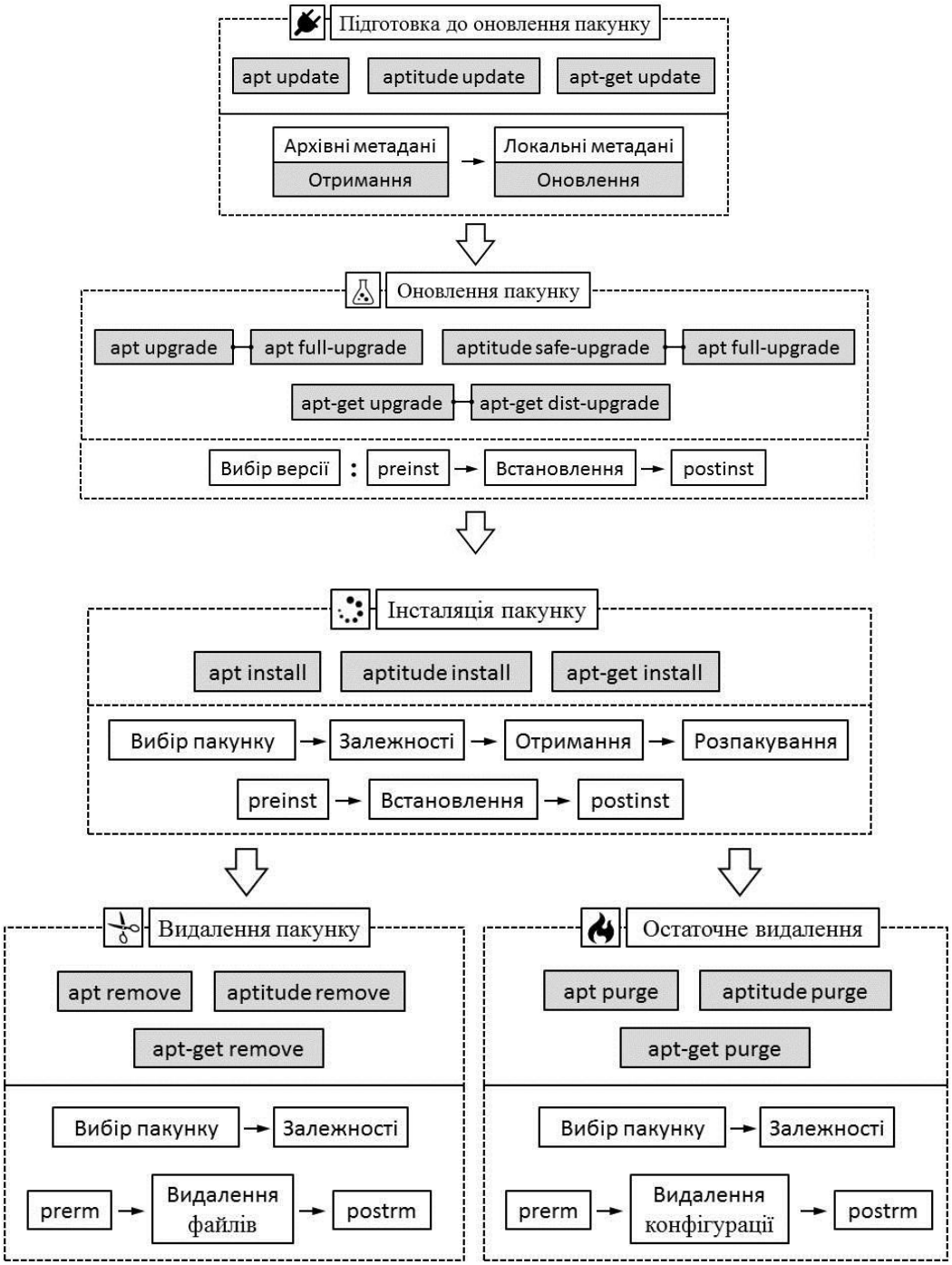

Рис. 3. Загальна схема функціонування пакувальника Debian

Підготовка до видалення пакунку (команди «apt remove», «aptitude remove» або «apt-get remove») включає у себе вибір пакунку, вказаного у командному рядку, аналіз залежностей пакунку, запуск сценарію prerm, видалення встановлених файли (крім файлів конфігурації) та запуск сценарію postrm. При остаточному видаленні (команди «apt purge»,

«aptitude purge» або «apt-get purge») відрізняються тим що при цьому видаляються і конфігураційні файли.

Розглянута модель визначає загальні уявлення етапи життєвого циклу пакувальника системи Debian. Для налагодження ефективної роботи пакувальника та визначення методології ефективного захисту системи при роботі з неперевіреними пакунками необхідно розглядати додаткові спеціалізовані команди, що не мають аналогів у «apt…», «aptitude…» та «apt-get…» [8; 14]. У табл. 1 показано їх скорочений перелік (у переліку не наведено очевидні аналоги команд «10», «11», «12» і «14»).

Таблиця 1

Перелік спеціалізованих команд, що використовуються пакувальником

	Команда	**Опис**
1	COLUMNS=120 dpkg -l <package_name_pattern>	Навести статус пакунку для звіту по помилки
2	dpkg -L <package_name>	Навести складові встановленого пакунку
3	dpkg -S <file_name_pattern>	Навести перелік пакунків з відповідною назвою
4	apt-file search <file_name_pattern>	Навести перелік пакунків з архіву з відповідною назвою
5	apt-file list <package_name_pattern>	Навести перелік складових пакунків з архіву з відповідною назвою
6	dpkg-reconfigure <package_name>	Реконфігурувати пакунок
7	configure-debian	Реконфігурувати пакунки екранного меню
8	dpkg --audit	Перевірити частково встановлені пакунки
9	dpkg --configure -a	Конфігурувати частково встановлені пакунки
10	apt-cache madison <package_name>	Вказати доступну версію та архівнк інформацію для пакунку
11	apt-get build-dep <package_name>	Встановити вказані пакунки
12	apt-get source <package_name>	Завантажити вихідний пакунок з архіву
13	debuild binary	Побудувати пакунки з локального дерева джерел (source tree)

14	make-kpkg kernel_image	Побудувати пакунки з дерева джерел ядра
15	dpkg -i <package_name>_<version>-<debian_version>_<arch>.deb	Встановити у систему локальний пакунок
13	apt install /path/to/<package_filename>.deb	Встановити у систему локальний пакунок і встановити заложності у автоматичному режимі
14	dpkg --get-selections '*' >selection.txt	Зберегти інформацію про стан вибір стану пакунку рівня dpkg
15	dpkg --set-selections <selection.txt	Встановити інформацію про стан вибір стану пакунку рівня dpkg

Команди конфігурації та інсталяції системи потрібно запускати під рутом (з правами адміністратора), крім того слід зазначити, що якщо «aptitude» використовує «regex», то інші команди керування пакунками використовують шаблон, подібний до «shell glob». Набір відповідностей між командою, пакунком і запуском утиліти показано у табл. 2.

Таблиця 2

Набір відповідностей між командою, пакунком і запуском утиліти

Команда	Пакунок	Запуск
«apt-file»	apt-file	apt-file update
«configure-debian»	configure-debian	dpkg-reconfigure (як сервер)
«apt-get build-dep» "apt-get source"	apt-get	запис «deb-src» у «/etc/apt/sources.list»
«apt-cache showsrc»	apt-cache	запис "deb-src" у "/etc/apt/sources.list"
«dget» «debuild» «debi»	devscripts	Запуск пакунку

Таким чином, розробка методології ефективної та надійної роботи пакувальника системи «Debian» має включати у себе наступні етапи:

1) визначення базової архітектури через вибір універсальної утиліти та відповідних утиліт бекенд- і фронтенд рівня;

2) врахування стандартного набору правил безпеки роботи з пакувальником, що включає у себе визначення потенційних ризиків та зменшення їх рівня;
3) декларація рівня залежності між пакунками та їх версіями, що визначають характер взаємодії ПЗ;
4) побудова базової моделі життєвого циклу пакувальника;
5) визначення спеціалізованих команд, зазначення відповідностей між командою, пакунком і запуском утиліти.

Розроблена схема керування пакунками та утилітами ОС Debian надає можливість побудувати ефективний інструментарій безпечного встановлення, корегування і видаленням програмних додатків, а також контролю системних ресурсів і прогнозування ризиків, що виникають на програмно-апаратному рівні.

Висновки. У роботі розглянуто особливості організації базової архітектури системи керування пакунками та менеджерами пакунків ОС Debian від бекенд- до фронтенд-рівнів. Зазначено типові правила безпеки роботи з пакувальником, визначено причини у відповідності до яких користувачі та системні адміністратори не дотримуються даних правил, вказано на відповідні ризики та підходи, що дозволяють зменшити вплив потенційних загроз. Наведено методику включення у схему моніторингу аналізу рівнів залежності між пакунками та їх версіями, що визначають характер взаємодії програмних додатків. Запропонована базова модель життєвого циклу пакувальника та визначено спеціалізовані команди, що можуть бути не включені до даної моделі, а також зазначено відповідності між окремою командою, пакунком і запуском утиліти. В результаті проведеного аналізу розроблено схема керування пакунками та менеджерами пакувальників, що дозволяє визначити вимоги для безпечної роботи з пакунками.

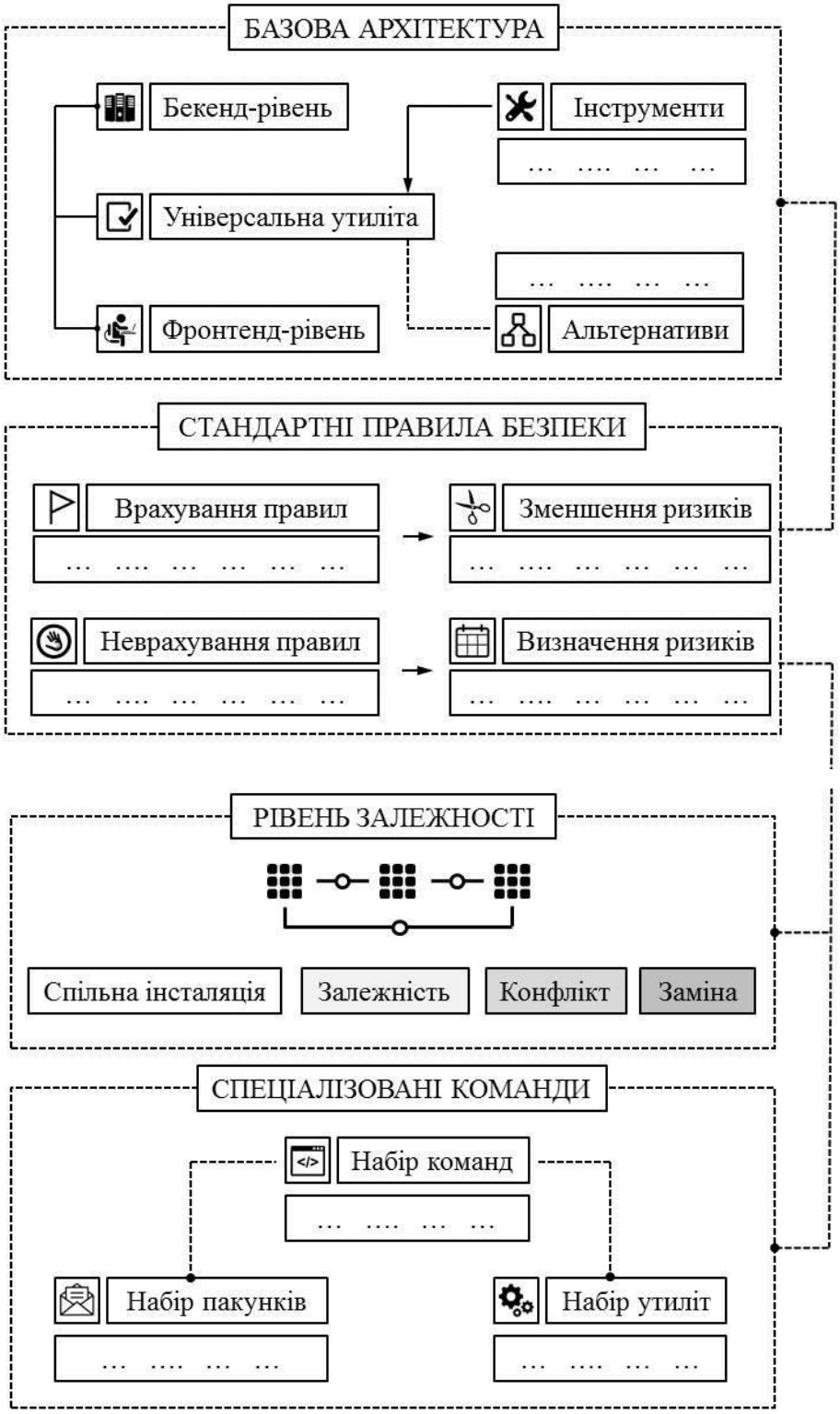

Рис. 4. Комплексна схема забезпечення надійної роботи пакувальника

Література

1. Coleman, E. Gabriella (2013). «Two Ethical Moments in Debian». Coding Freedom: The Ethics and Aesthetics of Hacking. Princeton University Press. ISBN 978-0-691-14461-0.
2. Hertzog, Raphaël (2013). The Debian Administrator's Handbook. Lulu. ISBN 979-10-91414-02-9. Retrieved 2014-06-22.
3. Coleman, Gabriella (Sep. 15, 2005). «Three Ethical Moments in Debian». NYU — Department of Media, Culture, and Communication. pp. 14–15.
4. Ludovic Courtès, Functional Package Management with Guix, June 2013, Madrid, European Lisp Symposium 2013.
5. Waters, John K. (8 September 2015). «JFrog Releases 'Universal' Artifact Repository». ADT Mag. Application Development Trends Magazine.
6. Claes, M., Mens, T., Cosmo, R. D., & Vouillon, J. (2015). A Historical Analysis of Debian Package Incompatibilities. 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. doi:10.1109/msr.2015.27.
7. Sousa, O. F. (2009). Analysis of the package dependency on Debian GNU/Linux. Journal of Computational Interdisciplinary Sciences, 1(1). doi:10.6062/jcis.2009.01.02.0015.
8. Hertzog, R., Mas, R., Zacchiroli, S., Nussbaum, L., & McGovern, N. (2016). Debian 8 Jessie: GNU-Linux. Paris: Eyrolles.
9. Hertzog, R., & Mas, R. (2015). The Debian administrators handbook: Debian Jessie from discovery to mastery. Sorbiers: Freexian Sarl.
10. Negus, C. (2013). Ubuntu Linux toolbox: 1000 commands for Ubuntu and Debian power users. Indianapolis, IN: Wiley.
11. Prakash, A., & PrakashI, A. (2018, December 10). Difference Between apt and apt-get Explained. Retrieved from https://itsfoss.com/apt-vs-apt-get-difference.

12. Linux man page: Aptitude (8). (n.d.). Retrieved from https://linux.die.net/man/8/aptitude.
13. Perform an unattended installation of a Debian package. (n.d.). Retrieved from microhowto.info/howto/perform_an_unattended_installation_of_a_debian_package.html.
14. This Debian Reference. Version 2.73: 2018-07-01 UTC. (n.d.). Retrieved from debian.org/doc/manuals/debian-reference/debian-reference.en.pdf.