УДК 004.056.53

ПОСЛУГА ПОВНОЇ ДОВІРЧОЇ КОНФІДЕНЦІЙНОСТІ ДЛЯ ЗАХИЩЕНОЇ ОС НА БАЗІ GNU/LINUX З РОЗШИРЕННЯМ RSBAC

Є.Ю. Іванніков

Київський національний університет ім. Тараса Шевченка,
03680, Київ, проспект Академіка Глушкова, 2,
тел.: 38 044-259-0139, e-mail: ivannikoff@meta.ua

Розглядається реалізація послуги безпеки «Повна довірча конфіденційність» для захищеноїопераційної системи (ОС) на базі Linux. Основою для реалізації виступає система розмежування доступу RSBAC. Визначені суттєві недоліки підсистеми довірчого керування у системі RSBAC та запропоноване її вдосконалення. Модифікація підсистеми гарантує дотримання усіх вимог на реалізацію послуги безпеки.

An implementation of a discretionary access control security service for a trusted OS based on GNU/Linux is considered. The implementation is based on RSBAC access control system. Some significant drawbacks of the discretionary acccess control subsystem in RSBAC are identified and its improvement is proposed. Modification of the subsystem guarantees that all requirements to the implementation of the security service are met.

Вступ

Під довірчим керуванням доступом згідно [1] будемо розуміти керування, при якому засоби захисту дозволяють звичайним користувачам управляти потоками інформації (у тому числі створювати нові потоки інформації) між іншими користувачами та об'єктами свого домену. Як правило, об'єкти, що належать домену користувача, це об'єкти, власником яких є користувач (створені користувачем). Математичною основою довірчого керування доступом є формальна дискреційна модель Харрісона–Руззо–Ульмана [2].

Довірче керування доступом розподіляється [3] на дві функціональні послуги (послуги безпеки): послуга довірчої конфіденційності (послуга «КД»), яка забезпечує захист від несанкціонованого ознайомлення з інформацією та послуга довірчої цілісності (послуга «ЦД»), яка забезпечує захист від несанкціонованої модифікації інформації.

Нормативний документ [3] вводить ранжування послуг довірчого керування на підставі повноти захисту й вибірковості керування. Всього виділяють чотири рівні реалізації кожної послуги, від найнижчого, «КД-1. Мінімальна довірча конфіденційність» та «ЦД-1. Мінімальна довірча цілісність» до найвищого, «КД-4. Абсолютна довірча конфіденційність» та «ЦД-4. Абсолютна довірча цілісність».

Доповідь є частиною науково-дослідної роботи з розробки захищеної ОС, що проводилась на факультеті кібернетики КНУ ім. Т. Шевченка. Функціональний профіль безпеки розроблюваної захищеної операційної системи (ЗОС) включає послуги довірчого керування доступом. ЗОС створюється на основі дистрибутиву Debian ОС Linux, комплекс засобів захисту (КЗЗ) базується на системі розмежування доступу RSBAC. Система RSBAC, яка є розширенням ядра ОС Linux, пропонує реалізацію довірчого керування доступом на основі використання списків контролю доступом. Доповідь є продовженням робіт автора [7, 8] з дослідження розширення RSBAC.

У роботі [8] на основі аналізу вихідних кодів системи RSBAC відновлено алгоритм, згідно якого відбувається перевірка допустимості запитів при довірчому керуванні. Стверджувалось, що за умови розробки процедур експорту та імпорту об'єктів, модуль ACL може бути використаний для впровадження послуг довірчого керування доступом, забезпечуючи функціонування послуг «ЦД-1» та «КД-3».

Слід визнати, що подальше дослідження та досвід експлуатації виявили, що алгоритм відрізняється від стандартного алгоритму перевірки прав доступу, введеному в стандарті POSIX 1003.1e [6], та його частковому випадку, який традиційно використовується в ОС типу Unix. Певні особливості алгоритму є неочікуваними для користувачів і можуть, у крайніх випадках, призвести до реалізації загроз конфіденційності.

У роботі пропонується вдосконалення системи RSBAC, покликане позбавити підсистему довірчого керування виявлених недоліків та гарантувати дотримання рівня «КД-3» надання послуги довірчої конфіденційності. Модифікований алгоритм перевірки запитів відповідає алгоритму перевірки прав доступу, описаному в стандарті [6] та повністю задовольняє вимогам, що накладаються на реалізацію послуги повної довірчої конфіденційності («КД-3»).

Вимоги до послуг довірчої конфіденційності

Мінімальна довірча конфіденційність («КД-1») дозволяє обмежити потоки інформації від об'єкта тільки до певних процесів. Хоч і не існує обмеження хто може активізувати процес, тобто отримувати інформацію, КЗЗ обмежує потоки інформації фіксованому списку процесів, ґрунтуючись на атрибутах доступу об'єктів і процесів.

© Є.Ю. Іванніков, 2010

Більш високі рівні послуги дозволяють обмежити потоки інформації від об'єкта до користувачів. У системі, яка реалізує послугу довірчої конфіденційності на рівні «КД-2», атрибути доступу об'єктів і користувачів містять інформацію, що використовується КЗЗ для розмежування доступу до об'єктів з боку конкретного користувача. Додатково існує можливість встановлювати, які користувачі можуть активізувати конкретний процес, що дозволяє отримати можливість обмеженого керування потоками інформації. Керування правами доступу на даному рівні має невисоку вибірковість. Користувач, домену якого належить об'єкт (процес), може вказати, які групи користувачів і, можливо, які конкретні користувачі мають право отримувати інформацію від об'єкта (ініціювати процес).

Повна довірча конфіденційність («КД-3») забезпечує більш високу вибірковість керування тим, які користувачі можуть отримати інформацію від об'єкта або ініціювати процес. Користувач, домену якого належить об'єкт, може вказати права доступу для кожного конкретного користувача і групи користувачів. Можливе включення або вилучення користувачів із списку доступу. Головна відмінність між рівнями «КД-2» та «КД-3» полягає у можливості визначити для останнього конкретну множину користувачів, які не мають права доступу до захищеного об'єкта.

Послуга абсолютної довірчої конфіденційності дозволяє визначити користувачів, процеси і пари процес / користувач, які можуть отримати інформацію від об'єкта, забезпечуючи повне керування потоками інформації в комп'ютерній системі (КС).

Вкажемо [2, 9] вимоги, що висуваються до реалізації послуги («КД-3. Повна довірча конфіденційність»).

1. Політика довірчої конфіденційності, що реалізується КЗЗ, має відноситись до всіх об'єктів КС.
2. КЗЗ має здійснювати розмежування доступу на підставі атрибутів доступу користувача і захищеного об'єкта.
3. Запити на зміну прав доступу до об'єкта мають оброблятися КЗЗ на підставі атрибутів доступу ініціатора запиту та об'єкта.
4. КЗЗ має надавати користувачу можливість для кожного захищеного об'єкта, що належить його домену, визначити конкретних користувачів (групи користувачів), які мають, а також тих, які не мають права одержувати інформацію від об'єкта.
5. КЗЗ має надавати користувачу можливість для кожного процесу, що належить його домену, визначити конкретних користувачів (групи користувачів), які мають, а також тих, що не мають права ініціювати процес.
6. Права доступу до кожного захищеного об'єкта мають встановлюватись у момент його створення або ініціалізації. Як частина політики довірчої конфіденційності мають бути представлені правила збереження атрибутів доступу об'єктів під час їх експорту та імпорту.

Система розмежування доступу RSBAC

Система RSBAC, запропонована в [5] – це надбудова над ядром ОС Linux і набір утиліт адміністрування, які дозволяють створити на базі дистрибутиву Linux захищену ОС. Певні системні виклики доповнюються кодом, що виконує звернення до центрального компонента RSBAC, який приймає рішення про допустимість чи недопустимість системного виклику.

Суб'єкти доступу є завжди процесами, що діють від імені користувачів. Об'єкти в RSBAC називаються цілями (доступу). Вони згруповані у типи цілей. Визначено такі типи цілей: файли (file), директорії (dir), символічні посилання (symlink), пристрої (dev), процеси як об'єкти доступу (process), міжпроцесна взаємодія (ipc), користувачі (user), групи користувачів (group), мережеві об'єкти (netobj) та ряд інших. Кількість операцій доступу (запитів), що контролюються, перевищує 50.

RSBAC розроблено на основі моделі розмежування доступу GFAC (Generalized Framework for Access Control), введеної в [4]. GFAC складається з окремих компонентів: 1) блока контролю системних викликів (AEF); 2) блока прийняття рішень (ADF); 3) сховища інформації про права доступу (ACI).

Перевірка допустимості системного виклику ініціюється блоком контролю системних викликів (AEF), який передає управління блоку прийняття рішень (ADF), разом з інформацією про тип доступу та про його учасників (суб'єкт та об'єкт запиту). Деякі з визначених типів доступу (запитів) наведено нижче.

1. CREATE – створити об'єкт;
2. READ_OPEN, READ_WRITE_OPEN – відкрити на читання, на запис та читання;
3. READ, WRITE – читати, записувати;
4. ADD_TO_KERNEL, REMOVE_FROM_KERNEL – завантажити, вивантажити модуль ядра;
5. EXECUTE – ініціювати процес.

Блок ADF за чергою опитує модулі політик безпеки. Модулі політик отримують інформацію про атрибути безпеки учасників доступу, які зберігаються у спеціальному сховищі інформації ACI. На основі відповідей модулів блок ADF вибудовує загальний результат, повертаючи його блоку AEF. У випадку допустимості запита, він буде виконаний, а блок AEF ініціюватиме процедуру корегування атрибутів учасників доступу. В іншому випадку, виконання системного виклику буде припинено із кодом помилки.

Компонентна архітектура GFAC дозволяє зробити захист незалежним від типу ОС та забезпечити одночасну роботу кількох різних модулів політик безпеки.

Реалізація архітектури GFAC у системі RSBAC показана на рис. 1.

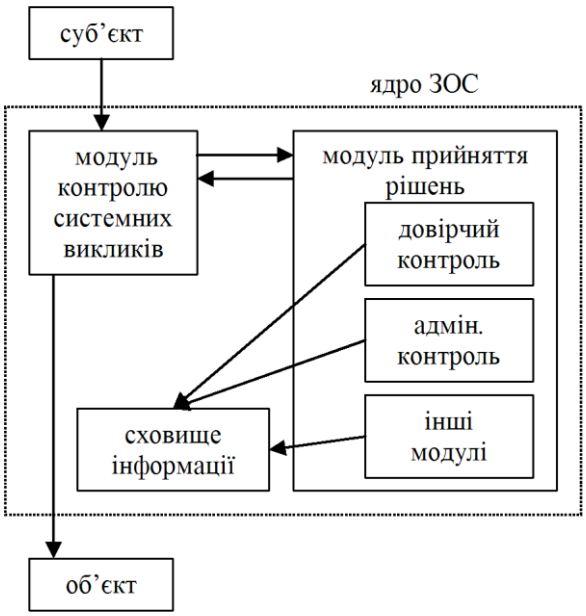

Рис. 1. Архітектура RSBAC

Серед інших модулів, RSBAC містить модуль ACL, який реалізує довірче керування доступом. Реалізація заснована на використанні списків контролю доступу (СКД).

Проведений у роботі автора [8] аналіз засвідчив принципову відповідність реалізації списків контролю доступу вимогам, що накладаються в [2] на реалізацію послуги повної довірчої конфіденційності. Разом з тим, порядок перевірки списків контролю доступу відрізняється від порядку, що традиційно використовується в ОС типу Unix. Дослідна експлуатація виявила, що найбільш неочікуваною особливістю алгоритма є адитивність прав користувача та груп користувачів. Ця особливість, в окремих випадках, не дозволяє власнику об'єкта обмежити доступ конкретного користувача до об'єкта. Далі наводиться алгоритм перевірки запитів.

Алгоритм перевірки запитів модулем ACL та його вдосконалення

Нагадаємо, що замість терміну «запит» модуль ACL оперує поняттям «право». Таким чином, списки контролю доступу суб'єкта *S* відповідають на запитання: «Чи має *S* право на певну дію щодо об'єкта *O*?». Алгоритм перевірки запитів приймає такі параметри: *owner* – користувач-власник процесу, що здійснює запит; *O* – об'єкт-ціль запиту; *right* – право, наявність якого слід перевірити.

Алгоритм повертає одне з двох значень: GRANTED – дозволити доступ власника owner до об'єкта *O* або NOT_GRANTED – заборонити доступ.

1. **для всіх** суб'єктів *S*, що є елементами множини {*owner*, група Everyone, роль процесу} **робити**
2. отримати *Rights* – множину прав суб'єкта *S* на об'єкт *O*.
3. **якщо** *Rights* невизначена **то**
4. **якщо** *O* знаходиться нагорі ієрархії успадкування **то**
5. *Rights* ← множина прав суб'єкта *S* за замовчуванням для об'єктів відповідного типу.
6. **інакше**
7. отримати *Mask* – маску успадкування для об'єкта *O*.
8. **якщо** *right* не міститься в *Mask* **то перейти** на рядок 1.
9. *O* ← батько для об'єкта *O*.
10. **перейти** на рядок 2.
11. **кінець якщо**
12. **кінець якщо**
13. **якщо** *right* належить множині *Rights* **то повернути** GRANTED.
14. **кінець для**
15. **для всіх** ACL-груп *G* таких, що *owner* належить *G* **робити**
16. отримати *Rights* – множину прав групи *G* на об'єкт *O*.
17. **якщо** *Rights* невизначена **то**
18. **якщо** *O* знаходиться нагорі ієрархії успадкування **то**
19. Rights ← множина прав суб'єкта *S* за замовчуванням для об'єктів відповідного типу.
20. **інакше**
21. отримати *Mask* – маску успадкування для об'єкта *O*.
22. **якщо** *right* не міститься в *Mask* **то перейти** на рядок 15.
23. *O* ← батько для об'єкта *O*.

24. **перейти** на рядок 16.
25. **кінець якщо**
26. **кінець якщо**
27. **якщо** *right* належить множині *Rights* **то повернути** GRANTED.
28. **кінець для**
29. **повернути** NOT_GRANTED.

Бачимо, що до прав користувача на об'єкт *O* по черзі додаються права групи Everyone, права ролі користувача, права всіх груп, членом яких є даний користувач. На наш погляд, подібний підхід до реалізації довірчого керування породжує деякі суттєві проблеми. По-перше, адитивність прав користувача, груп та всіх користувачів суперечить традиційному довірчому розмежуванню доступу в ОС типу Unix, заснованому на використанні тріад прав власника, групи власника та решти користувачів. Це може виявитись несподіваним для користувачів, призвести до некоректного призначення прав доступу і, як наслідок, до компрометації системи.

По-друге, використовуючи такий принцип перевірки прав доступу, виявляється складно (навіть неможливо) обмежити права доступу конкретного користувача. У цьому неважко пересвідчитись на такому прикладі. Нехай списки контролю доступу до об'єкта *O* включають права певної ACL групи *G*, яка в момент часу *t* не містить користувача *U*. Нехай права доступу налаштовані так, що в момент часу *t* користувач *U* не має доступу до об'єкта *O*. При цьому, власник об'єкта *O* не є власником групи *G* (отже, не може визначати членство в групі *G*). У такому випадку, не існує можливості обмежити права доступу користувача *U* на об'єкт *O*, оскільки в момент часу $t+\tau$ власник групи *G* може додати користувача *U* до групи *G*. Ясно, що користувач *U* отримає право доступу до об'єкта *O* без відома самого власника об'єкта.

Таким чином, реалізація модуля довірчого керування доступом у системі RSBAC не може гарантувати дотримання, в деяких випадках, вимог (4) та (5), які накладаються нормативним документом [2].

Вважаємо за необхідне модифікувати алгоритм перевірки прав доступу так, щоб існував однозначний та простий спосіб обмежити права доступу конкретного користувача. Будемо виходити з таких міркувань. Якщо списки контролю доступу до об'єкта *O* включають перелік прав деякого користувача *U*, то права цього користувача на об'єкт *O* мають визначатися цим і лише цим переліком, не додаючи права груп та права всіх користувачів. Лише якщо списки контролю доступу до об'єкта *O* не включають перелік прав деякого користувача *U* (враховуючи права, отримані через механізм успадкування), то потрібно здійснити перевірку прав груп, членом яких є користувач *U*. Останнім кроком стане перевірка прав всіх користувачів.

Таким чином, для того, щоб заборонити конкретний тип доступу для конкретного користувача або для конкретної групи користувачів, достатньо визначити для суб'єкта таку множину прав, яка не включає цей тип доступу.

На користь такої організації свідчить її добра узгодженість із алгоритмом перевірки прав доступу, описаним у стандарті POSIX 1003.1e [6, с. 43]. Дійсно, до реалізації списків контролю доступу стандарт висуває вимогу збереження такого порядку перевірки:

1) права власника об'єкта;
2) права конкретних користувачів;
3) права групи власника;
4) права конкретних груп;
5) права решти користувачів.

Далі пропонується модифікація алгоритму перевірки запитів модулем ACL. Модифікація полягає у зміні порядку перевірки списків контролю доступом та у використанні принципу, згідно якого перший знайдений запис визначає допустимість чи недопустимість запиту. Останній принцип запобігає об'єднанню множини прав користувача з множиною прав усіх груп, членом яких він є, та з множиною прав, спільною для всіх користувачів (група Everyone).

1. **для всіх** суб'єктів *S*, що є елементами множини {*owner*, роль процесу} **робити**
2. отримати *Rights* – множину прав суб'єкта *S* на об'єкт *O*.
3. **якщо** *Rights* невизначена **то**
4. **якщо** *O* знаходиться нагорі ієрархії успадкування **то**
5. *Rights* ← множина прав суб'єкта *S* за замовчуванням для об'єктів відповідного типу.
6. **інакше**
7. отримати *Mask* – маску успадкування для об'єкта *O*.
8. **якщо** *right* не міститься в *Mask* **то перейти** на рядок 1.
9. *O* ← батько для об'єкта *O*.
10. **перейти** на рядок 2.
11. **кінець якщо**
12. **кінець якщо**
13. **якщо** *right* належить множині *Rights* **то повернути** GRANTED.
14. **інакше повернути** NOT_GRANTED.
15. **кінець для**
16. **для всіх** ACL-груп *G* таких, що *owner* належить *G* (включаючи останнім кроком групу Everyone) **робити**
17. отримати *Rights* – множину прав групи *G* на об'єкт *O*.

18. **якщо** *Rights* невизначена, **то**
19. **якщо** *O* знаходиться нагорі ієрархії успадкування **то**
20. Rights ← множина прав суб'єкта *S* за замовчуванням для об'єктів відповідного типу.
21. **інакше**
22. отримати *Mask* – маску успадкування для об'єкта *O*.
23. **якщо** *right* не міститься в *Mask* **то перейти** на рядок 16.
24. *O* ← батько для об'єкта *O*.
25. **перейти** на рядок 17.
26. **кінець якщо**
27. **кінець якщо**
28. **якщо** *right* належить множині *Rights* **то повернути** GRANTED.
29. **інакше повернути** NOT_GRANTED.
30. **кінець для**
31. **повернути** NOT_GRANTED.

Модифікація вихідних кодів системи розмежування доступу RSBAC

У цьому підрозділі будуть запропоновані зміни, необхідні для реалізації модифікованого алгоритму перевірки запитів.

Після перехоплення системного виклику блоком AEF, управління передається блоку ADF, внутрішня функція rsbac_adf_request_int() якого ініціює запуск функцій перевірки запиту кожним модулем політик безпеки. Функція rsbac_adf_request_int() викликає головну функцію модуля ACL rsbac_adf_request_acl(). Та, виконавши загальну перевірку запиту, передає управління функції rsbac_acl_check_right().

Функція rsbac_acl_check_right() для кожного суб'єкта *S* (власник процесу, група Everyone, роль власника, групи, членом яких є власник процесу) викликає функцію rsbac_acl_get_single_right(), передаючи їй такі параметри:

1) *target* – ідентифікатор типу цілі – об'єкта запиту (file, dir, symlink, dev, process і т.д.);
2) *tid* – ідентифікатор цілі – об'єкта запиту (конкретний ідентифікатор об'єкта), пара (*target*, *tid*) однозначно визначає об'єкт запита *O;*
3) *subj_type* – ідентифікатор типу суб'єкта (користувач, роль, група);
4) *subj_id* – ідентифікатор суб'єкта (конкретний ідентифікатор користувача, ролі або групи), пара (*subj_type*, *subj_id*) однозначно визначає суб'єкт *S*, для якого здійснюється перевірка;
5) *right* – право, наявність якого потрібно перевірити;
6) **result* – вказівник на змінну, яка зберігатиме результат перевірки.

Функція rsbac_acl_get_single_right() здійснює пошук суб'єкта *S* у списках контролю доступу, ініціюючи, можливо, механізм успадкування списків. У випадку успішного відпрацювання функція повертає значення 0 та встановлює у значення змінної **result* результат перевірки. Це можуть бути значення TRUE – якщо списки контролю доступу (множина *Rights* в позначеннях алгоритму) об'єкта *O* містять запис для суб'єкта *S*, або FALSE – якщо списки контролю доступу об'єкта *O* не містять запис для суб'єкта *S*, або містять такий запис, який не включає шукане право *right*.

Дві функції rsbac_acl_check_right() та rsbac_acl_get_single_right() забезпечують реалізацію алгоритму перевірки запитів модулем ACL. Почнімо з модифікації функції rsbac_acl_get_single_right(). Зміна вихідного коду цієї функції потрібна для розрізнення трьох випадків та встановлення потрібного значення у змінну **result*:

1) множина *Rights* невизначена;
2) множина *Rights* визначена, і $right \in Rights$;
3) множина *Rights* визначена, але $right \notin Rights$.

Тип змінної **result* буде змінено з булівського на спеціальний перелічний тип rsbac_adf_req_ret_t, який може приймати значення із множини {NOT_GRANTED, GRANTED, DO_NOT_CARE, UNDEFINED}. У першому випадку функція rsbac_acl_get_single_right() повертатиме значення UNDEFINED, у другому – GRANTED, у третьому – NOT_GRANTED.

Функція rsbac_acl_check_right() буде змінена так, щоб порядок пошуку суб'єктів *S* відповідав стандартному порядку. Крім того, буде змінена логіка перевірки значення змінної **result*. Робота функції буде відповідати наведеній схемі:

1) викликати функцію rsbac_acl_get_single_right() для ініціатора запиту (власника процесу);
2) якщо **result* дорівнює GRANTED або NOT_GRANTED, то повернути **result*;
3) викликати функцію rsbac_acl_get_single_right() для ролі процесу;
4) якщо **result* дорівнює GRANTED або NOT_GRANTED, то повернути **result*;
5) для всіх груп, членом яких є ініціатор запита (включаючи, на останньому кроці, групу Everyone)
   a. викликати функцію rsbac_acl_get_single_right() для групи;
   b. якщо **result* дорівнює GRANTED або NOT_GRANTED, то повернути **result*;
6) повернути NOT_GRANTED.

Відповідність схеми роботи модифікованих функцій і запропонованого алгоритма очевидна.

Висновки

Модуль ACL системи розмежування доступу RSBAC реалізує довірче керування доступом, використовуючи списки контролю доступом. Модуль дозволяє користувачу задавати права доступу на об'єкти свого домена, встановлюючи допустимість чи недопустимість кожного з 51 визначеного в системі RSBAC запиту. Модуль підтримує механізм успадкування атрибутів та дає можливість задавати права доступу за замовчуванням.

Для використання модуля у складі КЗЗ розроблюваної ЗОС був проведений аналіз вихідних кодів. У ході аналізу було встановлено, що модуль в цілому задовольняє вимогам, які висуває [2] до реалізації послуги безпеки мінімальної довірчої цілісності («ЦД-1») та повної довірчої конфіденційності («КД-3»). Водночас, виявлена особливість алгоритма перевірки допустимості запиту (адитивність прав користувача та груп користувачів) є неочікуваною для користувачів; при певних налаштуваннях списків контролю доступу не виключена ситуація, за якої власник об'єкта не зможе обмежити доступ конкретних користувачів (груп) до об'єкта свого домена. Тим самим, експлуатація модуля ACL не може гарантувати дотримання вимог 4 та 5 на реалізацію послуги «КД-3». Нагадаємо їх:

4. КЗЗ повинен надавати користувачу можливість для кожного захищеного об'єкта, що належить його домену, визначити конкретних користувачів (і групи користувачів), які мають, а також тих, які не мають права отримувати інформацію від об'єкта.

5. КЗЗ має надавати користувачу можливість для кожного процесу, що належить його домену, визначити конкретних користувачів (і групи користувачів), які мають, а також тих, що не мають права ініціювати процес.

При підготовці роботи була здійснена модифікація алгоритму перевірки запитів. По-перше, порядок пошуку суб'єктів у списках контролю доступу був змінений на такий, який відповідає стандартному [6]. По-друге, проведені зміни дозволяють обмежити право конкретних користувачів (і групи користувачів) отримувати інформацію від об'єкта, шляхом визначення для них таких списків доступу, які не містять прав на виконання відповідних запитів. По-третє, модифікований алгоритм дозволяє обмежити право конкретних користувачів (групи користувачів) ініціювати процес, шляхом визначення для них списків доступу, які не містять права ініціації процесу (право на виконання запиту EXECUTE).

Таким чином, модифікований алгоритм перевірки запитів забезпечує виконання вимог 4 та 5 та відповідає вимогам на реалізацію послуги «КД-3». За умови розробки процедур експорту та імпорту об'єктів, модифікований модуль ACL системи RSBAC може бути використаний для забезпечення функціонування довірчого керування доступом рівнів «ЦД-1» та «КД-3».

1. *НД* ТЗІ 1.1-002-99. Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу. – К.: ДСТСЗІ СБ України, 1999. – 21 с.
2. *Зегжда Д.П., Ивашко А.М.* Основы безопасности информационных систем. – М.: Горячая линия – Телеком, 2000. – 452 с.
3. *НД* ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. – К.: ДСТСЗІ СБ України, 1999. – 59 с.
4. *LaPadula L.J.* Essay 9: Rule-Set Modeling of a Trusted Computer System // Information Security: An Integrated Collection of Essays. – Los Alamitos, California, USA: IEEE Computer Society Press.
5. *Ott A.* Regelsatz-basierte Zugriffskontrolle nach dem „Generalized Framework for Access Control" Ansatz am Beispiel Linux. – Diplomarbeit, Fachbereich Informatik, Universität Hamburg, 10. November 1997. http://www.rsbac.org/doc/media/dipl-ps.zip.
6. *IEEE* 1003.1e and 1003.2c: Draft Standard for Information Technology–Portable Operating System Interface (POSIX)–Part 1: System Application Program Interface (API) and Part 2: Shell and Utilities, draft 17 (withdrawn). October 1997. http://wt.xpilot.org/publications/posix.1e/
7. *Іванніков Є.Ю.* Адміністративне керування в ОС Linux на основі системи RSBAC // Матеріали всеукраїнської науково-технічної конференції студентів, аспірантів, молодих вчених з міжнародною участю «Інформаційно-керуючі системи і комплекси» – ІКСК'2008, Миколаїв: НУК, 2008. – С. 16–21.
8. *Іванніков Є.Ю.* Довірче керування в ОС GNU/Linux на основі системи розмежування доступу RSBAC // Матеріали 4-ї Міжнародної науково-технічної конференції «Комп'ютерні науки та інформаційні технології» – CSIT'2009. – Львів: Видавництво ПП «Вежа і Ко», 2009. – С. 286–290.