УДК 004.056.53

А.В. Анісімов, Є.Ю. Іванніков

ПОСЛУГА «КО-1. ПОВТОРНЕ ВИКОРИСТАННЯ ОБ'ЄКТІВ» ДЛЯ ЗАХИЩЕНОЇ ОС НА БАЗІ GNU/LINUX З РОЗШИРЕННЯМ RSBAC

Розглядається питання реалізації послуги безпеки «КО-1. Повторне використання об'єктів» для захищеної ОС, яка базується на ОС GNU/Linux з розширенням безпеки RSBAC. Встановлено, що стандартні засоби базової ОС та розширення RSBAC не повністю відповідають вимогам, які накладаються на послугу безпеки. Визначено вразливості, що можуть призвести до реалізації загроз конфіденційності та цілісності. Запропоновані додаткові заходи захисту для протидії загрозам.

Вступ

До захищених операційних систем (ЗОС) висувається ряд вимог, дотримання яких гарантує конфіденційність та цілісність оброблюваної інформації, а також доступність та спостережність комп'ютерної системи (КС) в цілому. Ці вимоги розглядаються як набір функціональних послуг (послуги безпеки) [1].

Однією із таких послуг є послуга «КО-1. Повторне використання об'єктів». Її реалізація дозволяє забезпечити коректність повторного використання розділювальних об'єктів, гарантуючи, що в разі, якщо розділювальний об'єкт виділяється новому користувачу або процесу, то він не містить інформації, яка залишилась від попереднього користувача або процесу [2].

Функціонально послуга «КО-1» належить до критеріїв захисту від несанкціонованого ознайомлення з інформацією (критерії конфіденційності) і розглядається як така, що є необхідною умовою у реалізації решти послуг із забезпечення конфіденційності інформації, що є цілком очевидним [2]. Більше того, втілення послуги «КО-1» вимагається і для надання високих рівнів послуг, які забезпечують цілісність інформації (послуги «ЦД-3», «ЦД-4», «ЦА-3» та «ЦА-4») [2].

Таким чином, послуга безпеки з повторного використання об'єктів являється важливою складовою частиною комплексу засобів захисту (КЗЗ). Її коректна реалізація дозволяє гарантувати, що зусилля із впровадженню високих рівнів інших послуг із захисту даних не виявляться марними.

Стаття є частиною науково-дослідної роботи із розробки захищеної ОС, що проводилась на факультеті кібернетики КНУ ім. Тараса Шевченка. Профіль безпеки розроблюваної ЗОС включає послугу «КО-1. Повторне використання об'єктів». Автори ставлять перед собою задачу впровадження названої послуги безпеки, вимагаючи, щоб її реалізація відповідала всім вимогам, які регламентує нормативний документ [2].

Проведений спеціальний аналіз ОС Linux та розширення RSBAC засвідчив, що сама по собі базова ОС не забезпечує повного виконання зазначених вимог. Далі буде показана можливість реалізації як загрози конфіденційності, так і загрози цілісності оброблюваної інформації. Протидія цим загрозам полягатиме в організації додаткових заходів адміністративного характеру, у використанні стороннього програмного забезпечення та у модифікації ряду системних додатків.

1. Огляд захищеного середовища

Розроблювана ЗОС базується на дистрибутиві Debian ОС GNU/Linux з розширенням RSBAC, яке представляє собою надбудову над ядром ОС Linux і набір утиліт адміністрування [3]. RSBAC функціонує на рівні ядра і додає власні перевірки до системних викликів. Система RSBAC розроблена на основі узагальненої моделі розмежування доступу GFAC

(Generalized Framework for Access Control), що була запропонована в роботі [4].

В архітектурі RSBAC засоби для примусового розмежування доступу, засоби для прийняття рішення про допустимість чи недопустимість доступу, а також дані про атрибути безпеки сторін – учасників доступу реалізовані як окремі компоненти. Структурно механізм захисту поділяється на три частини:

1) блок контролю системних викликів (Access Enforcement Facility – AEF),

2) блок прийняття рішень (Access Decision Facility – ADF),

3) сховище інформації про атрибути безпеки об'єктів і суб'єктів КС (Access Control Information – ACI).

Системні виклики доповнюються кодом блоку контролю AEF, який перетворює їх у один зі стандартних (визначених у системі RSBAC) запитів до блока прийняття рішень ADF. Останній здійснює опитування модулів політик безпеки та приймає рішення про допустимість чи недопустимість виклику (надання чи ненадання доступу) на основі відповідей модулів. Модулі політик безпеки отримують інформацію для прийняття рішень із атрибутів безпеки ініціатора запита і об'єкта доступу, які зберігаються у сховищі інформації ACI.

2. Вимоги до реалізації послуги «КО-1»

Згідно критеріїв [2] послуга «КО-1» мусить відповідати таким вимогам:

1) політика повторного використання об'єктів, що реалізується КЗЗ, має поширюватись на всі об'єкти КС;

2) перш ніж користувач або процес зможе отримати у своє розпорядження звільнений іншим користувачем або процесом об'єкт, встановлені для попереднього користувача або процесу права доступу до даного об'єкта мають бути скасовані;

3) перш ніж користувач або процес зможе отримати у своє розпорядження звільнений іншим користувачем або процесом об'єкт, вся інформація, що міститься в даному об'єкті, має стати недосяжною.

Ці вимоги потребують деяких коментарів. Друга вимога гарантує, що рішення про надання чи про ненадання доступу до об'єкта завжди будується на основі інформації про права доступу на цей об'єкт ініціатора запиту (суб'єкта).

У розроблюваній ЗОС інформація про права доступу на об'єкт зберігається окремо від самого об'єкта, тому постає питання про тривалість життя інформації про права доступу. А саме, КЗЗ має гарантувати, що за жодних обставин інформація про права доступу не містить дані про неіснуючі (видалені) об'єкти. Докладніше це буде розглянуто далі.

Якщо дотримання другої вимоги можна назвати захистом метаданих (про права доступу), то дотримання третьої забезпечує власне захист оброблюваної інформації. Тут ми розрізняємо ситуацію із видаленням об'єкта, що має супроводжуватись знищенням інформації, яка в ньому містилась, та ситуацію зі спільним використанням розділювальних об'єктів (пам'ять, файл або розділ підкачки, тимчасові файли і т. д.)

З огляду на перераховані вимоги, робота буде організована таким чином. Спочатку серед усіх об'єктів КС будуть визначені такі їхні типи, на які має поширюватись функціональність, яку надає послуга безпеки. Для об'єктів визначених типів буде проведено аналіз на предмет відповідності другій вимозі. Оскільки в розроблюваній ЗОС атрибути доступу зберігаються окремо від об'єктів, додатково буде досліджено ситуацію зі скасування атрибутів доступу об'єктів при їх видаленні. Нарешті, буде розглянуто третю вимогу про захист інформації, яка міститься в об'єктах, що видаляються або звільняються.

3. Об'єкти захисту

Розроблювана ЗОС базується на розширенні безпеки RSBAC. Захист від несанкціонованого доступу здійснюється, в тому числі, з урахуванням типу об'єкта. Система RSBAC розрізняє такі типи об'єктів:

1) об'єкти файлової системи (власне файли, каталоги, файли пристроїв); сюди відносимо символічні посилання та іменовані канали (FIFO-файли);

2) процеси як об'єкти системних викликів;

3) користувачі як об'єкти системних викликів;

4) об'єкти міжпроцесної взаємодії: семафори, черги повідомлень, спільна пам'ять, анонімні канали, Unix-сокети (іменовані та анонімні).

Об'єкти КС, на які має поширюватися захист послугою «КО-1», є розділювальними об'єктами, що можуть використовуватися повторно. З огляду на це, захисту має підлягати інформація, яка міститься у файлах та сторінках пам'яті (оперативної і віртуальної).

Уточнимо, що ми розуміємо під повторним використанням файлів. Справа полягає у тому, що при видаленні файла або при зменшенні його розміру файлова система лише позначає вивільнені блоки як «незайняті». Як такого знищення інформації, що зберігається в цих блоках, не відбувається. Слід наголосити на тому, що коли користувач видаляє файл, який містить конфіденційну інформацію, ця інформація виходить зі сфери контролю КЗЗ. Цілком зрозуміло, що в захищеній ОС видалення конфіденційної інформації має бути не лише логічною операцією файлової системи. Як правило, тут застосовується операція «безпечного видалення» файлів, яка полягає у перезаписі блоків, що вивільняються, нулями або випадковими даними.

Ті ж самі міркування стосуються й оперативної пам'яті. Вивільнення сторінки пам'яті процесом має супроводжуватись обнуленням її вмісту, так, щоб будь-який інший процес, звертаючись до вивільненої сторінки, не отримав інформацію, яка в ній містилась раніше.

Повноцінний захист пам'яті КС передбачає і захист віртуальної пам'яті – області підкачки (як правило, в середовищі Linux область підкачки являє собою не файл, а спеціально виділений окремий розділ диску). Справді, область підкачки містить сторінки пам'яті, тимчасово переміщені на диск. Не можна виключити, що вивантажені сторінки містять конфіденційну інформацію та/або результати її обробки. Вміст області підкачки жодним чином не контролюється КЗЗ, отже реалізація послуги «КО-1» повинна надати механізм, який гарантуватиме неможливість витоку інформації, яка зберігається в цій області.

Не можна не порушити питання про таке потенційне джерело компрометації інформації, як тимчасові файли. Вони створюються різними програмами під час своєї роботи для тимчасового зберігання інформації, і у деяких випадках (помилка у програмі або некоректна реалізація) можуть залишатись на диску, замість того, щоб бути видаленими.

Якщо для ОС загального призначення їхній вміст вважається «сміттям», то для захищеної ОС ці файли можуть містити конфіденційну інформацію, її фрагменти або ж результати обробки. Як правило, тимчасові файли всіх користувачів створюються у спеціальній системній директорії /tmp і мають випадкові імена. Через це, можливість впровадження адміністративного (або, ще гірше для даного випадку, довірчого) контролю доступу до них здається досить сумнівною. Ці файли підлягають автоматичному видаленню при завантаженні системи або при виході користувача із неї. Така реалізація хоч і не позбавлена певних недоліків, однак дозволяє гарантувати нетривалий час життя тимчасових файлів.

Також слід виділити ще одну область, на яку має розповсюджуватись послуга «КО-1». Мова йде про метадані – атрибути безпеки об'єктів, які видаляються. Цими об'єктами можуть бути користувачі, процеси, об'єкти файлової системи та об'єкти міжпроцесної взаємодії. Атрибути безпеки всіх сутностей КС (суб'єктів і об'єктів) зберігаються у внутрішньому сховищі інформації системи розмежування доступу RSBAC, окремо від самих сутностей. Для здійснення доступу до атрибутів безпеки система RSBAC використовує різні ідентифікатори, в залежності від типу сутності.

Так, користувачі ідентифікуються системою на підставі унікального числового ідентифікатора uid. Цей ідентифікатор пов'язується із кожним користувачем і є саме тим атрибутом, на основі якого ОС та КЗЗ у її складі здійснює довірче розмежування доступу.

Процеси ідентифікуються на підставі унікального числового ідентифікатора pid. Цей ідентифікатор визначає сутність – суб'єкт доступу при адміністративному розмежуванні доступу системою RSBAC.

Файлові об'єкти визначаються ідентифікатором пристрою та номером індексного дескриптора (inode). Відомо, що пара (ідентифікатор пристрою, номер індексного дескриптора) є унікальною у межах всієї віртуальної файлової системи (VFS), тому вона може використовуватися для взаємнооднозначного співставлення файловому об'єкту атрибутів безпеки. Цими атрибутами можуть бути списки контролю доступу, мітки адміністративного керування на основі мандатної моделі та дані інших модулів безпеки.

Для ідентифікації об'єктів міжпроцесної взаємодії використовується інформація про засіб взаємодії (семафор, черга, спільна пам'ять, анонімний канал, Unix-сокет) та унікальне числове значення.

Виникає запитання про те, що відбувається з атрибутами безпеки при а) видаленні облікового запису користувача; б) завершенні роботи процесу; в) видаленні об'єкта файлової системи; г) видаленні об'єкта міжпроцесної взаємодії.

При видаленні облікового запису користувача його числовий ідентифікатор вивільнюється і може бути використаний при додаванні нового користувача. При додаванні нового облікового запису, користувач разом із ім'ям отримує перший вільний ідентифікатор зі значенням більше 1000 (значення від 0 до 999, як правило, зарезервовані для ОС). Таким чином, новостворений користувач «отримає у спадок» входження у всі списки контролю доступу, що були встановлені для попереднього користувача з тим самим значенням uid.

Аналогічно, при завершенні роботи процесу його ідентифікатор pid вивільнюється. Не виключена ситуація, при якій процес, що запускається пізніше, може отримати таке значення ідентифікатора pid, яке використовувалось раніше. В такому випадку, цей процес отримає атрибути адміністративного допуску, встановлені для попереднього процесу з тим самим значенням pid.

При створенні нового об'єкта файлової системи або міжпроцесної взаємодії він може отримати таке ж значення ідентифікатора, яке використовувалось раніше видаленим об'єктом. Новий об'єкт отримає всі атрибути безпеки, що були встановлені для видаленого об'єкта.

Очевидно, що така ситуація є неприпустимою. Час життя атрибутів безпеки має не перевищувати час життя захищених об'єктів. З метою дослідження цієї проблеми було проведено аналіз вихідних кодів системи RSBAC. Результати аналізу пропонуються у наступному підрозділі.

Нами були названі об'єкти, на які має розповсюджуватись захист послугою «КО-1». Нагадаємо, що це є файли при їхньому видаленні або зменшенні розміру, сторінки оперативної пам'яті при завершенні виконання процесів, область підкачки, тимчасові файли, атрибути користувачів, процесів, об'єктів файлової системи і міжпроцесної взаємодії.

4. Захист атрибутів прав доступу

Задачу захисту прав доступу, що виникає при розгляді другої вимоги до реалізації послуги «КО-1» (вимога про скасування атрибутів), розділимо на дві частини. По-перше, проаналізуємо ситуацію з атрибутами доступу при запиті користувача або процесу на доступ до звільненого іншим користувачем або процесом об'єкта. По-друге, звернемося до питання про час життя атрибутів доступу відносно часу життя об'єктів.

Перша задача не викликає жодних труднощів. При перехопленні кожного системного виклику блок контролю системних викликів (AEF) завжди повідомляє

блоку прийняття рішень (ADF) інформацію про сторони, задіяні у запиті. Блок ADF вибудовує рішення про дозвіл або заборону запиту на підставі, серед іншого, таких відомостей про ініціатора запиту:

1) числовий ідентифікатор процесу (pid);

2) числовий ідентифікатор користувача – власника процесу (uid).

Виявляється, що цих відомостей цілком достатньо для задоволення другої вимоги. Справді, модулі контролю доступу системи RSBAC враховують ці дані при розмежуванні доступу. При адміністративному керуванні суб'єкт – ініціатор запиту є процесом, який ідентифікується за значенням pid. При довірчому керуванні суб'єкт – ініціатор запиту є користувачем, який ідентифікується за значенням uid.

Тому, коли користувач або процес намагається отримати доступ до звільненого іншим користувачем або процесом об'єкта, перевірка запиту здійснюється виходячи зі значень атрибутів ініціатора запиту. Права доступу попереднього користувача або процесу не враховуються.

Тепер перейдемо до другої задачі. Нагадаємо, що при додаванні облікового запису новостворений користувач може отримати значення uid, яке належало попередньому (видаленому) користувачу. Це призведе до того, що новий користувач отримає всі права довірчого доступу та значення адміністративного допуску від користувача, обліковий запис якого було видалено.

Така ситуація є вкрай небезпечною. Дійсно, з точки зору КЗЗ дії нового користувача будуть цілком санкціонованими, однак вони можуть становити реальну загрозу не лише конфіденційності інформації, але й цілісності.

Розширення безпеки RSBAC містить набір утиліт адміністрування. Серед інших, наявні дві утиліти: acl_rm_user та attr_rm_user. Перша утиліта видаляє всі входження вказаного користувача з усіх списків контролю доступу та з усіх груп довірчого керування, повністю скасовуючи права довірчого доступу заданого користувача.

Друга утиліта видаляє всі атрибути, пов'язані з вказаним користувачем, у тому числі, й атрибути адміністративного контролю (рівень допуску, множини категорій і т. д.)

Наявних програмних засобів цілком достатньо для вилучення прав і атрибутів користувача. Однак, оскільки ці засоби реалізовані як окремі утиліти, їхній запуск залежить цілковито від адміністратора безпеки (лише він має право на зміну атрибутів безпеки). Водночас, стандартні засоби роботи із обліковими записами користувачів доступні для запуску виключно адміністратору КС.

На наш погляд, керування обліковими записами (разом із вилученням атрибутів) має виконуватись однією особою – чи то адміністратором КС, чи то адміністратором безпеки, що визначається конкретною схемою розподілу обов'язків адміністраторів (послугою безпеки «НО-2» або «НО-3») [2]. Тому постає задача з модифікації стандартних засобів керування обліковими записами користувачів.

Модифікація засобів має забезпечити автоматичний запуск утиліт RSBAC для видалення атрибутів користувача із внутрішнього сховища інформації при вилученні облікового запису. Далі, стандартні дії, що виконуються при видаленні облікового запису, мають виконуватись від імені адміністратора КС; перед запуском утиліти RSBAC необхідна зміна ідентифікатора власника процесу на ідентифікатор адміністратора безпеки, що дозволить успішно виконати решту дій (вилучення атрибутів зі сховища інформації ACI). Для того, щоб уможливити зміну ідентифікатора uid необхідні налаштування модуля AUTH системи RSBAC, який, в загальному випадку, забороняє запити процесу на зміну значення uid.

Аналіз вихідних кодів RSBAC показує цілком коректне опрацювання ситуації, що виникає при завершенні процесу, при видаленні об'єктів файлової системи (звичайно, маємо на увазі видалення останнього жорсткого посилання на об'єкт) та об'єктів міжпроцесної взаємодії (див. рис. 1). Дійсно, завершення процесу є системним викликом, який конт-

ролюється КЗЗ, тому, під час його виконання функція do_exit() здійснює запуск функції прийняття рішень блоку ADF, яка, в свою чергу, викликає функцію rsbac_remove_target(). Остання видаляє всі атрибути допуску/доступу суб'єкта (об'єкта). Отже, новий процес не отримає атрибутів безпеки, які були встановлені для попереднього процесу з тим самим значенням pid.

Так само не становить проблеми і видалення файлового об'єкта, оскільки воно відбувається в результаті виконання системних викликів, контрольованих КЗЗ. Код КЗЗ доповнює функції віртуальної файлової системи vfs_unlink(), vfs_rmdir(), vfs_rename_dir(), vfs_rename_other(), так, що після видалення об'єкта файлової системи відбувається сповіщення блоку ADF та подальше скасування всіх атрибутів, які були встановлені для цього об'єкта.

Функції ядра для видалення семафорів semctl_down() та повідомлень msgctl_down() супроводжуються викликом блоку ADF із подальшим скасуванням атрибутів цих об'єктів міжпроцесної взаємодії.

Дещо інакше опрацьовується видалення об'єктів спільної пам'яті, анонімних каналів та Unix-сокетів. Функції, відповідно, shm_destroy(), pipe_release() та unix_release() безпосередньо викликають функцію rsbac_remove_target().

У кінцевому підсумку, задача захисту атрибутів прав доступу зводиться лише до задачі коректного вилучення облікових записів користувачів, що вимагає модифікації відповідних програмних засобів та налаштування політики безпеки.

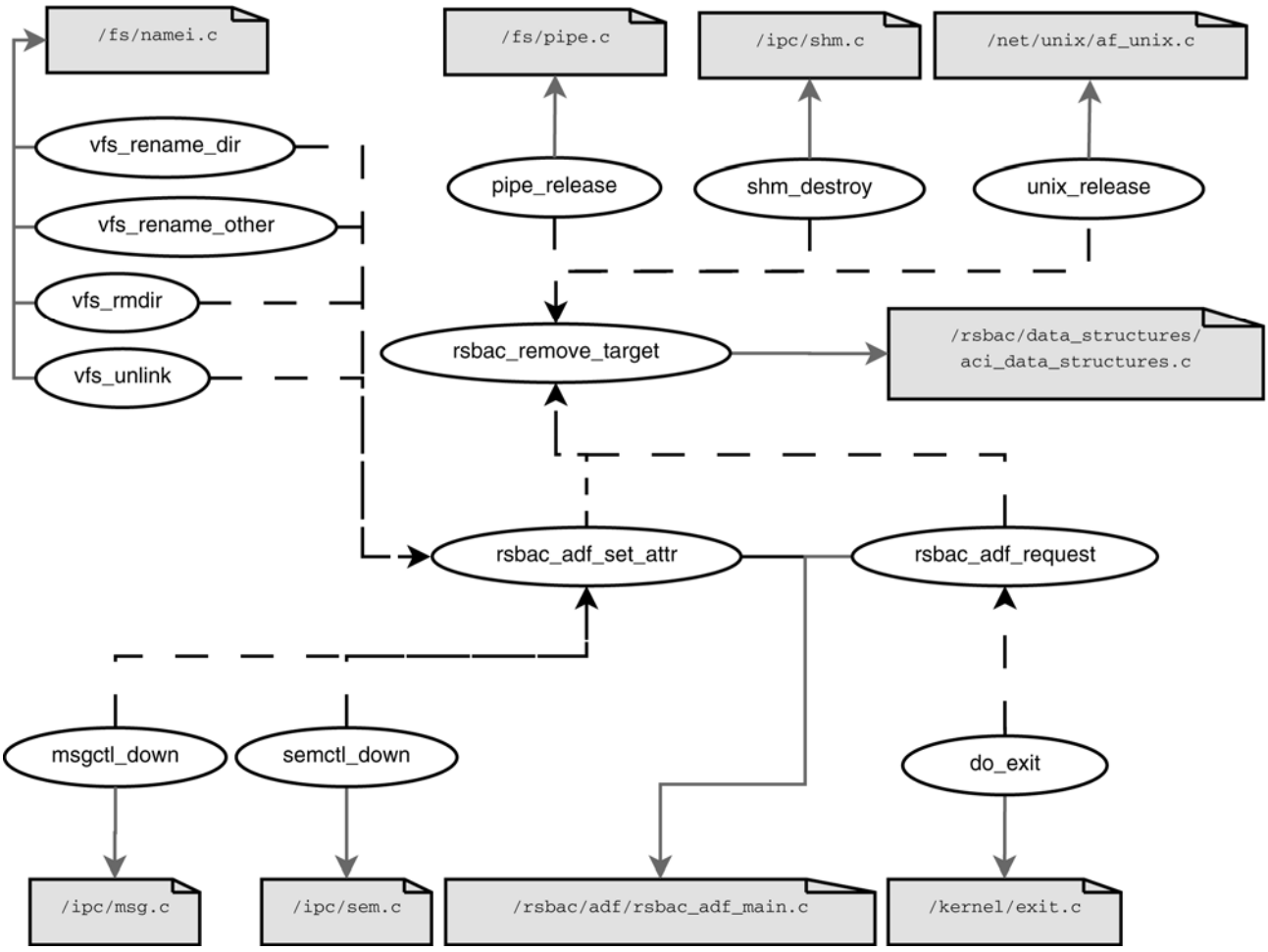

Рис. 1. Схема викликів функцій при завершенні роботи процесів та при видаленні об'єктів

5. Захист інформації, яка міститься в об'єктах

Розглянемо проблему захисту інформації, яка міститься у файлах, що видаляються. Як зазначалося, стандартний прийом тут полягає у перезаписі звільнених блоків нулями або випадковими даними. Хоча це можна зробити і за допомогою стандартних засобів ОС Linux або сторонніх утиліт, однак ми вважаємо, що в захищеній ОС потрібно якомога зменшити ймовірність помилки користувача. Якщо особа має у своєму розпорядженні засоби для видалення файлу як безпечним способом (із затиранням блоків) так і небезпечним, то ніхто не може гарантувати, що завжди буде реалізовуватись саме перший сценарій.

Окрім видалення, існує ще одна можливість виходу з-під сфери контролю КЗЗ інформації, що міститься у блоках файлів. Мова йде про зменшення розміру файла в результаті його модифікації. Вивільнені блоки позначаються як вільні і можуть стати в подальшому джерелом компрометації.

Очевидно, що задача контролю блоків, які вивільняються при зменшенні розміру файла (надалі – при врізанні файла) не може здійснюватись користувачем КС. Функціонування цієї частини послуги має бути цілком сферою відповідальності КЗЗ.

Слід також враховувати, що безпечне видалення файла є ресурсоємною операцією, яку недоцільно здійснювати для всіх без винятку файлів. Отже, політика безпеки, яка реалізується в КС, має визначати конкретні об'єкти, на які розповсюджується цей захист.

Тут ми бачимо дві можливості. Перша полягає у використанні модуля FF системи RSBAC. Модуль дозволяє для кожного файла зберігати набір атрибутів безпеки (прапорців), які є спільними для всіх суб'єктів доступу. Зокрема, передбачається прапорець, встановлення якого призведе до автоматичного затирання всіх блоків, зайнятих файлом при видаленні або врізанні. Адміністратор безпеки здатен визначити окремі файли або каталоги (оскільки прапорці можуть успадковуватись), які підлягатимуть безпечному видаленню та врізанню.

Друга можливість не виключає першу. Автори пропонують розширити функціональність, яку надає система RSBAC і використати можливості адміністративного контролю доступу для визначення файлів, які потребують такого типу захисту. Ідея проста: як відомо, з кожним файлом у моделі мандатного управління доступом пов'язується мітка безпеки. Якщо при видаленні файла або при зменшенні його розміру значення мітки безпеки перевищує деяке порогове значення, тоді блоки, що звільняються, необхідно перезаписати.

Модульна структура RSBAC дозволяє впровадити гнучку політику безпеки, в тому числі, і за рахунок розробки власних модулів безпеки. Контроль за безпечним видаленням може здійснюватись саме таким модулем.

Його реалізація забезпечить функціонування ефективної системи контролю об'єктів. Видалення (врізання) файлів, які підлягають адміністративному розмежуванню доступу, буде контролюватись автоматично. Крім того, в адміністратора безпеки завжди є можливість примусово гарантувати безпечне видалення (врізання) файла, встановивши відповідний прапорець модуля FF.

На черзі розгляд задачі захисту сторінок оперативної пам'яті. Виділення пам'яті виконується з використанням системних викликів mmap()/mmap2(). Вони резервують сторінки пам'яті для процесу. Посилання на кожну сторінку із таблиці сторінок віртуальної пам'яті після зазначеного системного виклику є посиланнями на нульову сторінку, яка доступна лише для читання всім процесам, та заповнена нулями, тому якщо процес намагається прочитати дані з пам'яті, то отримає нулі.

Ядро ОС очищує кожну сторінку пам'яті перед тим, як користувацький процес намагається вперше здійснити запис у неї. При цьому генерується виключна ситуація збою сторінки (page fault), яка обслуговується функцією ядра do_page_fault(). Послідовність ключових

викликів функцій у такому випадку показана на рис. 2.

Проведений аналіз дозволяє стверджувати, що операція виділення пам'яті для процесу ядром ОС GNU/Linux повністю відповідає вимогам, які висуваються до послуги повторного використання об'єктів.

Розглянемо захист інформації, яка міститься в області підкачки. Вдамося тут до криптографічних методів захисту. Існують вільні програмні засоби, які дозволяють здійснювати прозоре зашифрування і розшифрування даних віртуальної пам'яті. Для реалізації послуги було обрано пакет loop-AES [5]. loop-aes пропонує модифіковану версію стандартного модуля loop ядра ОС GNU/Linux. Цей модуль дозволяє створити псевдопристрій, пов'язаний із певним фізичним пристроєм і на стадії передачі даних від псевдопристрою до реального здійснює їхнє прозоре зашифрування. При передачі даних в оберненому напрямку відбувається їхнє розшифрування.

Отже, пов'язавши псевдопристрій, створений модулем, із реальним пристроєм, який відповідає за розділ підкачки та змонтувавши перший, отримаємо прозорий та ефективний криптографічний захист вмісту області підкачки. Після розмонтування псевдопристрою всі дані, фізично присутні на диску підкачки, будуть зашифрованими та недоступними для звичайного монтування.

Модуль loop-aes використовує алгоритм AES з довжиною ключа, яка обирається користувачем рівною 128, 192

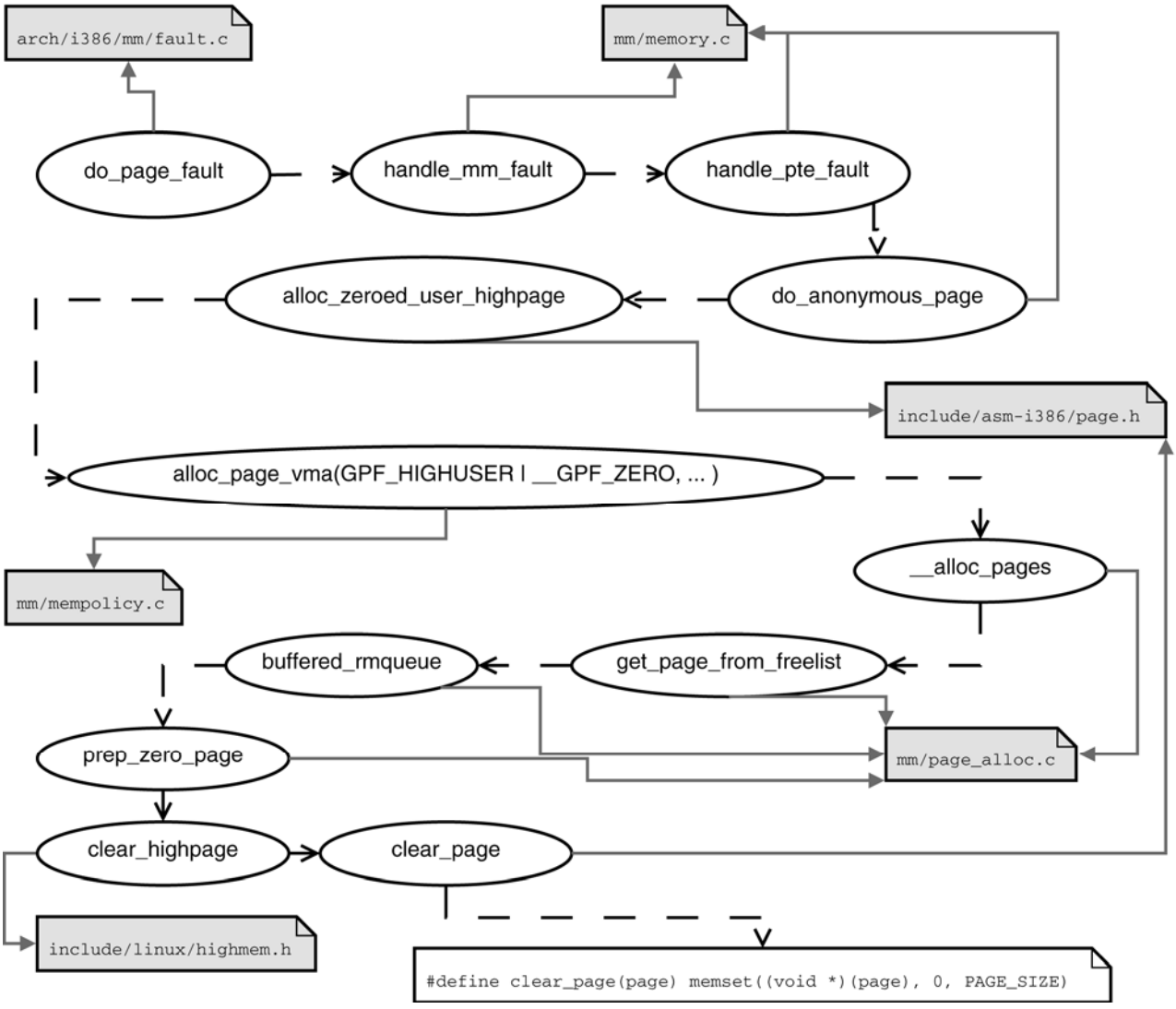

Рис. 2. Схема викликів функцій перед першим звертанням процесу до виділеної сторінки пам'яті

або 256 бітів, у режимі шифрування CBC (зчеплення блоків шифротексту). Для розділів підкачки застосовуються одноразові ключі – а саме, при кожному монтуванні розділу підкачки автоматично генерується значення нового ключа шифрування. Оскільки монтування розділу підкачки здійснюється при кожному завантаженні ОС, один ключ шифрування використовується виключно впродовж одного сеансу роботи КС.

Інсталяція пакету loop-aes для дистрибутиву Debian ОС GNU/Linux не становить труднощів. Усі необхідні кроки описані в інструкції адміністратора КС, яка входить до складу пакету.

Нарешті, розглянемо механізм захисту інформації, яка міститься в тимчасових файлах. Як зазначалося при постановці задачі, стандартний механізм захисту, який застосовується у ОС типу Unix, полягає в автоматичному очищенні директорії /tmp, яка містить тимчасові файли при кожному завантаженні системи.

Однак цей механізм слугує лише для обмеження часу життя тимчасових файлів і не здатний протидіяти реалізації загрози збирання сміття впродовж сеансу роботи КС. Дійсно, специфіка використання директорії /tmp зумовлює те, що ця директорія доступна для всіх користувачів. Права доступу, які встановлюють програми при створенні тимчасових файлів, цілком залежать від цих програм. Використання команди umask [6] для маскування прав доступу, що встановлюються за замовчуванням, не можна вважати адекватним рішенням з огляду на такі причини. По-перше, користувач або програма можуть змінити значення маски, яке було задане адміністратором КС. По-друге, користувач або програма можуть змінити права доступу файла, що був створений. Отже, ми не можемо гарантовано виключити некоректне призначення прав доступу, тому цілком реальною є ситуація, коли один користувач матиме можливість для несанкціонованого доступу до тимчасового файла, створеного програмою, яка діє від імені іншого користувача.

Для боротьби із таким потенційним джерелом компрометації автори пропонують використати концепцію багатоекземплярних директорій /tmp [7]. Зміст поняття полягає у тому, що для кожного користувача створюється окрема піддиректорія у директорії /tmp, власником якої є цей користувач. Ім’я піддиректорії має вигляд /tmp/tmp-inst/tmp-<uid>, де <uid> – значення ідентифікатора користувача. Ключовою особливістю є те, що користувацькі процеси звертаються, як і раніше, до директорії /tmp. При цьому відбувається розв’язання імені /tmp в ім’я /tmp/tmp-inst/tmp-<uid>. Справжній вміст директорії /tmp приховується. Таким чином, директорія /tmp для кожного користувача містить лише такі файли, що були створені процесами, які діяли від його імені. Доступ до тимчасових файлів інших користувачів стає неможливим завдяки тому, що кожна директорія з ім’ям вигляду /tmp/tmp-inst/tmp-<uid> доступна лише для її власника.

Реалізація схеми заснована на використанні модуля pam_namespace бібліотеки PAM [8]. Модуль автоматично і прозоро виконує вищеописані дії. Його конфігурування не викликає труднощів і не є предметом роботи.

Висновки

Авторами був проведений спеціальний аналіз базової ОС GNU/Linux та розширення безпеки RSBAC на предмет дотримання вимог, які висуває нормативний документ [2] щодо реалізації послуги «КО-1. Повторне використання об’єктів». Дослідження виявило, що деякі об’єкти КС не охоплюються контролем щодо повторного використання. Нагадаємо, що такими об’єктами є атрибути користувачів, файли при видаленні і врізанні, розділ підкачки та тимчасові файли. Для розширення сфери застосування послуги вважаємо за необхідне вжити таких додаткових заходів:

1) скористатись модулем FF системи RSBAC для встановлення контролю за видаленням і врізанням файлів; на розгляд вноситься пропозиція розширення адміністративного контролю системи

RSBAC для автоматичного визначення файлів, які потребують операції безпечного видалення і врізання на основі значення їхніх міток доступу;

2) використати стороннє ПЗ для прозорого шифрування файла підкачки;

3) використати модуль pam_namespace бібліотеки PAM для створення багатоекземплярних директорій, де зберігатимуться тимчасові файли.

Підготовка роботи супроводжувалась реалізацією та дослідною експлуатацією запропонованих рішень у ході розробки захищеної ОС на базі дистрибутиву Debian ОС GNU/Linux. Подальша робота над послугою полягає у модифікації системних утиліт для роботи із обліковими записами користувачів, що забезпечить повне впровадження послуги безпеки «КО-1. Повторне використання об'єктів».

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. *НД* ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. – К.: ДСТСЗІ СБ України, 1999. – 59 с.
3. *Ott A.* Regelsatz-basierte Zugriffskontrolle nach dem „Generalized Framework for Access Control" Ansatz am Beispiel Linux. – Diplomarbeit. Universität Hamburg, 10. November 1997. http://www.rsbac.org/doc/media/dipl-ps.zip.
4. *LaPadula L.J.* Essay 9: Rule-Set Modeling of a Trusted Computer System // Information Security: An Integrated Collection of Essays / Ed. by M. D. Abrams, S. Jajodia, H. J. Podell. — Los Alamitos, California, USA: IEEE Computer Society Press. http://www.acsac.org/secshelf/book001/09.pdf
5. http://loop-aes.sourceforge.net
6. *Сивер Э., Спейнауэр С., Фиггинс С., Хекман Д.* Linux. Справочник / Пер. с англ. – СПб: Символ-Плюс, 2001. – 912 с.
7. *Romans R.* Improve security with polyinstantiation using a Pluggable Authentication Module to protect private data // IBM developerWorks, 26 Feb 2008. – 9 p.
8. *Morgan A., Kukuk T.* The Linux-PAM System Administrators' Guide. http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_SAG.html

Отримано 12.01.2010

Про авторів:

Анісімов Анатолій Васильович,
доктор фізико-математичних наук,
член-кореспондент НАН України,
завідувач кафедрою математичної інформатики КНУ ім. Тараса Шевченка,

Іванніков Євген Юрійович,
аспірант кафедри математичної інформатики КНУ ім. Тараса Шевченка.

Місце роботи авторів:

Київський національний університет
ім. Тараса Шевченка,
Україна, 03680, Київ-680,
Проспект Академіка Глушкова, 2,
корпус 6.
Тел.: 38 044 259 0427
E-mail:
ava@unicyb.kiev.ua
ivannikoff@meta.ua