УДК 004.7:004.273

Є.В. Красовська, канд. техн. наук

ПРОГРАМНИЙ КОМПЛЕКС МОНІТОРИНГУ АКТИВНОСТІ КОРИСТУВАЧІВ КОРПОРАТИВНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ

Запропоновано підхід до створення та використання програм моніторингу діяльності та активності користувачів у комп'ютерній корпоративній мережі.

***Ключові слова**: комп'ютерна мережа, інформаційна безпека, програмний комплекс, моніторинг активності користувачів.*

E. V. Krasovskaya, PhD

CORPORATE COMPUTER NETWORK USERS' ACTIVITY MONITORING SOFTWARE SYSTEM

An approach of creation and monitoring programs usage of user activity in a computer corporative network.

***Keywords**: computer network, information security, software system, monitoring user activity.*

Е. В. Красовская, канд. техн. наук

ПРОГРАММНЫЙ КОМПЛЕКС МОНИТОРИНГА АКТИВНОСТИ ПОЛЬЗОВАТЕЛЕЙ КОРПОРАТИВНОЙ КОМПЬЮТЕРНОЙ СЕТИ

Предложен подход к созданию и использованию программ мониторинга деятельности и активности пользователей в компьютерной корпоративной сети.

***Ключевые слова:** компьютерная сеть, информационная безопасность, программный комплекс, мониторинг активности пользователей.*

Актуальність теми. Захист інформації в сучасних комп'ютерних інформаційних системах (ІС) є пріоритетним завданням. Викрадення конфіденційної інформації, знищення даних, викривлення інформації, виведення з ладу комп'ютерних систем (КС) – далеко не повний перелік усіх ризиків, що виникають у процесі експлуатації та використання сучасних ІС.

Комплексний характер системи безпеки для протидії різноманітним загрозам ІС має забезпечувати контроль за діяльністю службовців, які використовують різноманітні внутрішні ресурси системи, а також мають доступ до Інтернет-додатків.

Репутація і безпека сучасних компаній багато в чому залежить від діяльності її співробітників, а системи контролю і введення обмежень для персоналу є важливою складовою комплексу заходів, які спрямовані на підтримку інформаційної безпеки (ІБ).

В цьому напрямку актуальними є не тільки обмеження та фільтрація ресурсів і мережевих сервісів Інтернет з метою захисту корпоративної мережі (КМ), але й система, що контролює всі дії користувачів з метою подальшого аналізу й організаційних висновків.

Контроль за використанням комп'ютерів і пристроїв у мережі має за мету попередити несанкціонований доступ як до КМ в цілому, так і до окремих об'єктів спільного доступу, таких як мережеві файлові системи (ФС), директорії з конфіденційною інформацією, бази даних (БД), та запобігти втраті чи розголошенню цінної та важливої інформації.

Актуальним завданням є також моніторинг дій системних адміністраторів, які зазвичай можуть мати необмежені повноваження в системах та іноді стають джерелом витоку даних з компаній. При цьому важливо мати відповіді на ряд запитань: хто і коли працював у КС; хто мав доступ до БД та ФС спільного доступу; чим займаються співробітники в певний час; у разі надзвичайної події – чому деякі сервіси перестали бути доступними; які зміни в конфігурації було зроблено, з якої причини і ким?

Повний моніторинг дій користувачів КС дасть змогу отримати повну інформацію, що може бути використана для різноманітних висновків за результатами аналізу діяльності користувачів та результатів їх роботи.

Порівняльний огляд існуючих рішень. Популярною технологією віддаленого доступу до робочої станції є *VNC*, характерною особливістю якої є можливість організації кількох «точок підключення» на одному сервері.

Технологія *SELinux* – це розширення базової моделі безпеки ОС *Linux*, що додає механізм мандатного доступу. Вона є дуже гнучкою та потужною системою керування правами доступу, та має можливість реєстрації зловмисних дій, але вона має дуже складний синтаксис конфігураційних файлів та потребує досвіду в області ПЗ.

Іншою системою розмежування доступу є базова система контролю *RSBAC*, яка дозволяє створити на базі дистрибутиву *Linux* захищену ОС. Система *RSBAC* є одночасно досить гнучкою та багатогранною, окрім того, вона не настільки громіздка, як *SELinux*, тому може використовуватись більш широко.

Найпростішою та найдоступнішою для користувача є система розмежування прав доступу *AppArmor*, яка дасть змогу майже повністю переключити себе в режим стеження, але певний серверний додаток вимагає більш складної конфігурації, а контроль мережевих з'єднань в цій системі реалізовано на зародковому рівні, отже, для цього доведеться використовувати стороннє ПЗ.

З аналізу існуючих комплексів стає зрозуміло, що їх використання як системи спостереження вимагає, по-перше, глибокої реконфігурації та відключення непотрібних модулів, по-друге, кожен з них не є універсальним та потребує використання додаткового ПЗ. Іншою особливістю систем розподілення є необхідність виконання в режимі ядра, що накладає суттєві обмеження на вибір дистрибутиву та версії ядра.

Формулювання мети. Метою статті є проектування та розроблення програмного комплексу (ПК) для моніторингу дій користувачів КМ та проведення випробувань розробленого програмного забезпечення (ПЗ).

Постановка завдання. При розробленні ПК доцільно розподілити роботу на кілька етапів.

На першому створюється логічна схема ПК, визначається функціональне навантаження кожного автономного вузла, методи їх взаємодії та синхронізації.

На другому етапі для кожної автономної одиниці на основі її функціонального навантаження будується алгоритм роботи.

На третьому етапі створюється програмна схема модулів комплексу, проводиться вибір та враховуються особливості мови програмування, бібліотек, операційної системи (ОС) та апаратної платформи, вказуються місця та методи міжпроцесної взаємодії, описуються платформоспецифічні операції введення-виведення.

На останньому етапі проводиться безпосередньо написання вихідного коду, його оптимізація, налагодження та остаточне збирання програмних модулів.

Розроблення програмних модулів. Програмний комплекс моніторингу активності користувачів має клієнт-серверну архітектуру. Під клієнтом розуміється сукупність програмних компонентів, які встановлюються на підконтрольній КС (робоча станція, багатокористувацький сервер, тощо), та виконують реєстрацію певних подій, згідно налаштувань. Серверна ж частина встановлюється на виділеному сервері, вона призначена для отримання інформації від агентів та забезпечення її зберігання у певному форматі (або форматах).

На рис. 1 показано структуру ПК. Загальну структуру серверної частини зображено на рис. 2.

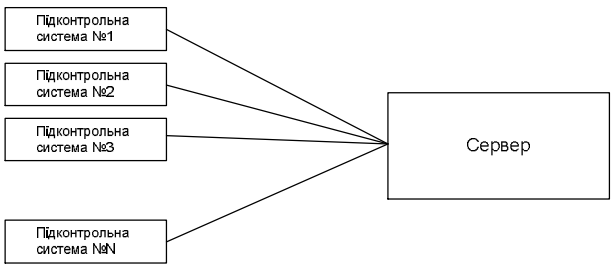

Рис. 1. Структура ПК

Прослуховування мережі, виконання авторизації, а також отримання та мультиплексацію потоків даних виконує серверний модуль, що являє собою резидентну програму (демон). Стартуючи, він зчитує файл налаштувань та запускає потрібні додаткові компоненти. Отримані дані передаються допоміжним компонентам – модулям виведення. Їх функція полягає у програмному розбиранні даних та забезпеченні їх зберігання. В даному ПК модуль виведення використовує для зберігання отриманої інформації текстові файли. При цьому для кожної підконтрольної машини створюється окремий файл.

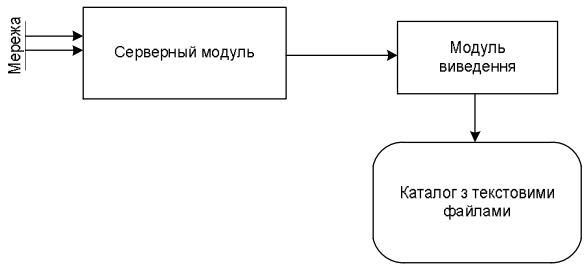

Рис. 2. Загальна структура серверної частини

Для гарантування безпеки каналу зв'язку між сервером та моніторинговими модулями використовується асиметричне шифрування каналу за допомогою технології *SSL* – криптографічного протоколу, який забезпечує встановлення безпечного з'єднання між клієнтом і сервером. Протокол уможливлює конфіденційність обміну даними між клієнтом і сервером, які застосовують протокол *TCP*/*IP*, причому для шифрування використовується асиметричний алгоритм з відкритим ключем (ВК). При шифруванні з ВК потрібно два ключі, причому будь-який з них може застосовуватися для шифрування повідомлення. Тим самим, якщо використано один ключ для шифрування, то відповідно для розшифрування потрібно інший ключ.

Сеанси зв'язку шифруються за допомогою алгоритму *AES*. Імена машин та шляхи до відповідних ключів прописані в конфігураційних файлах клієнтів та сервера. Для авторизації та подальшого криптографічного захисту каналу зв'язку використовується бібліотека *OpenSSL*.

У загальному випадку процедура авторизації виглядає так: після встановлення *TCP*-з'єднання *SSL*-клієнт і сервер виконують процедуру «рукостискання». Під час цього клієнт і сервер «домовляються» про різні параметри, які будуть використані для забезпечення безпеки з'єднання. «Рукостискання» починається тоді, коли клієнт підключається до сервера *SSL*. Запит безпечного з'єднання являє собою список підтримуваних шифрів і хеш-функцій. З цього списку сервер вибирає самий сильний шифр і хеш-функцію, яку він також підтримує, і повідомляє клієнтів про прийняте рішення.

Сервер відсилає це рішення у вигляді цифрового сертифікату. Сертифікат, зазвичай, містить ім'я сервера, довірений Центр Сертифікації і ВК шифрування сервера. Клієнт може зв'язатися з сервером, який видав сертифікат, і переконатися перш, ніж продовжити, що сертифікат є справжнім.

Для того, щоб згенерувати ключі сеансу, використовується безпечне з'єднання. Клієнт шифрує випадкове число за допомогою ВК сервера і відправляє результат на сервер. Тільки сервер може розшифрувати його (з його закритим ключем (ЗК)), і тільки цей факт робить ключі прихованими від третьої сторони. Клієнт знає ВК і випадкове число, а сервер знає ЗК і (після розшифрування повідомлення клієнта) випадкове число. Третя сторона, можливо, знає тільки ВК, якщо ЗК не був зламаний. З випадкового числа обидві сторони створюють ключові дані для шифрування і розшифрування.

На цьому «рукостискання» завершується і починається захищене з'єднання, яке зашифровується і розшифровується за допомогою ключових даних. Якщо будь-яка з перерахованих вище дій виконується невдало, то «рукостискання» *SSL* не вдалося і з'єднання не відбувається. В такому разі протоколювання подій починається в локальному режимі.

На рис. 3 зображено загальну структуру клієнтської частини комплексу.

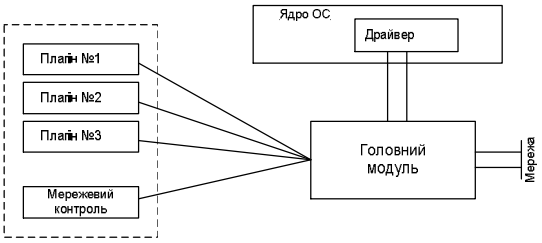

Рис. 3. Структура клієнтської частини ПК

Центральне місце в ній посідає модуль-колектор, який відповідає за клієнт-серверну взаємодію, а також збирає та консолідує дані, отримані від інших компонентів. Реєстрація подій ФС на низькому рівні може бути здійснена виключно в режимі ядра, тому цей функціонал забезпечується завантажуваним модулем. Для передавання даних з простору ядра в користувацький простір використано спеціальний шлюз віртуальної ФС.

Реєстрація подій встановлення та розірвання *TCP*-з'єднань винесена в простір користувача. Це, по-перше, знизить навантаженні на систему при великих значеннях утилізації процесорного часу, по-друге, дозволить ефективніше використовувати механізми мережевої підсистеми. Зазначимо, що для нашого випадку (реєстрація лише подій встановлення та розірвання з'єднань) перехоплення системного виклику, по аналогії із файловим доступом, є неоптимальним і не може бути використане. Для реєстрації подій встановлення та розірвання *TCP*-з'єднать нам достатньо можливостей, які надає підсистема *Conntrack*. Для забезпечення розроблення та виконання цього модуля в системі мають бути встановлені пакети: *libnetfilter-conntrack, conntrack, libnetfilter-dev*.

Система плагінів дозволяє значно розширювати можливості контролю: за допомогою спеціальних модулів можна проводити протоколювання *SQL*-транзакцій, робити періодичні знімки екрану тощо.

Для перехоплення подій доступу до підконтрольних файлових об'єктів застосовується спеціальний завантажуваний модуль режиму ядра системи (драйвер). На верхньому рівні *VFS* розташовується єдина *API*-абстракція таких функцій, як відкриття, закриття, читання та запис файлів. На нижньому рівні *VFS* знаходяться абстракції ФС, щоі визначають, як реалізуються функції верхнього рівня.

Нижче рівня ФС знаходиться кеш буферів, що надає загальний набір функцій до рівня ФС. Цей рівень кешування оптимізує доступ до фізичних пристроїв за рахунок короткострокового зберігання даних. Нижче кешу буферів знаходяться драйвери пристроїв, які реалізують інтерфейси для конкретних фізичних пристроїв.

На програмному рівні перехоплення системних викликів реалізується заміщенням відповідної адреси обробника в системній таблиці *sys_call_table*.

Програмний комплекс моніторингу активності користувачів створено за допомогою програмних засобів ОС *GNU*/*Linux*. Операції стеження за файловими об'єктами мають виконуватись на досить низькому рівні, тому компонент, що їх обслуговує, оформлено у вигляді драйвера і виконується в режимі ядра ОС. Для контролю мережевої підсистеми також використано компоненти ОС *Conntrack*, який дозволяє програмам отримувати інформацію про з'єднання та налаштовувати фільтрацію трафіку без використання драйверів. Це покращує інтеграцію в ОС, дає змогу уникнути небажаного погіршення швидкості відклику системи та проблем сумісності із драйверами специфічного мережевого обладнання. Також проект має підтримку додаткових модулів, що розширюють його функціональність – плагінів.

Серверна частина, для покращення швидкодії та зниження потреби в системних ресурсах (зокрема, ОП), реалізована за допомогою моделі програмних потоків (*Treads*).

Тестування ПК. Тестування ПК – це обов'язкова процедура, яка вимагає проведення в умовах близьких до робочих. Під час тестування можна оцінити як загальну стабільність програмних модулів, їх коректну взаємодію з системним та прикладним ПЗ, такжі ступінь навантаження на систему. Останній фактор може бути важливим при технічному об-

ґрунтуванні використання комплексу на багатокористувацькій, високонавантаженій системі. Для цього проведемо навантажувальне тестування на робочій КС із використанням типового графічного користувацького оточення. Як об'єкти моніторингу застосовуватимуться змонтована ФС та мережеві з'єднання (типова ситуація при використанні таких програм, як перегляд веб-сторінок в браузері, операції завантаження файлів тощо).

Для навантажувального тестування виберемо типовий комп'ютер, який виконуватиме роль клієнтської платформи.

Його апаратні характеристики:
центральний процесор – *AMD Phenom II* X2 565, тактова частота 3.4 ГГц;
материнська плата – *Gigabyte GA*-880*GA-UD*3*H;*
оперативна пам'ять – *DDR*3-1333, 3 ГБ;
жорсткий диск – *WD Caviar Black*, 500 Гб;
мережевий адаптер – вбудований, *Realtek* 8111*D.*

Тестова робоча станція підключена до КМ з використанням технології 100 *MBit Ethernet* та має доступ до мережі Інтернет на швидкості до 10 Мбіт/с.

Як ПЗ використовується ОС *Debian GNU*/*Linux.* Версія ядра: 3,2,0-*amd*64; версія системної бібліотеки: *EGlibc*: 2,13.

Потрібно зауважити, що на тестовому стенді використовується 64-бітна версія ОС *GNU*/*Linux* та всього системного і прикладного ПЗ. Незважаючи на все ще значне розповсюдження застарілого, 32-бітного ПЗ, воно не має можливості повноцінно використовувати сучасні апаратні ресурси, тому його застосування не є рекомендованим. Інші програмні засоби прикладного рівня, які використовувались на тестовому стенді: графічне оточення робочого столу *KDE* 4.6.5 та супутнє ПЗ; браузери: *Google Chrome* 12, *Mozilla Firefox* 9; поштовий клієнт *IceDove* 3.1.16; консольний менеджер завантаження *WGet*; офісний пакет *LibreOffice* 3.4.5; компілятор *GCC* 4.6.2.

На момент тестування версії ПЗ відповідала найновішим у тестовій гілці дистрибутива *Debian GNU*/*Linux*. Розділи жорсткого диску були відформатовані у ФС *XFS*, яка спочатку була розрахована для використання на дисках великого обсягу і досить ефективно працює як із файлами великого, так і малого розміру, в тому числі при групових операціях із великою кількістю файлових об'єктів.

Безпосередньо тестування ПК відбувалось у три етапи. На кожному з них відтворювалась одна з трьох типових ситуацій: праця офісного працівника, тобто переважна робота з текстовими документами та електронною поштою; ситуація, що відтворює інтенсивне використання дискового введення-виведення та значне мережеве навантаження (паралельне завантаження файлів великого розміру); відтворення роботи програміста (компіляція великих обсягів вихідного коду). Реєстрації підлягали файлові об'єкти та мережеві *TCP*-з'єднання.

Перший етап полягав у відтворенні ситуації, коли за робочою станцією працює офісний співробітник, основна робота якого відбувалась у програмах офісного пакета *OpenOffice,* браузері та поштовому клієнті. Тривалість випробування становила 6 годин. Для замірів показників навантаження процесора, жорсткого диска, використання розділу підкачки та графічного відображення отриманої інформації використовувались скрипти, що отримували значення від стандартних утиліт, а також через файловий інтерфейс/*proc*, з подальшим аналізом та побудовою графіків у пакеті *OpenOffice*. На рис. 4 і 5 подано рівень використання процесора в режимі користувача та ядра, та інтенсивність використання дискового введення-виведення.

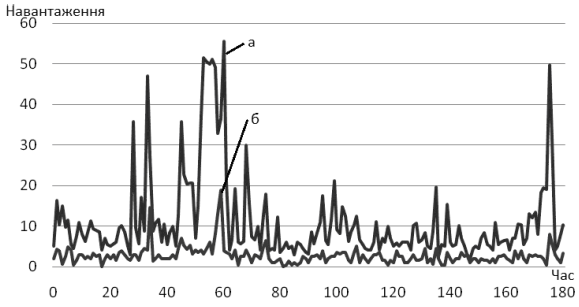

Рис. 4. Показники навантаження процесора:
а – в режимі користувача, %;
б – в режимі ядра, %

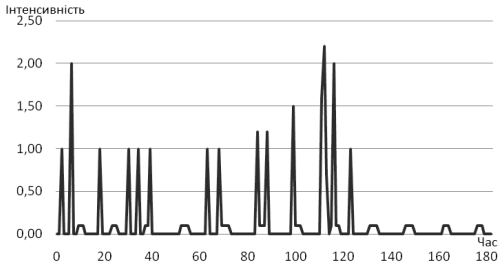

Рис. 5. Інтенсивність дискових операцій

На рис. 6 та 7 відповідно відображено показники, аналогічні вище поданим, але зафіксовані на системі без використання ПЗ моніторингу користувацької активності.

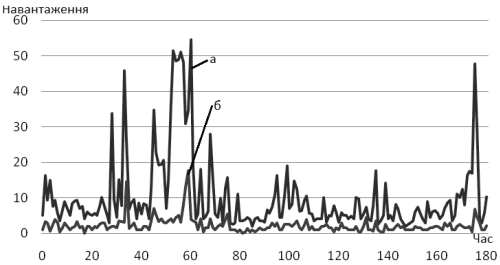

Рис. 6. Показники навантаження процесора (без ПЗ моніторингу): а – в режимі користувача, %; б – в режимі ядра, %

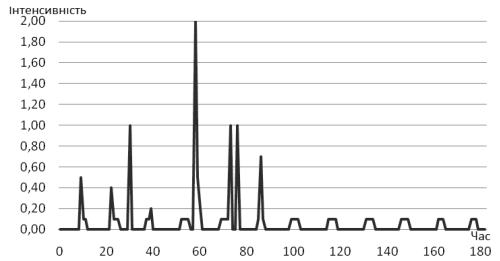

Рис. 7. Інтенсивність дискових операцій (без ПЗ моніторингу)

Як видно з графіків, використання розробленого ПК на машині, якщо завантаження нижче середнього (типова офісна робоча станція), збільшує використання процесорного часу на 2 – 3 %, що є майже невідчутним для користувача. Збільшення дискової активності на 4 – 6 % пов'язане з тим, що клієнтська та серверна частини були встановлені на одній і тій самій тестовій машині. Ці додаткові звертання до накопичувачів виникали, як правило, під час операцій звертання до користувацьких файлів. Використання ж ОП не змінилось.

Другий етап полягав у відтворенні ситуації, коли, поряд із значним навантаженням на дискову підсистему, має місце інтенсивне передавання даних через КМ. Для цього виконувалось паралельне завантаження файлів через КМ (одночасно 10 потоків) та зберігання їх на підконтрольний розділ жорсткого диску. Розмір файлів варіював від 20 Мб до 50 Гб. Файли дрібного розміру потрібні для виявлення наявних «вузьких місць» при роботі з метаданими ФС. Тривалість експерименту становила 6 год. Під час випробування проводились заміри навантаження процесора в режимі користувача та ядра (рис. 8) та навантаження дискової підсистеми (рис. 9). На рис. 10 та 11, відповідно, подано аналогічні показники, отримані для системи без використання розробленого ПК.

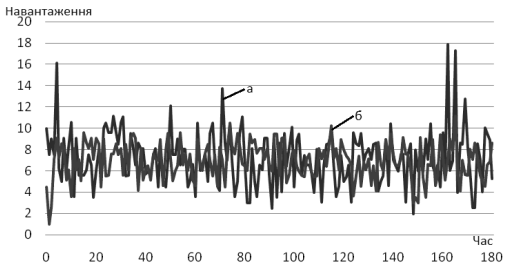

Рис. 8. Показники навантаження процесора: а – в режимі користувача, %; б – в режимі ядра, %

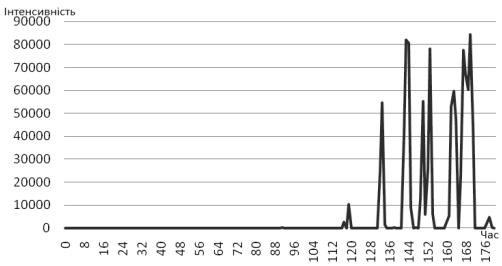

Рис. 9. Інтенсивність дискових операцій

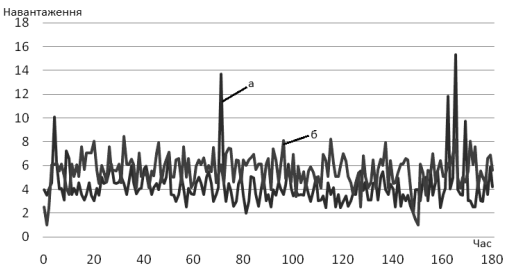

Рис. 10. Показники навантаження процесора (без ПЗ моніторингу): а – в режимі користувача, %; б – в режимі ядра, %

Як видно з графіків, в даному випробуванні реєструється відчутне збільшення використання процесорного часу. Причому пік його припадає на операції з великою кількістю дрібних файлів. Однак, якщо збільшення завантаження процесора в даному випадку невідчутне для користувача, то інтенсивні операції з файлами були більш «помітні». На графіку реєструється збільшення дискової активності під час операцій із дрібними файлами на 4 – 6%, на операціях із великими файлами різниця не перевищувала 3 – 4 %. Реєструється збільшення завантаженості системи в режимі користувача (рис. 8, 10), що пояснюється активністю моніторингу мережевої підсистеми.

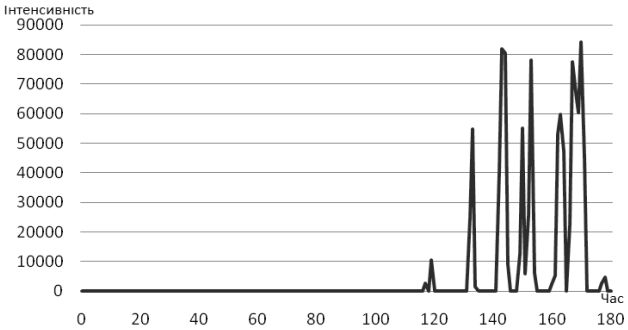

Рис. 11. Інтенсивність дискових операцій (без ПЗ моніторингу)

Третій етап полягав у відтворенні роботи програміста. Для цього виконувався збір з вихідних кодів ядра системи та повного графічного середовища *KDE*4 за допомогою штатного компілятора *GCC*. Тривалість експерименту становила близько 3 год., що відповідає повній компіляції вищезгаданих програмних продуктів.Під час роботи компілятор, по-перше, створює значне навантаження на процесор, по-друге, при компіляції складних проектів оперує великою кількістю дрібних тимчасових файлів, створюючи відповідне навантаження на дискову підсистему.

Під час навантаження були виміряні завантаження процесора системи режиму користувача та ядра (рис. 12), та навантаження дискової підсистеми (рис. 13). На рис. 14 та 15, відповідно, подано аналогічні показники, отримані для системи без використання розробленого ПК.

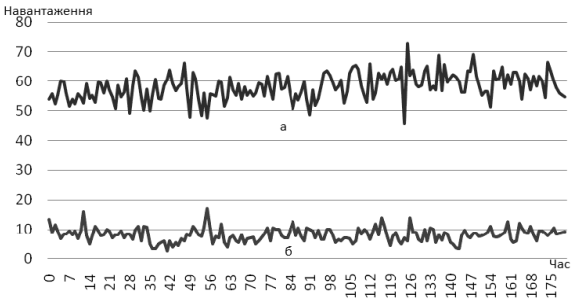

Рис. 12. Показники навантаження процесора: а – в режимі користувача, %; б – в режимі ядра, %

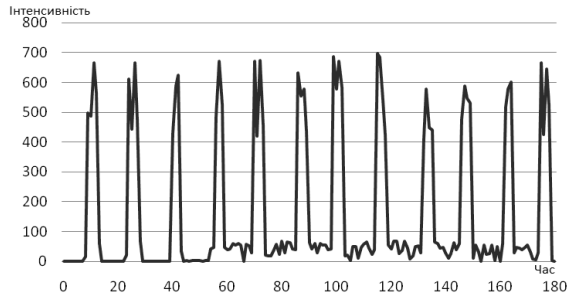

Рис. 13. Інтенсивність дискових операцій

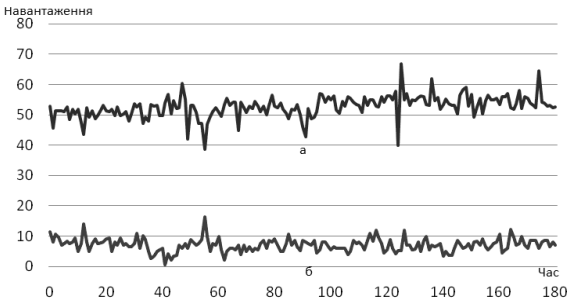

Рис. 14. Показники навантаження процесора (без ПЗ моніторингу) : а – в режимі користувача, %; б – в режимі ядра, %

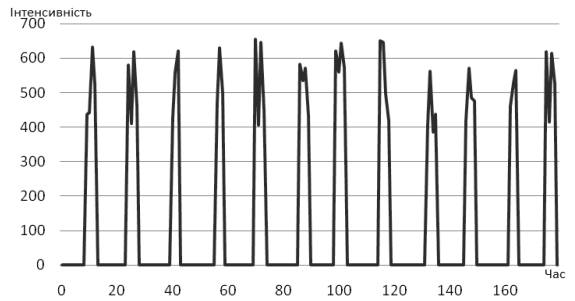

Рис. 15. Інтенсивність дискових операцій (без ПЗ моніторингу)

Як видно із графіків, збільшується завантаження процесора, пов'язане з моніторингом файлової активності, а також відповідно зростає завантаження системи режиму ядра. Також реєструється помітне збільшення активності дискової підсистеми, що пояснюється записом зареєстрованих змін у ФС. Суб'єктивно процес компіляції великих проектів можна характеризувати як досить «важке» завдання, яке саме по собі створює відчутне навантаження та сповільнює швидкість реакції системи, і в даному випадку використання комплексу відчутним не було.

Висновки. На основі аналізу загроз та проблем адміністрування мереж обґрунтована актуальність розроблення та використання програмних засобів прихованої реєстрації дій користувачів. Реалізовано ПЗ моніторингу діяльності та активності на комп'ютерах корпоративної мережі компанії, що використовує ОС *GNU/Linux*.

Система моніторингу дій користувачів КМ дасть змогу отримати повну інформацію, що може бути використана для висновків за результатами аналізу діяльності користувачів та результатів їх роботи. Програмний комплекс дозволяє проводити контроль за активністю ПЗ, в тому числі пропрієтарного, яке може бути потенційним джерелом інформаційних загроз та підтримувати рівень безпеки на відповідному рівні.

Було проведене тестування ПК для моніторингу користувацької активності в умовах моделювання трьох типових випадків: робота офісного персоналу, активне навантаження дискової та мережевої підсистем, а також компіляція складних програмних продуктів. В усіх випадках реєструвалось певне збільшення використання ресурсів процесора та дискової підсистеми. Але, по-перше, різниця була досить помірна (до 10 % навіть при піковому навантаженні), а по-друге, суб'єктивне враження операторів від роботи на підконтрольній системі дозволяє говорити про малопомітний вплив операцій моніторингу на швидкість відгуку системи. Тому можна зробити висновок, що розроблений ПК може бути рекомендований для використання в КС з багатокористувацьким доступом, в тому числі у вузлах, де планується значне навантаження. З іншого боку, при подальшому розвитку даного ПЗ постає необхідність додаткової оптимізації роботи із дисковою підсистемою, а також можливої оптимізації модуля моніторингу мережевої активності, в тому числі його адаптації до сучасних програмних засобів новіших ядер *Linux*.

Список використаної літератури

1. Вейрле, К. *Linux*: сетевая архитектура. Структура и реализация сетевых протоколов в ядре / К. Вейрле, Ф. Пэльке. – М. : КУДИЦ-Образ, 2006. – 656 с.

2. Лав, Р. Разработка ядра *Linux* / Р. Лав. – Пер. с англ. – М. : ООО Вильямс, 2008. – 448 с.

3. Мельников, В. Защита информации в компьютерних системах / В. Мельников. – М. : Финанси и статистика, 1997. – 368 с.

4. Немет, Э. Руководство администратора *Linux* / Э. Немет, Г. Снайдер, Т. Хейн. – [2-е изд.]; пер. с англ. – М. : ООО Вильямс, 2008. – 1072 с.

Отримано 08.09.2012

References

1. Veyrle, K. Linux: Network architecture. The structure and implementation of network protocols in the kernel / K. Veyrle, F. Pelke. – Moscow: KUDITS–Obraz, 2006. – 656 p. [in Russian].

2. Lav, R. Linux core Development / R. Lav. – Eng. trans. – Moscow : Vil'yams, 2008. – 448 p. [in Russian].

3. Melnikov, V. Protection of information in computer systems / V. Melnikov. – Moscow : Finance and Statistics, 1997. – 368 p. [in Russian].

4. Nemet, E. Administrator's Guide Linux / E. Nemet, G. Snayder, T. Heyn – [2nd ed.]; Eng. trans – Moscow : Vil'yams, 2008. – 1072 p. [in Russian].

Красовська
Євгенія Вікторівна,
к.т.н., доц. каф. Комп'ютерних систем та мереж Нац. авіац. ун-ту.
Україна, м. Київ.
Тел. 044-406-76-78
E-mail: pretty1001@yandex.ru