ЗМІСТ

ВСТУП

Комп'ютерна безпека - заходи безпеки, які застосовуються для захисту обчислювальних пристроїв (комп'ютери, смартфони та інші), а також комп'ютерних мереж (приватних і публічних мереж, включаючи Інтернет). Поле діяльності системних адміністраторів охоплює всі процеси і механізми, за допомогою яких цифрове обладнання, інформаційне поле і послуги захищаються від випадкового або несанкціонованого доступу, зміни або знищення даних, і набуває все більшого значення у зв'язку зі зростаючою залежністю від комп'ютерних систем в розвиненому суспільстві.

Кібербезпека - процес використання заходів безпеки для забезпечення конфіденційності, цілісності та доступності даних. Системний адміністратор забезпечує захист активів, включаючи дані локальної мережі комп'ютерів, серверів. Крім того, під охорону беруться безпосередньо будівлі та, найголовніше, персонал. Метою забезпечення кібербезпеки є захист даних (як в процесі передачі і / або обміну так і знаходяться на зберіганні). З метою забезпечення безпеки даних можуть бути застосовані і контрзаходи. Деякі з цих заходів включають (але не обмежуються) контроль доступу, навчання персоналу, аудит і звітність, оцінку можливих ризиків, тестування на проникнення і вимога

Важливим розділом Кібербезпеки є IT Аудит. Аудит інформаційних технологій або аудит інформаційних систем - це перевірка засобів управління в рамках інфраструктури інформаційних технологій (IT). Оцінка отриманих доказів визначає, чи є інформаційні системи захищати активи, зберігати цілісність даних та ефективно працювати для досягнення цілей організації. Ці огляди можуть виконуватися спільно з аудитом фінансової звітності, внутрішнім аудитом або іншою формою атестаційної участі.

Аудит інформаційних систем (IC) - це дослідження і аналіз існуючої інформаційної системи на підприємстві, що дозволяє оцінити її ефективність. Аудит IC проводиться в ході комплексного IT-аудиту та ставить за мету

визначити рівень відповідності апаратно-програмних комплексів і поточних потреб бізнесу. Частиною аудиту ІС є Аудит Безпеки.

Аудит інформаційної безпеки - системний процес отримання об'єктивних якісних і кількісних оцінок про поточний стан інформаційної безпеки компанії відповідно до визначених критеріїв та показниками безпеки.

Зараз є декілька способів Аудиту безпеки ПК, вони достатньо потужні і мають великий функціонал, але основний їх недолік є у складності його налаштування і використання. Аудит безпеки включає в себе багато перевірок різних факторів, але найважливішим є Аудит системних подій. Отже варто звернути увагу на створені інструментів Аудиту системних подій, які будуть доступні та зрозумілі для кожного користувача з усуненням недоліків існуючих засобів.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Загальні відомості про Аудит системних подій

Аудит системних подій не забезпечує ніякого додаткового захисту, але надає докладну інформацію про порушення безпеки, на підставі якої можна вжити конкретних заходів.

Уся система Аудиту системних подій призначена для відстеження критичних з точки зору безпеки системних подій. Як приклади таких подій можна навести такі (список далеко не повний):

- Запуск і завершення роботи системи;
- читання, запис і зміна прав доступу до файлів;
- ініціація мережевих з'єднань;
- спроби невдалої авторизації в системі;
- зміна мережевих налаштувань;
- зміна інформації про користувачів і групи;
- запуск і зупинка додатків; виконання системних викликів.

Жодне з названих подій не може відбутися без використання системних викликів ядра. Щоб їх відслідковувати, досить просто перехоплювати відповідні системні виклики. Саме це і робить підсистема аудиту

1.2 Аудит безпеки в Windows 10

1.2.1 Вступ

Аудит безпеки є одним з найбільш потужних інструментів, які можна вибрати для підтримки цілісності вашої системи. Як компонент загальної стратегії забезпечення безпеки слід визначити рівень аудиту, який найкращим чином підходить для вашої середовища. Аудит повинен виявляти атаки (успішні чи ні), які становлять загрозу вашої мережі, а також атаки на ресурси, які ви визначили як цінні при аналізі ризиків.

1.2.2 Базові політики аудиту безпеки

Перед реалізацією аудиту необхідно вибрати політику аудиту. Базова політика аудиту визначає категорії пов'язаних з безпекою подій, які потрібно відстежувати. Якщо ця версія Windows встановлена вперше, відключені всі категорії аудиту. Включивши різні категорії подій аудиту, ви можете реалізувати політику аудиту, яка здатна задовольняти вимоги щодо безпеки вашої організації.

Можна вибрати такі категорії подій для аудиту.

- Події аудиту входу в обліковий запис
- Аудит управління обліковим записом
- Аудит доступу до служби каталогів
- Аудит події входу
- Аудит доступу до об'єктів
- Аудит зміни політики
- Аудит використання привілеїв
- Аудит відстеження процесу
- Аудит системних подій

Якщо ви вирішили відстежувати доступ до об'єктів у рамках політики аудиту, необхідно включити категорію аудиту доступу до служби каталогів

(Щоб перевіряти об'єкти на контролері домену) або категорію аудиту доступу до об'єктів (щоб перевіряти об'єкти на рядовому сервері або робочої станції). Після включення категорії доступу до об'єктів ви можете вказати відслідковують типи доступу для кожної групи або користувача.

1.2.3 **Створення базової політики аудиту для категорії подій**

Задавши параметри для певних категорій користувачів, ви можете створити політику аудиту, що відповідає вимогам до безпеки для вашої організації. На пристроях, приєднаних до домену, параметри аудиту для категорій подій не задані за замовчуванням. На контролерах доменів аудит включений за замовчуванням.

Щоб виконати цю процедуру, вам буде потрібно увійти від імені учасника вбудованої групи "Адміністратори". Установка або зміна параметрів політики аудиту для категорії подій на локальному комп'ютері

- Під час використання оснащення "Локальна політика безпеки" (secpol.msc) і виберіть Локальні політики.
- Натисніть Політика аудиту.

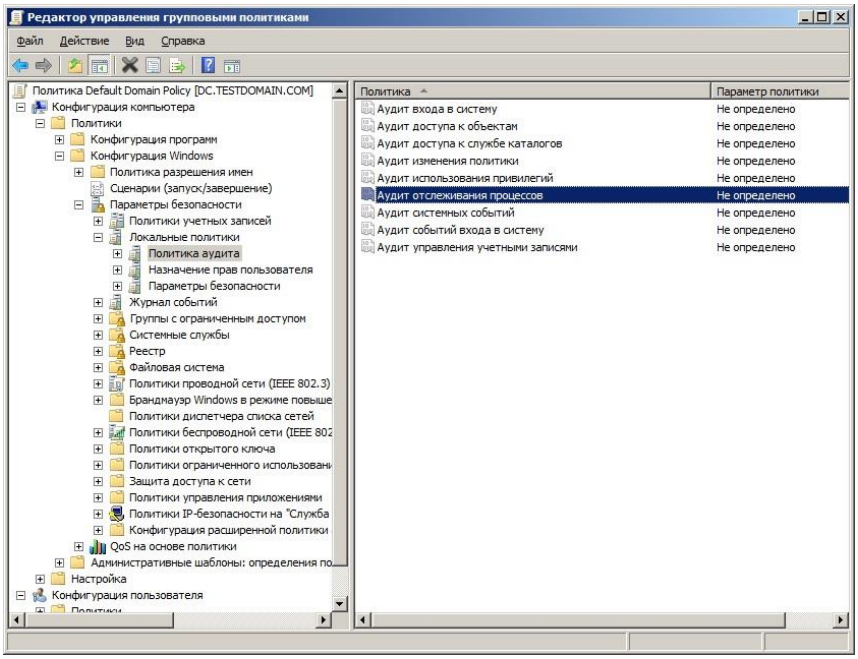

Рисунок 1.1. Приклад використання Локальної політики безпеки

- В області результатів двічі клацніть категорію подій, для якої потрібно змінити параметри політики аудиту.
- Виконайте одну або обидві наведених нижче дії, а потім натисніть кнопку ОК.
    - Для аудиту успішних спроб встановіть прапорець Успіх.
    - Для аудиту невдалих спроб встановіть прапорець Відмова.

Щоб виконати цю процедуру, необхідно увійти в систему від імені учасника групи "Адміністратори домену".

Установка або зміна параметрів політики аудиту для категорії подій домену або підрозділу, якщо ви працюєте на вхідному в підрозділ сервері або робочої станції, приєднаної до домену:

- Відкрийте консоль управління груповими політиками (GPMC).
- У дереві консолі перейдіть в ліс, а потім в домен і двічі клацніть вузол Об'єкти групової політики, що містить об'єкт Default Domain Policy, який потрібно змінити.
- Клацніть об'єкт Default Domain Policy правою кнопкою миші і виберіть пункт Змінити.
- У консолі управління груповими політиками перейдіть до розділу Конфігурація комп'ютера, Параметри Windows, Параметри безпеки і виберіть пункт Політика аудиту.
- В області результатів двічі клацніть категорію подій, для якої потрібно змінити параметри політики аудиту.
- Якщо ви вперше налаштовуєте параметри політики аудиту, встановіть прапорець Визначити наступні параметри політики.
- Виконайте одну або обидві наведених нижче дії, а потім натисніть кнопку ОК.
    - Для аудиту успішних спроб встановіть прапорець Успіх.
    - Для аудиту невдалих спроб встановіть прапорець Відмова.

Додаткові рекомендації:

- Для аудиту доступу до об'єктів, впевніться категорію подій, виконавши зазначені вище дії. Потім включите аудит для конкретного об'єкта.
- Коли політика аудиту буде налаштована, події будуть записані в журнал безпеки. Щоб переглянути ці події, відкрийте журнал безпеки.
- За замовчуванням для політики аудиту контролерів домену заданий параметр Немає аудиту. Це означає, що навіть якщо в домені включений аудит, контролери домену не успадковують політику аудиту локально. Щоб політика аудиту домену застосовувалася до контролерів домену, необхідно змінити цей параметр політики.

1.2.4 **Застосування базової політики аудиту до файлу або папці**

Ви можете застосовувати політики аудиту до окремих файлів і папок на комп'ютері, налаштувавши тип дозволу, щоб записувати успішні або невдалі спроби доступу в журнал безпеки.

Для завершення цієї процедури необхідно увійти в систему як член вбудованої групи «Адміністратори» або ж вам необхідно право Управління аудитом і журналом безпеки.

Застосування або зміна параметрів політики аудиту для локального файлу або папки:

- Клацніть правою кнопкою файл або папку, яку потрібно відстежувати, виберіть Властивості та відкрийте вкладку Безпека.
- Клацніть Додатково.
- У діалоговому вікні Додаткові параметри безпеки виберіть вкладку Аудит і натисніть кнопку Продовжити.
- Виконайте одну з таких дій.

- Щоб налаштувати аудит для нового користувача або групи, натисніть кнопку Додати. Клацніть Виберіть суб'єкт, введіть ім'я потрібного користувача або групи, а потім натисніть ОК.
- Щоб відключити аудит для існуючої групи або користувача, натисніть кнопку Видалити, клацніть ОК, а потім пропустіть решту процедури.
- Щоб переглянути або змінити налаштування аудиту для існуючої групи або користувача, клацніть потрібне ім'я і натисніть кнопку Змінити.

- У полі Тип вкажіть, які дії потрібно перевіряти, встановивши відповідні прапорці.
  - Для перевірки успішних подій клацніть Успіх.
  - Для перевірки подій відмови клацніть Відмова.
  - Для аудиту всіх подій виберіть пункт Усі.

Важливо Перш ніж настроювати аудиту файлів і папок, необхідно включити аудит доступу до об'єктів, визначивши параметри політики аудиту для категорії подій доступу до об'єктів. Якщо цього не зробити, ви побачите повідомлення про помилку під час налаштування аудиту файлів і папок, при цьому аудит файлів і папок буде недоступний.

Додаткові рекомендації:

- Після включення аудиту доступу до об'єктів перегляньте журнал безпеки в «Перегляді подій», щоб побачити результати змін.
- Ви можете налаштувати аудит файлів і папок тільки на дисках NTFS.
- Так як розмір журналу безпеки обмежений, необхідно ретельно вибирати файли і папки для аудиту. Також проаналізуйте обсяг дискового простору, яке потрібно виділити журналу безпеки. Максимальний розмір журналу безпеки визначається в «Перегляді подій».

1.2.5 **Перегляд журналу подій безпеки**

У журналі безпеки фіксуються всі події, певні політиками аудиту, які ви задаєте для кожного об'єкта.

Перегляд журналу безпеки:

- Під час використання засіб перегляду подій.
- У дереві консолі розгорніть Журнали Windows і натисніть Безпека. В області результатів перераховані окремі події безпеки.
- Щоб переглянути детальну інформацію про конкретну подію клацніть його в області результатів.

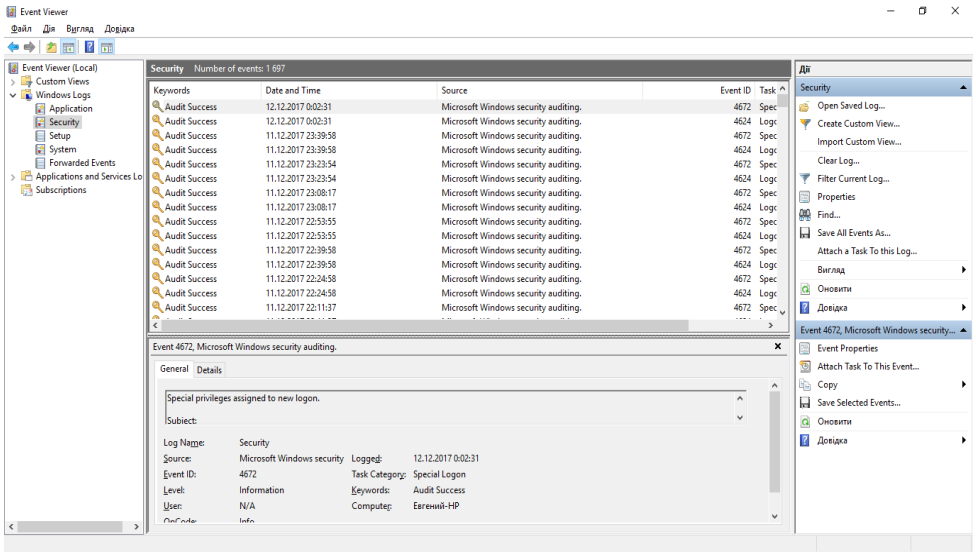

Рисунок 1.2. Приклад використання Журналу подій безпеки

1.2.6 **Параметри базової політики аудиту безпеки**

Параметри базової політики аудиту безпеки знаходяться в розділі "Конфігурація комп'ютера \ Параметри Windows \ Параметри безпеки \ Локальні політики \ Політика аудиту".

Аудит подій входу в систему

Визначає, чи підлягає аудиту кожна спроба користувача увійти в систему за допомогою іншого пристрою або вийти з неї, за умови що цей пристрій використовується для перевірки автентичності профілю.

Аудит управління обліковими записами:

Визначає, чи підлягають аудиту всі події, пов'язані з управлінням обліковими записами на пристрої.

Аудит доступу до служби каталогів:

Визначає, чи підлягає аудиту подія доступу користувача до об'єкта каталогу Active Directory, для якого задано власний системний список управління доступом (SACL).

Аудит події входу:

Визначає, буде виконуватися аудит кожної спроби входу користувача в систему або виходу з неї на цьому пристрої.

Аудит доступу до об'єктів:

Визначає, чи підлягає аудиту подія доступу користувача до об'єкта - наприклад до файлу, папки, розділу реєстру, принтеру і т. П., - для якого заданий власний системний список управління доступом (SACL).

Аудит зміни політики:

Визначає, чи підлягає аудиту кожен факт зміни політик призначення прав користувача, політик аудиту і політик довіри.

Аудит використання привілеїв:

Визначає, чи підлягає аудиту кожна спроба користувача скористатися наданим йому правом.

Аудит відстеження процесів:

Визначає, чи підлягають аудиту докладні відомості про відстеження таких подій, як активація програми, завершення процесу, дублювання дескрипторів і непрямий доступ до об'єкта.

Аудит системних подій:

Визначає, чи підлягають аудиту події перезавантаження або завершення роботи комп'ютера користувачем, а також події, що впливають на системну безпеку або журнал безпеки.

1.3 Аудит безпеки в Ubuntu

1.3.1 Установка

Щоб почати працювати з підсистемою аудиту, потрібно встановити пакет auditd (тут і далі наводяться приклади команд для ОС Ubuntu 14.04):

$ sudo apt-get install auditd

У состав цього пакета входять демон auditd і кілька допоміжних утиліт:

- auditctl - утиліта для управління демоном auditd; дозволяє отримувати інформацію про поточний стан підсистеми аудиту, а також додавати і видаляти правила;
- autrace - утиліта для аудиту подій, породжуваних процесами (працює за тим же принципом, що і strace);
- ausearch - утиліта для пошуку подій в журнальних файлах;
- aureport - утиліта для генерації звітів про роботу системи аудиту.

1.3.2 Конфігурація

Перш ніж почати роботу з підсистемою аудиту, відкриємо конфігураційний файл etc / audit / auditd.conf. Він містить в числі інших такі параметри:

- log_file - файл, в якому будуть зберігатися логи підсистеми аудиту;
- log_format - формат, в якому буде збережено логи;
- freq - максимальне число записів протоколу, які можуть зберігатися в буфері;
- flush - режим синхронізації буфера з диском (none - нічого не робити, incremental - переносити дані з буфера на диск з частотою, зазначеної в значенні параметра freq; data - синхронізувати негайно, sync - синхронізувати як дані, так і метадані файлу під час запису на диск);
- max_log_file - максимальний розмір файлу логу в мегабайтах;

- max_log_file_action - дія при перевищенні максимального розміру файлу лога;
- space_left - мінімум вільного простору в мегабайтах, після досягнення якого повинно бути здійснене дію, вказане в наступному параметрі;
- space_left_admin - вказує, що робити, коли на диску недостатньо вільного місця (ignore - нічого не робити; syslog - відправляти в syslog, email - відправляти повідомлення поштою; suspend - припинити запис логів на диск; single - перейти в одного користувача режим; halt - вимкнути машину);
- disk_full_action - дія, яку потрібно здійснити при переповненні диска (цей параметр може приймати ті ж значення, що і space_left_admin).

1.3.3 **Створення правил**

Для додавання і налаштування правил використовується команда auditctl. Ось список її опцій:

-l - вивести список наявних правил;

-a - додати нове правило;

-d - видалити правило зі списку;

-D - видалити всі наявні правила.

Щоб створити нове правило, потрібно виконати команду виду:

$ auditctl -a <список>, <дію> -S <ім'я системного виклику> -F <фільтри>

Спочатку після опції -a вказується список, в який потрібно додати правило. Всього існує 5 таких списків:

- task - події, пов'язані зі створенням нових процесів;
- entry - події, які мають місце при вході в системний виклик;
- exit - події, які мають місце при виході з системного виклику;

- user - події, що використовують параметри робочих просторів;
- exclude - використовується для виключення подій.

Потім вказується, що потрібно робити після настання події. Тут можливі два варіанти: always (події будуть записуватися в журнал) і never (НЕ будуть).

Після опції -S йде ім'я системного виклику, при якому подія потрібно перехопити (open, close і т.п.).

Після опції -F вказуються додаткові параметри фільтрації. Наприклад, якщо нам потрібно вести аудит звернень до файлів з каталогу / etc, правило буде виглядати так:

$ auditctl -a exit, always -S open -F path = / etc /

Можна встановити і додатковий фільтр:

$ auditctl -a exit, always -S open -F path = / etc / -F perm = aw

Абревіатура aw означає наступне: a - зміна атрибута (attribute change), w - запис (write). Формулювання perm = aw вказує, що для директорії / etc потрібно відстежувати всі факти зміни атрибутів (a - attribute change) і w (w - write).

Під час налаштування стеження за окремими файлами можна опустити опцію -S, наприклад:

$ auditctl -a exit, always -F path = / etc / -F perm = aw

1.3.4 **файли правил**

Правила можна не тільки задавати через командний рядок, а й прописувати в файлі etc / audit / audit.rules.

Починається цей файл з так званих метаправо, в яких задаються загальні настройки журналирования:

Видаляємо всі раніше створені правила

-D

Задаємо кількість буферів, в яких будуть зберігатися повідомлення

-b 320

Вказуємо, що робити в критичній ситуації (наприклад, при переповненні буферів): 0 - нічого не робити; 1 - відправляти повідомлення в dmesg, 2 - відправляти ядро в паніку

-f 1

Далі слідують призначені для користувача правила. Їх синтаксис гранично простий: досить просто перерахувати відповідні опції команди auditctl. Розглянемо приклад типового конфігураційного файлу:

Відстежувати системні виклики unlink () і rmdir ()

-a exit, always -S unlink -S rmdir

Відстежувати системні виклики open () від користувача з UID 1001

-a exit, always -S open -F loginuid = 1001

Відстежувати доступ до файлів паролів і груп і спроби їх зміни:

-w / etc / group -p wa

-w / etc / passwd -p wa

-w / etc / shadow -p wa

-w / etc / sudoers -p wa

Відстежувати доступ до наступної директорії:

-w / etc / my_directory -p r

Закрити доступ до конфігураційного файлу для запобігання змін

-e-e 2

Зміни конфігурації вступлять в силу після перезапуску демона auditd:

$ sudo service auditd restart

1.3.5 **Аналіз журнальних файлів: утиліта aureport**

Все журнальні файли зберігаються в директорії / var / log / audit в машиночитаемом форматі. Їх можна зробити человекопонятний с допомогою утиліти aureport.

Якщо ввести команду aureport без аргументів, ми побачимо загальну системну статистику (кількість користувачів системи, загальна кількість системних викликів, число відкритих терміналів і т.п.):

$ sudo aureport

Summary Report

==========

Range of time in logs: 07/31/2015 14:04:23.870 - 08/04/2015 09:37:13.200

Selected time for report: 07/31/2015 14:04:23 - 08/04/2015 09:37:13.200

Number of changes in configuration: 0

Number of changes to accounts, groups, or roles: 3

Number of logins: 0

Number of failed logins: 0

Number of authentications: 0

Number of failed authentications: 61205

Number of users: 2

Number of terminals: 5

Number of host names: 73

Number of executables: 6

Number of files: 0

Number of AVC's: 0

Number of MAC events: 0

Number of failed syscalls: 0

Number of anomaly events: 0

Number of responses to anomaly events: 0

Number of crypto events: 0

Number of keys: 0

Number of process IDs: 17858

Number of events: 61870

Вона не має особливої практичної цінності. Набагато більший інтерес представляють спеціалізовані звіти. Ось так, наприклад, можна переглянути інформацію про всі системні виклики:

$ sudo aureport -s

Syscall Report

==========

date time syscall pid comm auid event

==========

1. 08/03/2015 15:45:03 313 10285 modprobe -1 52501

2. 08/03/2015 15:45:03 313 10290 modprobe -1 52502

3. 08/03/2015 15:45:03 54 10296 iptables -1 52503

4. 08/03/2015 15:45:03 54 10302 iptables -1 52504

5. 08/03/2015 15:45:03 54 10305 iptables -1 52505

6. 08/03/2015 15:45:03 54 10313 iptables -1 52506

7. 08/03/2015 15:45:03 54 10325 iptables -1 52507

8. 08/03/2015 15:45:03 54 10329 iptables -1 52508

9. 08/03/2015 15:45:03 54 10343 iptables -1 52509

10.08/03/2015 15:45:03 54 10345 iptables -1 52510

11.08/03/2015 15:45:03 54 10349 iptables -1 52511

Скориставшись опцією -au (або --auth), можна переглянути інформацію про всі спроби входу в систему:

$ sudo aureport -au

Authentication Report

==========

date time acct host term exe success event

==========

1. 08/31/2015 18:00:19 ubnt static-166-6-249-80.stalowa.pilicka.pl ssh /usr/sbin/sshd no 333

2. 08/31/2015 18:01:38 root 59.63.188.31 ssh /usr/sbin/sshd no 334

3. 08/31/2015 18:01:41 root 59.63.188.31 ssh /usr/sbin/sshd no 335

4. 08/31/2015 18:01:45 root 59.63.188.31 ssh /usr/sbin/sshd no 336

5. 08/31/2015 18:01:53 root 59.63.188.31 ssh /usr/sbin/sshd no 337

6. 08/31/2015 18:01:57 root 59.63.188.31 ssh /usr/sbin/sshd no 338

7. 08/31/2015 18:01:59 root 59.63.188.31 ssh /usr/sbin/sshd no 339

У aureport підтримується фільтрація по даті і часу:

$ Sudo aureport -s --start 07/31/15 12:00 --end 07/31/15 13:00

Можна вказувати як конкретні час і дату, так і спеціальні Человекопонятние конструкції:

now - поточний момент;

yesterday - вчорашнє добу;

recent - 10 хвилин назад;

this-week (або this-month, this-year) - поточний тиждень (місяць, рік).

За допомогою aureport можна переглянути інформацію про дії будь-якого користувача системи. Для цього потрібно спочатку дізнатися id цього користувача:

$ Id user

uid = 1000 (user) gid = 1000 (andrei) groups = 1000 (andrei), 27 (sudo)

і потім виконати наступну команду:

$ Sudo ausearch -ui 1000 --interpret

1.3.6 **Ausearch: пошук і аналіз подій**

Для перегляду детальної інформації про подію використовується утиліта ausearch:

$ Sudo ausearch -a <номер події>

Висновок наведеної вище команди виглядає так:

type = SYSCALL msg = audit (1364481363.243: 24287): arch = c000003e syscall = 2 success = no exit = -13 a0 = 7fffd19c5592 a1 = 0 a2 = 7fffd19c4b50 a3 = a items = 1 ppid = 2686 pid = 3538 auid = 500 uid = 500 gid = 500 euid = 500 suid =

500 fsuid = 500 egid = 500 sgid = 500 fsgid = 500 tty = pts0 ses = 1 comm = "cat" exe = "/ bin / cat" subj = unconfined_u: unconfined_r: unconfined_t: s0-s0: c0.c1023 key = "sshd_config"

Розглянемо його структуру більш докладно. В поле type вказується тип запису; type = syscall означає, що запис був зроблений після виконання системного виклику. В поле msg вказано час події в форматі Unix Timestamp і його унікальний ідентифікаційний номер.

В поле arch міститься інформація про використовувану архітектурі системи (c000003e означає x86_84), представлена в закодованому шестнадцатеричном форматі. Щоб вона виводилася в человекочітаемом вигляді, можна скористатися опцією -i або --interpret.

В поле syscall зазначений тип системного виклику - в нашому випадку це 2, тобто виклик open. Параметр success повідомляє, чи був виклик оброблений успішно чи ні. У нашому прикладі виклик був оброблений невдало (success = no).

Для кожного виклику в звіті також перераховуються індивідуальні параметри; більш детально про них можна почитати в офіційному керівництві. Вивести на консоль інформацію про будь-якому параметрі в человекочітаемой формі можна отримати за допомогою згаданої вище опції -i або --interpret, наприклад:

$ Sudo ausearch --interpet --exit -13

Опція -sc дозволяє включати в список події, пов'язані з вказаною системному виклику, наприклад:

$ Sudo ausearch -sc ptrace

Опція -ui служить для пошуку подій за ідентифікатором користувача:

$ Ausearch -ui 33

Пошук по іменах демонів здійснюється за допомогою опції -tm:

$ Ausearch -x -tm cron

Для пошуку потрібних подій можна також використовувати ключі, наприклад:

$ Sudo auditctl -k root-actions

Наведена команда виведе список всіх дій, скоєних від імені root-користувача.

Підтримується також фільтрація по даті і часу, аналогічна тій, що була описана вище.

Вивести список подій, що завершилися невдало, можна за допомогою опції --failed.

1.3.7 **Аналіз процесів за допомогою утиліти autrace**

У деяких випадках буває корисним отримати інформацію про події, пов'язані з одним конкретним процесом. Для цієї мети можна скористатися утилітою autrace. Припустимо, нам потрібно відстежити процес date і дізнатися, які системні виклики і файли він використовує. Виконаємо команду:

$ Sudo autrace / bin / date

На консолі з'явиться наступний текст:

Waiting to execute: / bin / date

Mon Aug 31 17:06:32 MSK 2015

Cleaning up ...

Trace complete. You can locate the records with 'ausearch -i -p 29234'

Звернемо увагу на останній рядок виводу: в ній зазначена команда, за допомогою якої можна отримати більш детальну інформацію. Виконаємо цю команду і передамо висновок утиліті aureport, яка перетворює його в человекочітаемий формат:

$ Sudo ausearch -p 29215 --raw | aureport -f -i

В результаті ми отримаємо ось такий звіт:

File Report

==========

Date time file syscall success exe auid event

==========

1. 08/31/2015 16:52:16 / bin / date execve yes / bin / date root 25

2. 08/31/2015 16:52:16 /etc/ld.so.nohwcap access no / bin / date root 27

3. 08/31/2015 16:52:16 /etc/ld.so.preload access no / bin / date root 29

4. 08/31/2015 16:52:16 /etc/ld.so.cache open yes / bin / date root 30

5. 08/31/2015 16:52:16 /etc/ld.so.nohwcap access no / bin / date root 34

6. 08/31/2015 16:52:16 /lib/x86_64-linux-gnu/libc.so.6 open yes / bin / date root 35

7. 08/31/2015 16:52:16 / usr / lib / locale / locale-archive open yes / bin / date root 52

8. 08/31/2015 16:52:16 / etc / localtime open yes / bin / date root 56

1.3.8 **Централізоване зберігання логів**

Для відправки логів підсистеми аудиту в централізоване сховище використовується плагін audisp-remote. Він входить в пакет audisp-plugins, який потрібно встановлювати додатково:

$ Sudo apt-get install audisp-plugins

Файли всіх плагінів зберігаються в директорії /etc/audisp/plugins.d.

Налаштування віддаленого логгірованія прописуються в файлі конфігурації /etc/audisp/plugins.d/audisp-remote.conf. За замовчуванням цей файл виглядає так:

active = no

direction = out7

path = / sbin / audisp-remote

type = always

#args =

format = string

Щоб активувати відправку логів в віддалене сховище, замінимо значення параметра active на yes. Потім відкриємо файл etc / audisp / audisp-remote.conf і в якості значення параметра remote_server вкажемо буквений або IP-адреса сервера, на якому будуть зберігатися логи.

Щоб приймати логи з віддалених хостів і зберігати їх на сервері, в файлі /etc/audit/auditd.conf потрібно прописати наступні параметри:

tcp_listen_port = 60

tcp_listen_queue = 5

tcp_max_per_addr = 1

tcp_client_ports = 1024-65535 #optional

tcp_client_max_idle = 0

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділу було розглянуто засоби аудиту безпеки в WINDOWS 10, та її альтернатива у UNIX системах

Як бачимо Windows 10 має потужний засіб аудиту безпеки, велику кількість конфігурацій, та зручний інтерфейс. Але є недоліки:

- Неможливо конфігурувати аудит системних подій(тільки включити моніторинг всіх подій або відключити).
- Важка конфігурація для звичайного користувача
- Є підтримка аудиту тільки в повній версії Windows

Тому розглянемо альтернативні засоби, наприклад у системі Linux, які надають гнучке конфігурування саме аудиту системних викликів. Також їх перевага є у швидкості отримання разультату. Але є багато недоліків:

- Немає графічного інтерфейсу
- Додання правил і вивід результатів це дві різні програми.
- Не зручне конфігурування

Розглянувши недоліки та сформувавши вимоги, потрібно розробити програмне забезпечення, яке надасть можливість зручно конфігурувати аудит системних подій та перевіряти їх результат.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ

2.1 Опис предметної області

Аудит безпеки обчислювального вузла – середовище, що надає можливість проводити контролю та перевірки деяких подій за допомогою певних маніпуляцій.

Аудит безпеки повинен підтримувати основні дії:

1) Моніторинг різних подій
2) Додання певних правил аудиту
3) Маніпулювання над правилами аудиту

В контексті даної роботи проводиться аудит системних викликів у персональному комп'ютері певного користувача або групи користувачів. Системні виклики – це процеси які викликанні ядром операційної системи.

Основний принцип роботи аудиту системних подій - для кожної події, ставиться тригер, який вмикається і вимикається за допомогою існуючих правил у системі аудиту, якщо тригер ввімкнений то кожний виклик цього процесу оброблюється аудитом.

В правилах потрібно указувати назву системного виклику, її подію та дію, а також необовязково деякі фільтри для цього виклику (наприклад обмежений шлях виклику, певний час тощо.)

Спосіб обробки цих події - виводити інформацію о них у певний лог-файл. Аналізуючи цей лог, можна проаналізувати цю подію.

2.2 **Функції та вимоги до програмного забезпечення**

Основними функціями, що має виконувати додаток є:

- Можливість перевірити наявні правила системи аудиту.
- Можливість редагувати список правил.
- Аналіз подій які задані у правилах

До вимог слід віднести:

- Зрозумілість у використанні.
- Простий дизайн.
- Одночасне виконання моніторингу і додавання правил.

2.3 **Опис функціоналу програми**

Програмне забезпечення повинно надавати користувачеві можливість виконувати наступні дії:

1) Додання правила аудиту за назвою системної події .
2) Видалення правила аудиту.
3) Перевірка наявності системної події
4) Перевірка всіх наявних правил у системі аудиту
5) Вивести інформацію про обраний системний виклик

2.4 **Прецеденти**

Після визначення функцій, які виконуватиме програма, деталізуємо сценарії використання та побудуємо діаграму варіантів використання. На рис. 2.1. зображена діаграма прецедентів високого рівня. На ній зображені можливі дій користувачі, абстраговані від деталей. Детальніша ієрархія прецедентів зображена на рис. 2.2 – 2.4.

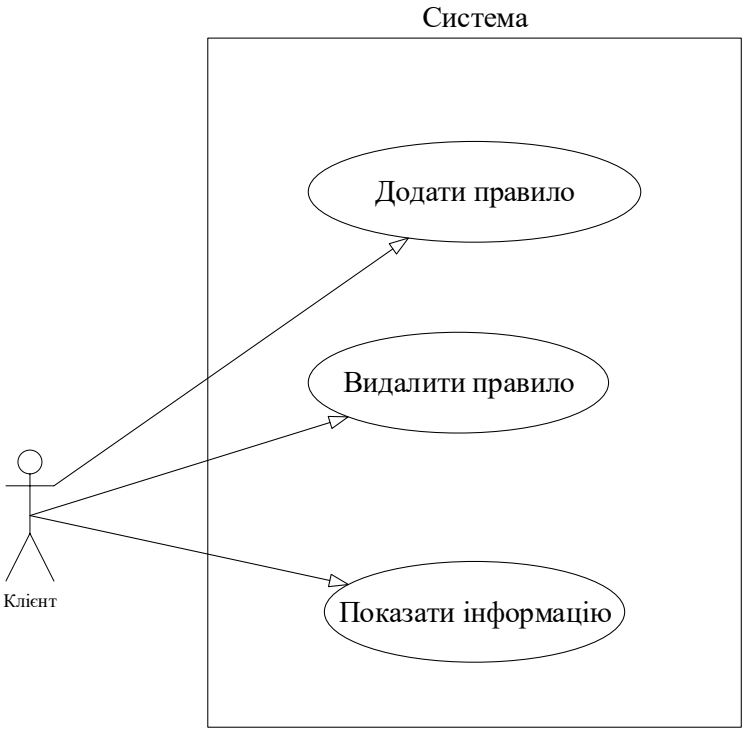

Рисунок 2.1. Діаграма прецедентів додатку

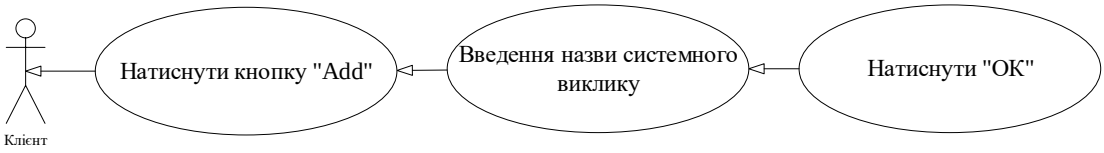

Рисунок 2.2. Ієрархія прецеденту «Додати правило»

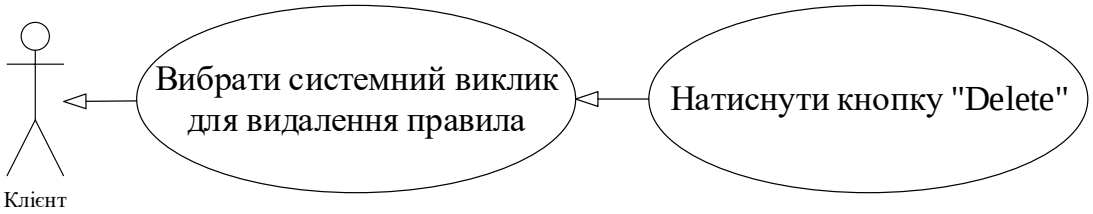

Рисунок 2.3. Ієрархія прецеденту «Видалити правило»

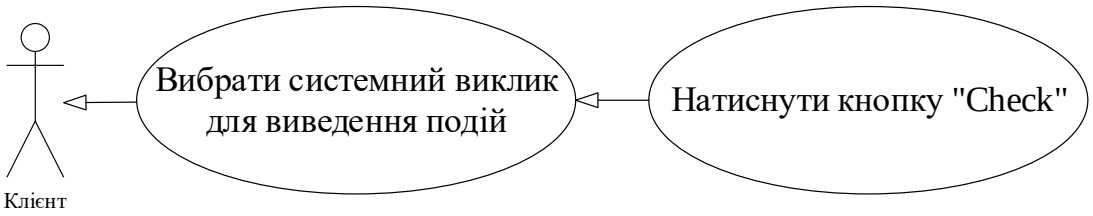

Рисунок 2.4. Ієрархія прецеденту «Показати інформацію»

Розглянемо детально прецеденти:

1. Додати правило;
2. Видалити правило;
3. Показати інформацію

2.4.1 **Додати правило**

Даний прецедент є одним із основних у логіці програми; він використовується для додавання правил системних викликів. Сценарій може мати декілька варіантів розвитку:

1. Користувач ввів правильний системний виклик, - опис цього сценарію наведено у табл. 2.1;
2. Користувач ввів неіснуючий системний виклик, - опис сценарію у табл. 2.2.
3. Користувач ввів вже доданий у правило системний виклик, - опис сценарію у табл. 2.3.

Таблиця 2.1. Опис прецеденту « Правильний системний виклик »

Користувач	Система
1. Користувач натиснув кнопку «Add»	2. Система відображає поле вводу системного виклику
3. Користувач вводить назву системного виклику	4. Система перевіряє корректність вводу та додає нове правило.

Таблиця 2.2. Опис прецеденту «Неіснуючий системний виклик»

Користувач	Система
1. Користувач натиснув кнопку «Add»	2. Система відображає поле вводу системного виклику
3. Користувач вводить назву неіснуючого системного виклику	4. Система перевіряє корректність вводу та виводить помилку

Таблиця 2.3. Опис прецеденту «Вже доданий системний виклик»

Користувач	Система
1. Користувач натиснув кнопку «Add»	2. Система відображає поле вводу системного виклику
3. Користувач вводить назву вже доданого системного виклику	4. Система перевіряє корректність вводу та виводить помилку

Альтернатива: користувач натискає кнопку «назад» до завершення сценарію, тоді на екрані відображається початкове. Наприкінці кожного сценарію система оновлює і показує список існуючих правил.

2.4.2 **Видалити правило**

Даний прецедент є одним із основних у логіці програми; він використовується для видалення правил системних викликів. Сценарій може мати декілька варіантів розвитку:

1. Користувач вибрав системний виклик із списку і видалив його- опис сценарію у табл. 2.4;
2. Користувач вибрав системний виклик із списку і відмовився видаляти його, - опис сценарію у табл. 2.5.

Таблиця 2.4. Опис прецеденту « Підтверджене видалення »

Користувач	Система
1. Користувач вибрав системний виклик із списку правил і натисну кнопку «Delete»	2. Система відображає вікно підтвердження видалення правила
3. Користувач підтверджує видалення	4. Система видаляє правило із списку, перестає моніторити цей системний виклик і показує оновлений список правил

Таблиця 2.5. Опис прецеденту « Відмінене видалення »

Користувач	Система
1. Користувач вибрав системний виклик із списку правил і натисну кнопку «Delete»	2. Система відображає вікно підтвердження видалення правила
3. Користувач відміняє видалення	4. Система продовжує моніторити цей системний виклик.

Альтернатива: користувач не може видалити правило, якщо не додано ні одного правила аудиту системного виклику, користувач просто не має варіантів вибору системних викликів для видалення.

2.4.3 **Перевірити правило**

Даний прецедент є одним із основних у логіці програми; він використовується для перевірки правила системного виклику. Виводиться інформація про усі події системного виклику починаючи з запуску системи.

Таблиця 2.6. Опис прецеденту « Перевірити правило »

Користувач	Система
1. Користувач вибрав системний виклик із списку правил і натисну кнопку «Check»	2. Система обновлює текстову зону яка відповідає виводу інформації про події системного виклику до вибраного правила

Альтернатива: користувач не може перевірити правило, якщо не додано ні одного правила аудиту системного виклику, користувач просто не має варіантів вибору системних викликів для перевірки.

2.5 **Проектування графічного інтерфейсу**

Графічний інтерфейс програми має включати в себе такі компоненти:

1. Компонент виводу списку правил;
2. Компонент виводу записів подій системних викликів;
3. Кнопку для додання правил;
4. Кнопку для видалення правил;
5. Кнопку для перевірки правил.

На рисунку 2.5 зображено схематичну структуру інтерфейсу , а на рисунку 2.6 макет вигляду цієї програми

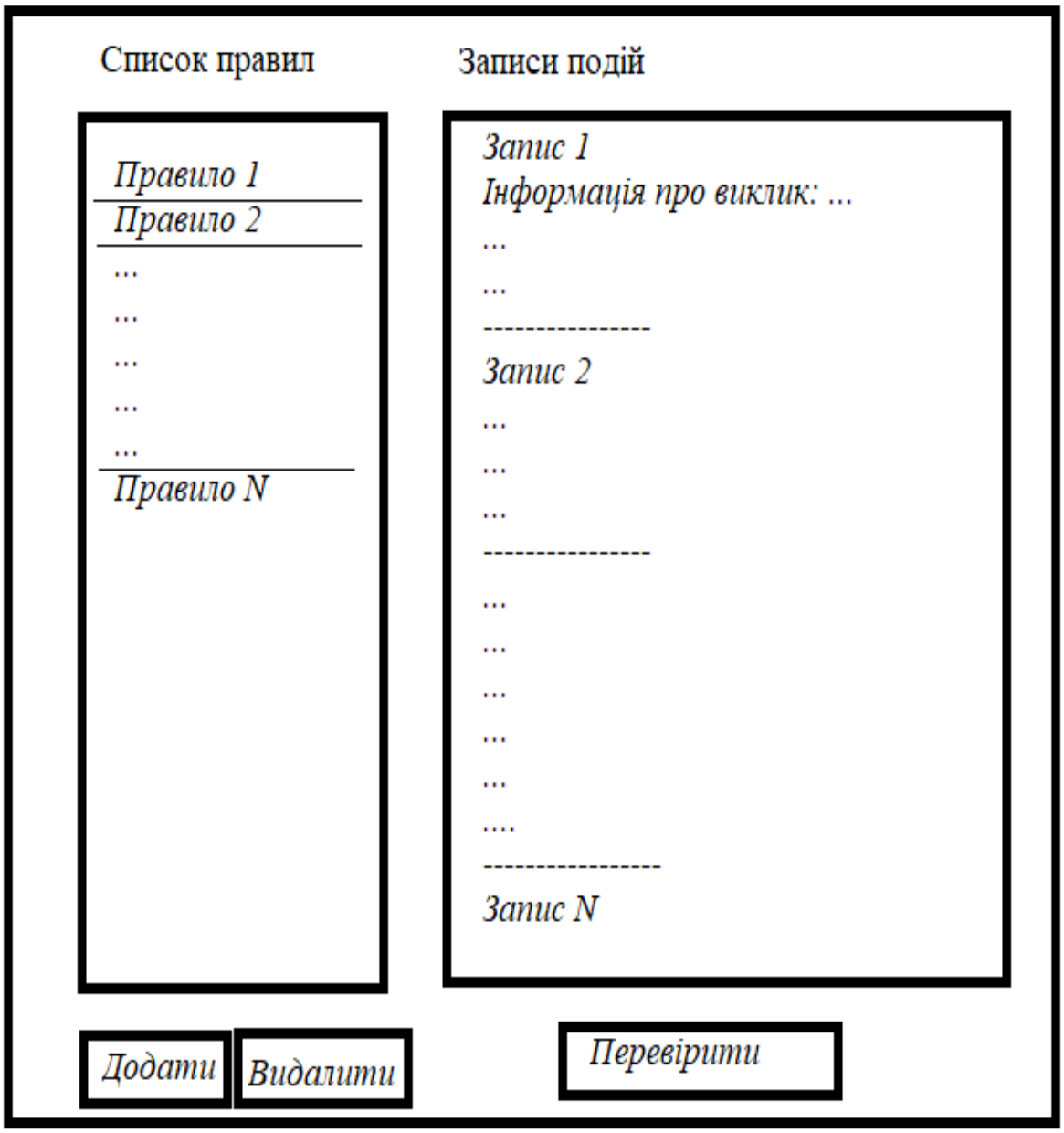

Рисунок 2.5. Схематична структура інтерфейсу

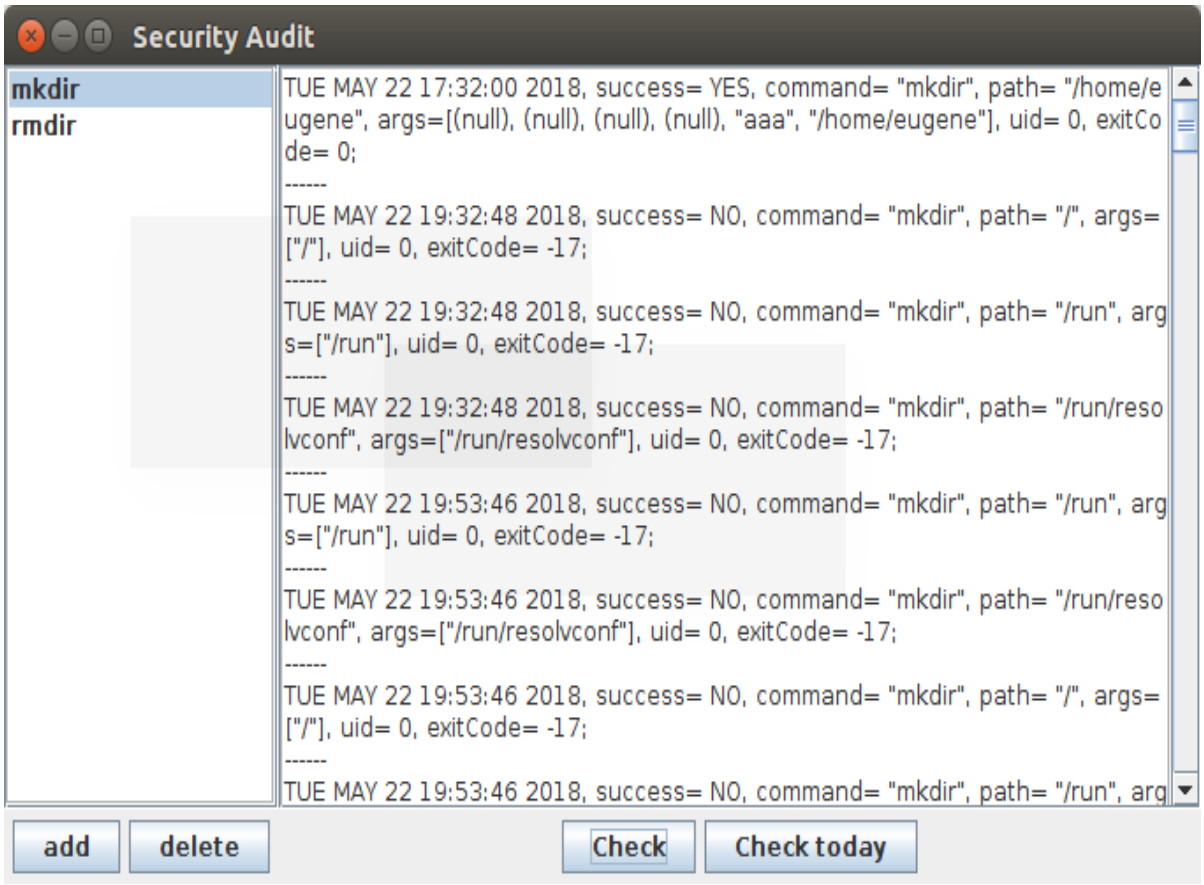

Рисунок 2.6. Макет вигляду програми на екрані

ВИСНОВОК ДО РОЗДІЛУ 2

У цьому розділі було розроблено загальний опис програмного забеспечення, сформовано основні вимоги до функціоналу. Окрім цього, було окреслено можливі сценарії використання та визначені можливі помилки, що можуть виникати при їх виконанні, а також методи їх запобігання та обходу.

За результатами проведених вище досліджень було побудовано діаграми варіантів використання (англ. Use-case diagram) для основних прецедентів. Детально було проаналізовано та побудовано описи у вигляді таблиць для наступних прецедентів: додавання правила, видаленння правила та перевірка правила, оскільки вони можуть мати різні сценарії розвитку в залежності від вибору користувача. Врахування таких можливостей на етапі проектування збереже програиу від неочікуваних помилок.

Також розділ покриває тему графічного інтерфейсу додатку. Було побудовано схематичну структуру для головних дій у програмі, а також макет вигляду програми.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ

3.1 Вибір технології та їх обґрунтування

3.1.1 Вибір операційної системи для додатку

Виходячи з вимог, поставлених програмі, необхідно досягнути максимальної доступності для широкого кола користувачів, також передбачена можливість використання системних команд. Отже для виконання поставлених вимог найкраще потрібно обрати операційну систему для персональних комп’ютерів. Розглянемо декілька можливих операційних систем

Windows — узагальнююча назва операційних систем для ЕОМ, розроблених корпорацією Microsoft.

Перші версії були не повноцінними операційними системами, а лише оболонками до ОС MS-DOS. На 2014 рік, за даними сайтів NetApplications та GoStats, Microsoft Windows встановлена більш як на 90% персональних комп'ютерів світу.

Пакет Microsoft Windows включає стандартні застосунки, такі як браузер (Internet Explorer), поштовий клієнт (Outlook Express), програвач (Windows Media Player). За допомогою технологій COM і OLE їхні компоненти можуть вбудовуватися в інші застосунки, у тому числі й сторонніх виробників. Ці продукти позиціонуються як безплатні і можуть бути вільно викачані з офіційного сайту Microsoft, проте для установки деяких з них необхідно мати робочу ліцензію Windows. Запуск цих програм під іншими операційними системами можливий за допомогою емуляторів середовища Windows. Існують також версії деяких з них, спеціально розроблені для інших операційних систем. Ці версії поступаються оригінальним версіям набором функцій і можливостей, а також частотою оновлення.

Навколо факту включення таких стандартних продуктів в ОС Windows розгортається багато суперечок і дискусій, оскільки це створює серйозні перепони для розповсюдження конкуруючих продуктів. Так, часто оспорюють лідерство і ставлять під сумнів якість браузера Internet Explorer, пояснюючи його популярність входженням в пакет Windows і поганою обізнаністю користувачів про наявність альтернатив. А в березні 2004 року Європейська Комісія оштрафувала Майкрософт і зобов'язала створити для продажу в Європі версію Windows без Windows Media Player. Сама компанія Майкрософт продовжує відстоювати своє право включати свої продукти в пакет Windows і розробляє нові інтегровані продукти

Нині Windows встановлена на більшості персональних комп'ютерів і робочих станцій. Багато користувачів зробили вибір на користь Microsoft Windows, але дуже велика кількість зовсім не знайомі з альтернативами, такими як Mac OS X x86, GNU/Linux, BSD і іншими системами. В Україні до початку 2000-х років майже всі персональні комп'ютери продавалися з передвстановленою системою Windows. Боротьба з розповсюдженням нелегальних копій програмних продуктів привела до появи інтересу до вільних операційних систем. Так, наприклад, стало можливим придбати персональний комп'ютер з передвстановленою безкоштовною операційною системою GNU/Linux, FreeBSD і іншими

Таблиця 3.1 Розповсюдженість версій MS Windows

Джерело	NetApplications	NetApplications	NetApplications	NetApplications	GoStats
Дата	січень 2010	лютий 2010	березень 2010	квітень 2010	травень 2011
Усі версії	91,70 %	93,16 %	91,63 %	91,5 %	96,93 %
Windows XP	66,15 %	65,49 %	64,46 %	63,41 %	59,78 %
Windows Vista	17,47 %	16,51 %	16,01 %	15,60 %	15,77 %
Windows 2000	0,57 %	0,56 %	0,56 %	0,50 %	0,53 %
Windows 7	7,57 %	8,92 %	10,23 %	11,68 %	32,3 %
Windows Server 2003	-	-	-	-	0,17 %
Windows 98	0,08 %	0,07 %	0,07 %	0,07 %	
Windows ME	0,05 %	0,04 %	0,04 %	0,03 %	
Windows NT	0,11 %	0,52 %	0,21 %	0,17 %	
Windows CE	0,06 %	0,05 %	0,05 %	0,04 %	

Таблиця 3.2 Життєві цикли версій MS Windows

Клієнтська операційна система	Дата загальної доступності	Кінець роздрібних продажів	Кінець продажів ПК зі встановленою ОС
Windows XP	31 грудня 2001	30 червня 2008	22 жовтня 2010
Windows Vista	30 січня 2007	22 жовтня 2010	22 жовтня 2011
Windows 7	22 жовтня 2009	31 жовтня 2013	31 жовтня 2014
Windows 8	26 жовтня 2012	31 жовтня 2014	
Windows 8.1	18 жовтня 2013		

Остання версія Windows 10 — операційна система від компанії Microsoft для персональних комп'ютерів, ноутбуків, планшетів, лептопів-трансформерів і смартфонів. У компанії цю версію операційної системи називають останньою, позаяк надалі вона надаватиметься за моделлю «програмне забезпечення як послуга» і періодично оновлюватиметься.

Microsoft представила попередню версію Windows 10 у Сан-Франциско 30 вересня 2014 року. Реліз Windows 10 відбувся влітку 2015 року, а саме 29 липня в 190 країнах і 111-ма мовами. Протягом першого року після виходу системи користувачі мали змогу безкоштовно оновитися до Windows 10 на будь-якому пристрої під керуванням офіційних версій Windows 7, Windows 8.1 і Windows Phone 8.1, що відповідають певним вимогам.

Яку версію Windows вибрати?

Операційна версія Windows є незаперечним лідером серед всіх операційних систем, більш того, Windows 7 є найбільш популярною системою серед інших продуктів Microsoft. Склалася така ситуація в основному завдяки тому, що саме на Windows випускається найбільшу кількість ігор і якісного програмного забезпечення.

Популярність Windows в країнах СНД пояснюється переважно тим, що велика частина людей користується піратськими версіями.

Windows XP

Рисунок 3.1 Вигляд заставки WindowsXP

Незважаючи на моральне старіння, даною системою, як і раніше користується велика кількість людей. Є деякі, вузькоспеціалізовані програми, які можна запустити виключно на даній операційній системі. На жаль, сучасні ігри та програми в більшості своїй, не запускаються на Windows XP, крім того, в цій системі просто не вистачить оперативної пам'яті, адже для підтримки більше 3 Гб ОЗУ, необхідна 64 бітна версія ОС, яка з'явилася лише з виходом Windows 7

Windows 7

Рисунок 3.2 Вигляд заставки Windows 7

Незважаючи на те, що перед Windows 7 була випущена Vista, вона не отримала такої популярності як Windows 7. Дана система має максимальну стабільністю, зручним інтерфейсом, практично всіма встановленими драйверами і багатьма іншими цікавими можливостями. Для людей, що не володіють процесорами з 64 бітної архітектурою, Microsoft випустила і 32 бітну Windows 7. Останнім часом, дана система почала застарівати. Багато ігор, вимагають, по крайней мере, пакет оновлень SP1

Windows 8

Рисунок 3.3 Вигляд заставки Windows 8

Практично нічим не відрізняється від попередньої ОС, більш того, багато користувачів відзначають, що Windows 8 працює значно швидше, навіть більш пізніх редакцій системи. Незважаючи на це, Windows 8 не отримала бажаного поширення, причиною всьому, стало не полюбився багатьма плитковий інтерфейс і відсутність клавіші «Пуск». Пізніше, Microsoft прислухалася до вимог користувачів, в пакеті оновлень SP1 додали звичний інтерфейс. По суті, Windows 8 не заслужила настільки гнівних відгуків, адже дана система, перш за все, призначалася для сенсорних екранів, де плитковий інтерфейс був би куди зручніше клавіші «Пуск».

Windows 10

Рисунок 3.4 Вигляд заставки Windows 10

Остання на даний момент операційна система від Microsoft, яка ввібрала в себе, всі напрацювання попередніх ОС. На жаль, Windows 10 має ряд недоліків, які перекреслюють всі переваги. Адже по суті, дана система, не має революційних нововведень, зате, має цілий стос вбудованих функцій відстеження і неможливість відключити оновлення ОС.

Linux - По суті, Linux єдина система, що конкурує з Windows, хоч і значно відстає в популярності. Є ще Mac ОС, але говорити про неї немає сенсу, адже для використання даної системи, необхідно купувати спеціальний комп'ютер від компанії Apple.

Що стосується Linux, дану систему встановлюють, перш за все, в офісах і інших комерційних закладах, адже даний продукт має вільну ліцензію, і кожен користувач може абсолютно безкоштовно встановити її на свій комп'ютер.

На жаль, для любителів ігор, Linux абсолютно не придатний. У всьому ж іншому, мультимедійні можливості даної системи не поступаються Windows. Linux без праці може відтворювати відеофайли, музику, відкривати графічні

файли і працювати з текстом. Практично кожній програмі, на Linux є своя, безкоштовна альтернатива. Що без сумніву є перевагою.

Іншим, важливою перевагою є те, що Linux, практично невразлива для атак і вірусних програм. По суті, на дану ОС, навіть не встановлюють антивірусне ПЗ.

Як і Windows, Linux має поділ на 32 і 64 бітні системи, проте, на відміну від Windows, в Linux немає одного розробника. Так як дана система має відкритий вихідний код, кожен може самостійно працювати над програмним кодом системи.

Іншими словами, Debian, Ubuntu, Fedora, все це операційні системи на базі Linux і розробляються різними не пов'язаними між собою виробниками. Що стосується вибору найкращою системи, відповісти складно, адже крім усього іншого ОС на Linux мають свій графічний інтерфейс, який можна при бажанні замінити.

Найстарішими і поширеними є Debian, Ubuntu. Дані системи мають найбільш простий і зрозумілий інтерфейс.

Розглянемо найпопулярніший дистрибутив Linux – Ubuntu.

Рисунок 3.5 Вигляд робочого столу Ubuntu

Ubuntu — операційна система для робочих станцій, лептопів і серверів, найпопулярніший у світі дистрибутив Linux. Серед основних цілей Ubuntu — надання сучасного й водночас стабільного програмного забезпечення для пересічного користувача із сильним акцентом на простоту встановлення та користування.

Ubuntu надає користувачу мінімальний набір програм загального призначення: багатовіконне стільничне середовище, засоби для перегляду Інтернету, організації електронної пошти, офісні програми з можливістю читати і записувати файли у форматах, що використовуються в пакеті програм Microsoft Office, редактор зображень, програвач компакт-дисків тощо. Спеціалізоване програмне забезпечення, потрібне досвідченішим користувачам, можна отримати з відповідних репозиторіїв. Серверний варіант системи включає також засоби, потрібні для організації сервера баз даних, веб-сервера, сервера електронної пошти тощо.

Ubuntu побудований на основі Debian GNU/Linux — іншого популярного дистрибутиву Лінукс. Спонсорується Canonical Ltd., власником якої є бізнесмен із ПАР Марк Шаттлворт. Назва дистрибутиву походить від зулуської концепції «убунту[en]», яку можна висловити приблизно, як «людяність». Дистрибутив так названий для популяризації духу цієї філософії у світі програмного забезпечення. Ubuntu належить до вільного програмного забезпечення і може безкоштовно передаватись необмеженій кількості користувачів.

Офіційні підпроекти Lubuntu та Xubuntu використовують як основні стільничні середовища LXDE та Xfce відповідно замість Unity, що встановлюється за замовчуванням в дистрибутивах основного проекту. Ці дистрибутиви мають менші системні вимоги і підходять для використання на слабких та старих комп'ютерах.

До 6 лютого 2012-го ще одним офіційним підпроектом була Kubuntu із основною стільницею KDE. Canonical припинила її підтримку, пояснивши це відсутністю комерційного успіху проекту протягом 7 років.

Інший офіційний підпроект, Edubuntu, розробляється для шкіл і використання дітьми вдома.

Gobuntu є офіційним підпроектом, в якому використовується винятково вільне програмне забезпечення.

Ще одним офіційним підпроектом є JeOS (вимовляється «Джюс»), створений для використання на віртуальних машинах.

Нові версії Ubuntu виходять кожні шість місяців. Кожен реліз підтримується оновленнями програмного забезпечення протягом 18 місяців. Кожні два роки виходять LTS-випуски (англ. Long Term Support — довгострокова підтримка), які підтримуються оновленнями протягом п'яти років, як десктопні версії, так і серверні (до версії 12.04 LTS-релізи для десктопів підтримувалися протягом трьох років, для серверів — п'яти). LTS релізами були версії 6.04, 8.04, 10.04, 12.04, 14.04. Останній LTS випуск, Ubuntu 16.04.1 (Xenial Xerus), вийшов 21 липня 2016.

Ubuntu поставляється з найсвіжішою версією GNOME, але починаючи з версії 11.04 компанія Canonical вирішила розробити свою власну стільницю під назвою Unity. Програмне забезпечення підбирається таким чином, щоб інсталяційні пакунки та Live CD помістилися на одному компакт диску, водночас надавши користувачеві змогу створити для себе зручне робоче середовище. Live CD включає в себе версії популярних крос-платформених програм (Mozilla Firefox, Mozilla Thunderbird, Empathy, OpenOffice.org/LibreOffice та F-Spot), які дають змогу користувачеві ознайомитися із системою ще до встановлення на твердий диск.

Педро Корте-Реаль (Pedro Côrte-Real) провів дослідження складу базового сховища «main» дистрибутиву Ubuntu 11.04 за обсягом коду (у вибірці фігурувало число рядків коду). З великих проектів відзначені: Linux-ядро (9%, але з них 3% займають пов'язані з ядром зовнішні компоненти, такі як iptables і udev), інструменти GNU, такі як компілятор GCC і системна бібліотека Glibc (8%), X.Org (3%), Java (6%), Mozilla (6%), KDE (8%), GNOME (5%). Але домінують в дистрибутиві маленькі проекти, сукупна частка яких становить 56%.

Для розробки програми необхідно вибрати ОС яка має більший функціонал для роботи з системними викликами, тому доцільно вибрати ОС Ubuntu. Яка на відміну від Windows має можливість напряму працювати з ядром ОС, ставити тригери для кожної системної події. Також тільки у Windows pro є можливість виконувати всі оснастки для роботи за аудитом, але вона доступна не всім.

3.1.2 **Вибір мови програмування**

Для розробки програмного забезпечення було обрано один з найпопулярніших об’єктно орієнтованих мов програмування – Java.

Java- об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та

полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Java вплинула на розвиток J++, що розроблялась компанією «Microsoft». Роботу над J++ було зупинено через судовий позов «Sun Microsystems», оскільки ця мова програмування була модифікацією Java. Пізніше в новій платформі «Microsoft» .NET випустили J#, щоб полегшити міграцію програмістів J++ або Java на нову платформу. З часом нова мова програмування C# стала основною мовою платформи, перейнявши багато чого з Java. J# востаннє включався в версію Microsoft Visual Studio 2005. Мова сценаріїв JavaScript має схожу із Java назву і синтаксис, але не пов'язана із Java.

У створенні мови програмування Java було п'ять початкових цілей:

1) Синтаксис мови повинен бути «простим, об'єктно-орієнтовним та звичним».
2) Реалізація має бути «безвідмовною та безпечною».
3) Повинна зберегтися «незалежність від архітектури та переносність».
4) Висока продуктивність виконання
5) Мова має бути «інтерпретованою, багатонитевою, із динамічним зв'язуванням модулів».

Під «незалежністю від архітектури» мається на увазі те, що програма, написана на мові Java, працюватиме на будь-якій підтримуваній апаратній чи системній платформі без змін у початковому коді та перекомпіляції.

Цього можна досягти, компілюючи початковий Java код у байт-код, який є спрощеними машинними командами. Потім програму можна виконати на будь-якій платформі, що має встановлену віртуальну машину Java, яка інтерпретує байткод у код, пристосований до специфіки конкретної операційної системи і процесора. Зараз віртуальні машини Java існують для більшості процесорів і операційних систем.

Стандартні бібліотеки забезпечують загальний спосіб доступу до таких платформозалежних особливостей, як обробка графіки, багатопотоковість та роботу з мережами. У деяких версіях задля збільшення продуктивності JVM байт-код можна компілювати у машинний код до або під час виконання програми.

Основна перевага використання байт-коду — це портативність. Тим не менш, додаткові витрати на інтерпретацію означають, що інтерпретовані програми будуть майже завжди працювати повільніше, ніж скомпільовані у машинний код, і саме тому Java одержала репутацію «повільної»[джерело?] мови. Проте, цей розрив суттєво скоротився після введення декількох методів оптимізації у сучасних реалізаціях JVM.

Одним із таких методів є just-in-time компіляція (JIT, що перетворює байт-код Java у машинний під час першого запуску програми, а потім кешує його. У результаті така програма запускається і виконується швидше, ніж простий інтерпретований код, але ціною додаткових витрат на компіляцію під час виконання. Складніші віртуальні машини також використовують динамічну рекомпіляцію, яка полягає в тому, що віртуальна машина аналізує поведінку запущеної програми й вибірково рекомпілює та оптимізує певні її частини. З використанням динамічної рекомпіляції можна досягти більшого рівня оптимізації, ніж за статичної компіляції, оскільки динамічний компілятор може робити оптимізації на базі знань про довкілля періоду виконання та про завантажені класи. До того ж він може виявляти так звані гарячі точки (англ. hot

spots) — частини програми, найчастіше внутрішні цикли, які займають найбільше часу при виконанні. JIT-компіляція та динамічна рекомпіляція збільшує швидкість Java-програм, не втрачаючи при цьому портативності.

Існує ще одна технологія оптимізації байткоду, широко відома як статична компіляція, або компіляція ahead-of-time (AOT). Цей метод передбачає, як і традиційні компілятори, безпосередню компіляцію у машинний код. Це забезпечує хороші показники в порівнянні з інтерпретацією, але за рахунок втрати переносності: скомпільовану таким способом програму можна запустити тільки на одній, цільовій платформі.

Швидкість офіційної віртуальної машини Java значно покращилася з моменту випуску ранніх версій, до того ж, деякі випробування показали, що продуктивність JIT-компіляторів у порівнянні зі звичайними компіляторами у машинний код майже однакова. Проте ефективність компіляторів не завжди свідчить про швидкість виконання скомпільованого коду, тільки ретельне тестування може виявити справжню ефективність у даній системі.

На противагу C++, Java об'єктно-орієнтованіша. Всі дані і дії групуються в класи об'єктів. Виключенням з повної об'єктності (як скажімо в Smalltalk) є примітивні типи (int, float тощо). Це було свідомим рішенням проектувальників мови задля збільшення швидкості. Через це Java не вважається повністю об'єктно-орієнтовною мовою.

У Java всі об'єкти є похідними від головного об'єкта (він називається просто Object), з якого вони успадковують базову поведінку і властивості.

Хоча у C++ вперше стало доступне множинне успадкування, але у Java можливе тільки одинарне успадкування, завдяки чому виключається можливість конфліктів між членами класу (методи і змінні), які успадковуються від базових класів.

У намірах проектувальників Java мала замінити C++ — об'єктного наступника мови C.[8] Проектувальники почали з аналізу властивостей C++, які є причиною найбільшого числа помилок, щоби створити просту, безпечну і безвідмовну мову програмування.

В Java існує система винятків або ситуацій, коли програма зустрічається з неочікуваними труднощами, наприклад:

- операції над елементом масиву поза його межами або над порожнім елементом
- читання з недоступного каталогу або неправильної адреси URL
- ввід недопустимих даних користувачем

Одна з особливостей концепції віртуальної машини полягає в тому, що помилки (виключення) не призводять до повного краху системи. Крім того, існують інструменти, які «приєднуються» до середовища періоду виконання і кожен раз, коли сталося певне виключення, записують інформацію з пам'яті для відлагодження програми. Ці інструменти автоматизованої обробки виключень надають основну інформацію щодо виключень в програмах на Java.

Проте мову програмування Java не рекомендується використовувати в системах, збій в роботі яких може призвести до смерті, травм чи значних фізичних ушкоджень (наприклад, програмне забезпечення для керування атомними електростанціями, польотами, систем життєзабезпечення чи систем озброєння) через ненадійність програм, написаних на мові програмування Java.

Java використовує автоматичний збирач сміття для керування пам'яттю під час життєвого циклу об'єкта. Програміст вирішує, коли створювати об'єкти, а віртуальна машина відповідальна за звільнення пам'яті після того, як об'єкт стає непотрібним. Коли до певного об'єкта вже не залишається посилань, збирач сміття може автоматично прибирати його із пам'яті. Проте, витік пам'яті все ж може статися, якщо код, написаний програмістом, має посилання на вже непотрібні об'єкти, наприклад на об'єкти, що зберігаються у діючих контейнерах.

Збирання сміття дозволене у будь-який час. В ідеалі воно відбувається під час бездіяльності програми. Збірка сміття автоматично форсується при нестачі вільної пам'яті в купі для розміщення нового об'єкта, що може призводити до кілька секундного зависання. Тому існують реалізації віртуальної машини Java з прибиральником сміття, спеціально створеним для програмування систем реального часу.

Java не має підтримки вказівників у стилі C/C++. Це зроблено задля безпеки й надійності, аби дозволити збирачу сміття переміщувати вказівникові об'єкти.

3.1.3 **Мінімальні вимоги до запуску програми**

- Java 8 – тому що в програмі використовуються лямбда вирази які доступні з восьмої версії
- Версія ядра Лінукс 2.6 і вище - починаючи з версії 2.6, ядро Linux включає в себе підсистему, спеціально розроблену для проведення аудиту. Вона дозволяє вести стеження за виконанням системних викликів
- наявність демону auditd – утиліта яка займається журналюванням подій системних викликів. Установка в Ubuntu - sudo apt-get install auditd
- Термінал Xterm – Програма на джаві не може викликати команди операційної системи, через те що працює у своїй віртуальній машині JVM. Тому потрібно використовувати функції терміналів у програмі, я використовував Xterm
- Запуск програми від імені супер користувача(root)

3.1.4 **Вибір допоміжних бібліотек**

Крім самого демона auditd, разом з пакетом будуть встановлені деякі підсобні утиліти, що використовуються для управління системою аудиту та пошуку записів в журнальних файлах.

У програмі буде використано деякі з них:

- auditctl - утиліта, керуюча поведінкою системи аудиту. Дозволяє дізнатися поточний стан системи, додати або видалити правила. Будемо використовувати її для додання і видалення правил перевірки системних викликів.
- ausearch - команда, що дозволяє здійснювати пошук подій в журнальних файлах. Будемо використовувати її для відображення подій обраного системного виклику.

Для створення графічного інтерфейсу використаємо бібліотеку SWING:

Swing — інструментарій для створення графічного інтерфейсу користувача (GUI) мовою програмування Java. Це частина бібліотеки базових класів Java (JFC, Java Foundation Classes).

Swing розробляли для забезпечення функціонального набору програмних компонентів для створення графічного інтерфейсу користувача, ніж у ранішого інструментарію AWT. Компоненти Swing підтримують специфічні look-and-feel модулі, що динамічно підключаються. Завдяки ним можлива емуляція графічного інтерфейсу платформи (тобто до компоненту можна динамічно підключити інші, специфічні для даної операційної системи вигляд і поведінку). Основним недоліком таких компонентів є відносно повільна робота, хоча останнім часом це не вдалося підтвердити через зростання потужності персональних комп'ютерів. Позитивна сторона — універсальність інтерфейсу створених програм на всіх платформах.

Для взаємодії з оточенням, в якому запущена Java програма використаємо Клас Runtime.

3.2 **Основні рішення з реалізації додатку та його компонентів**

Розроблюваний додаток можна умовно поділити на наступні частини:

- Реалізація основних компонентів програми;
- Реалізація класу для роботи з системними утилітами і командами;
- Реалізація графічного інтерфейсу.

3.2.1 **Реалізація основних компонентів програми**

Програма містить такі компоненти:

- Правила
- Записи
- Контролер

Клас Rule –представляє собою правило перевірки подій системного виклику. Він включає в себе такі поля:

- systemCall – Поле яке відповідає за назву системної команди
- scroll – Поле яке відповідає списку подій при якому потрібно додати правило. У програмі використовуються події виходу (exit) з системного виклику щоб перевірити успішність або неуспішність виконання системного виклику
- action – дія при яка трапляється у відповідь на виникну подію. У програмі використовується запис у журнал подій(always)
- filters – Список додаткових параметрів фільтрації події. В програмі використовується вказання на архітектуру системи (arch=b64)

Методи класу надають доступ до полів

Клас також має конструктор який сам генерує правило, за назвою системного виклику

Клас Record представляє собою запис результату виникненої події по будь якому правилу, він має поля які несуть інформацію про цю подію:

- date – Поле яке відповідає даті та часу виникнення події системного виклику.
- success – Поле яке показує статус успіху завершення цієї події
- command – Поле яке відповідає назві команді при якої був викнан системний виклик
- path – Поле яке відповідає назві шляху з якого був надійшов цей системний виклик
- args – Список аргументів команди які були при системному виклику
- uid – id користувача який виконав команду
- exitCodr – Код помилки при невдалому завершені системного виклику, 0 – при успішному завершені.

Методи класу надають доступ до полів

Також є метод toString, який віддає інформацію у зручному для читанні вигляді.

Інтерфейс Controller – має виконувати основний функціонал програми. Він має такі методи:

- void add(Rule) – Додає правило до системи
- void remove(Rule) – Видаляє правило з системи
- List<Rule> checkRules() – Вивести всі наявні правила у системі
- List<Record> showResult(Rule) – Вивести список подій відповідного правила системного виклику.

Реалізацією цього інтерфейсу є клас ControllerImpl, який має викликати користувач при роботі з Аудитом системних подій, а для виконання основних методів інтерфейсу він використовує клас AuditRuntime

Клас також має допоміжні методи для парсингу результатів робот системних команд які викликаються в методах showResult і checkRules. Вони переводять результат у вигляді строки в компоненти програм (Правил і Записів).

3.2.2 Реалізація класу для роботи з системними утилітами і командами

Клас AuditRuntime має методи які визивають системні команди для різних методів класу ControllerImpl. Для написання цих методів потрібно розуміти як працюють системні команди і утиліти для роботи з Аудитом системних подій.

Розглянемо основні вимоги для програми:

1) Можливість додавати правило
2) Можливість видаляти правило
3) Можливість перевірити додані правила
4) Перевірити результат подій системного виклику.

Для перших трьох цілей використаємо утиліту auditctl

- Для додання правила у систему Аудиту використовується команда:

$ auditctl -a exit, always -S <Назва системного виклику> -F arch=b64

- Для видалення правила використовується команда:

$ auditctl -d exit, always -S <Назва системного виклику> -F arch=b64

- Для перевірки правил використовується команда

$ auditctl –l

Для перевірки результату подій системного виклику використовується команда ausearch, вона перевіряє журнал подій і виводить потрібні записи

$ ausearch -x < Назва системного виклику >

Отже в класі має бути 4 методи які парсять компоненти програми до потрібного вигляду, і викликають їх за допомогою класу Runtime:

- auditAdd(Rule) - приймає вже створене правило, парсить його до строки команди додання правила і викликає це правило у рантаймі, і якщо потрібно виводить помилку

- auditDelete(Rule) - приймає вже створене правило, парсить його до строки команди видалення і викликає це правило у рантаймі, і якщо потрібно виводить помилку
- auditShow() – у рантаймі викликає команду яка виводить всі існуючі правила
- ausearch(Rule) – приймає вже створене правило, парсить його до строки команди показу інформації, викликає це правило у рантаймі. Через можливо велику кількість записів у результаті виконання, створюється тимчасовий файл, потім з нього виводиться інформація до графічного інтерфейсу, а потім видаляється цей файл

3.2.3 **Реалізація графічного інтерфейсу**

Відповідно до спроектованого раніше дизайну у програмі реалізовано наступні компоненти:

1. Головний екран;

2. Список Правил;

3. Список Записів;

4. Необхідні кнопки для роботи через графічний інтерфейс;

Головний екран реалізовано за допомогою використання бібліотеки SWING (інструментарій для створення графічного інтерфейсу користувача (GUI) мовою програмування Java), а саме у програмі - клас JFrame. Який вмішує в себе панель розташування JPanel, яка керує коректним розташуванням всіх компонентів програми незалежно від розширення. Крім того, клас має всі необхідні методи для відображення внутрішнього фрагменту. Програма автоматично вибирає положення і розмір вікна відповідно до розширення монітору.

Основна панель може вміщати в себе інші компоненти, в тому числі і внутрішні панелі які вже самі розташовують свої компоненти.

Використані компоненти:

- JPanel – Панель для коректного розташування компонентів
- JList – Відображує список елементів
- JTextAre – Відображує текст
- JButton – При натисненні запускає відповідну подію

Для написання подій виклику, і визначення слухачів була використана бібліотека AWT.

Abstract Window Toolkit (AWT — абстрактний віконний інтерфейс) — це оригінальний пакет класів мови програмування Java, що слугує для створення графічного інтерфейсу користувача (GUI). AWT є частиною Java Foundation Classes (JFC) — стандартного API для реалізації графічного інтерфейсу для Java-програми. Пакет містить платформо-незалежні елементи графічного інтерфейсу, щоправда їхній вигляд залежить від конкретної системи.

ВИСНОВОК ДО РОЗДІЛУ 3

У цьому розділі було розглянуто аспекти вибору стеку технологій та бібліотек для реалізації програмного забезпечення, наведено обґрунтування вибору платформи для розробки програми, мови написання програми та множини пристроїв, які підтримуватимуться. Враховуючи вимоги до програмного продукту вибрано мову програмування та платформу, обрано допоміжні бібліотеки, які спростять процес розробки та дозволять побудувати гнучку архітектуру. Також розглянуті, можливі, варіанти реалізації функцій роботи Аудиту системних викликів.

Проведений опис основних рішень та підходів щодо реалізації проекту. Розроблено гнучку структуру компонентів, яка проектувалась із розрахунком на можливе розширення, підхід для використання команд у рантаймы та графічний інтерфейс користувача.

ВИСНОВКИ

Даний дипломний проект присвячений дослідженню Аудиту безпеки обчислювального вузла в Windows 10, порівняння з іншими системами Аудиту і знаходження кращого варіанту Аудиту безпеки. Метою проекту було створення програмного забезпечення для цього.

Під час виконання роботи було розглянуто актуальність використання Аудиту безпеки та існуючі рішення для Аудиту безпеки. Розглянувши різні готові рішення та визначивши їх переваги та недоліки, було вирішено що найважливішим є Аудит системних подій і було створено список мінімального функціоналу програми.

Проведено дослідження предметної області, визначено вимоги і завдання, які має виконувати програма. Розглянуто основні прецеденти та варіанти використання програми. Детально розглянуто прецеденти із можливими альтернативними варіантами. Уся інформація про прецеденти подана у структурованому вигляді – таблиць. Спроектовано концепцію дизайну програми. Створено вимоги до програми.

Виходячи із вимог, проведено детальний аналіз можливості використання технологій та платформ для реалізації програми. Обрано операційну систему для програми, визначено особливості розробки програм для цієї ОС. Також було обрано мову програмування: Java. Крім того проведено аналіз допоміжних бібліотек для написання програми, визначено та обґрунтовано доцільність їх використання.

В рамках програми було реалізовано заздалегідь визначений функціонал: Додання правила аудиту за назвою системної події, видалення правила аудиту, перевірка наявності системної події, перевірка всіх наявних правил у системі аудиту, вивести інформацію про обраний системний виклик.

Список використаної літератури

1. B. Phillips Android Programming: The Big Nerd Ranch. Third Edition / B. Phillips, C. Stewart, B. Hardy, K. N. Marsicano. - Indianopolis: Pearson Technology Group, 2017. – 650 с.
2. J. Bloch Effective Java (3rd Edition) / Joshua Bloch. – New York: Addison-Wesley, 2018. – 901 с.
3. H. Schildt The Complete Reference (10th Edition) / Herbert Schildt. - McGraw-Hill Education, 2018. – 1923 с.
4. Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java
5. Microsoft Windows [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Windows
6. Ubuntu [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Ubuntu
7. Windows 10 [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Windows_10
8. Аудит безпеки [Електронний ресурс] – Режим доступу до ресурсу: https://technet.microsoft.com/ru-ru/library/mt431897%28v=vs.85%29.aspx
9. Аудит системних подій в Linux [Електронний ресурс] – Режим доступу до ресурсу: https://xakep.ru/2011/03/30/54897/
10. Моніторинг системних викликів Linux [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/company/southbridge/blog/316902/