Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра кібербезпеки та комп'ютерних систем і мереж

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Удосконалення методу захисту інформаційних ресурсів від шкідливого програмного забезпечення

Назва теми

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

КРМКІ.015055.19.01.12 ПЗ

Виконав: студент 2 курсу, група КІ1м-19-1 _____ Казіміров В.О.
Підпис

Керівник доц., к. т. н, доцент кафедри КбКСМ _____ Орленко В.С.
Підпис

Нормоконтролер доц., к. т. н, доцент кафедри КбКСМ _____ Муляр І.В.
Підпис

До захисту допускаю:
Зав. кафедри КбКСМ, к.т.н., доц _____ Кльоц Ю.П.
Підпис

10 12 2020 р.

Хмельницький, 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Кібербезпеки та комп'ютерних систем і мереж

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітньо-професійна програма підготовки магістра

ЗАТВЕРДЖУЮ

Зав. кафедри Ю.П. Кльоц

"10" 12 2020 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Казімірову Володимиру Олеговичувичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Удосконалення методу захисту інформаційних ресурсів від шкідливого програмного забезпечення

1. Керівник проекту (роботи) к.т.н., доц. Орленко В.С.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом № 118 ректора університету додаток №23 від 01.09.2020

2. Строк подання студентом проєкту (роботи) на кафедру 1.12.2020

3. Вихідні дані до проекту (роботи) Проведення порівняльного аналізу підходів до вирішення задачі захисту від загрозливих програм на основі контролю доступу до інформаційних ресурсів. Оцінити актуальність загроз впровадження та запуску загрозливих програм для інформаційної системи. Розробити модель і методи захисту від загрозливих програм. Оцінити ефективність розроблених методів.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз сучасного стану захисту інформаційних ресурсів від загрозливих програм.

Моделі і методи захисту від загрозливих програм на основі контролю доступу до ресурсів.

Реалізація розроблених методів захисту інформаційних ресурсів.

Оцінка ефективності захисту інформаційних ресурсів.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Тема, мета магістерської роботи, об'єкт, предмет, задачі дослідження, наукова новизна, практична цінність, апробація роботи. Класифікація загрозливих програм. Модель оцінки ймовірності здійснення атаки. Модель захисту від загрозливих програм на основі контролю доступу до ресурсів. Метод захисту від загрозливих програм на основі контролю доступу до ресурсів. Функціональна схема механізму захисту. Метод розмежувальної політики на основі методів захисту від виконуваних загрозливих програм..Оцінка ефективності захисту інформаційних ресурсів Висновки.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання
Відповідальний за оформлення ДП	Муляр І.В.		

7. Дата видачі завдання «1» лютого 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи	Термін виконання етапів проекту (роботи)	Прим
1	Вибір напряму дослідження та узгодження тематики ДРМ з керівником	2.02.2020	
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	2.03.2020	
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	1.04.2020	
4	Робота над розділом 2 – розробка моделей і методів для вирішення поставленої задачі	1.05.2020	
5	Робота над науковою статтею	1.06.2020	
6	Робота над розділом 3 – розробка алгоритмів та технологій, їх аналіз	1.09.2020	
7	Робота над розділом 4 – проектування ПЗ для вирішення поставленої задачі	1.10.2020	
8	Узгодження отриманих; оформлення пояснювальної записки згідно вимог	1.11.2020	
9	Оформлення графічної частини	11.11.2020	
10	Попередній захист КРМ	15.11.2020	
11	Захист КРМ на засіданні ЕК	10.12.2020	

Студент _____ Казіміров В.І.
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Орленко В.С.
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: Удосконалення методу захисту інформаційних ресурсів від шкідливого програмного забезпечення.

Автор роботи: Казіміров В.О.

Керівник роботи: к.т.н., доц. Орленко В.С.

Пояснювальна записка: 88 с., 22 рис., 3 табл., 3 дод., 38 джерел.

КОМП'ЮТЕРНА БЕЗПЕКА, ПРОТИДІЯ ЗАГРОЗЛИВИМ ПРОГРАМАМ, ЗАХИСТ ІНФОРМАЦІЇ

Мета магістерської роботи полягає в розвитку методів захисту інформаційних ресурсів шляхом застосування розмежувальної політики доступу до виконуваних об'єктів.

Дана кваліфікаційна робота присвячена розробці моделі системи захисту, яка базується на контролі доступу що дозволяї оцінити можливі способи витоку прав доступу і сформулювати вимоги, дотримання яких забезпечує ефективний захист.

Використовуючи розглянуту модель засобів захисту та сформульовані вимоги, до інтенсивності виявлення помилок засобами захисту, при виконанні яких досягається високий рівень ефективності застосування пропонованого рішення на практиці.

Дата 9.12.20 Підпис студента

ANNOTATION

a master's degree work of Kazimirov Volodymyr

entitled «mproving the method of protecting information resources from malicious software».

Mentor: Viktoriia Orlenko

Total volume of work: 88 pages, 22 figures, 3 tables, 3 appendices, 38 references.

COMPUTER SECURITY, COMBATING THREATENING PROGRAMS, INFORMATION PROTECTION

The purpose of the master's work is to develop methods of protection of information resources through the application of a delimiting policy of access to executable objects.

This qualification work is devoted to the development of a model of security system, which is based on access control that allows to assess possible ways of leakage of access rights and to formulate requirements, compliance with which provides effective protection.

Using the considered model of means of protection and the formulated requirements, to intensity of detection of errors by means of protection at which performance the high level of efficiency of application of the offered decision in practice is reached.

Date 9. 12. 20 Signature

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ЗАХИСТУ ВІД ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ..... 11
1.1 Класифікація загрозливого програмного абезпечення..... 11
1.2 Способи виявлення загроз ……. 22
1.3 Підходи до виявлення загроз на інформаційні ресурси..... 26
1.4 Класифікація загрозливих програм за способами виконання шкідливих файлів……..……..……..……... 33
1.5 Постановка задачі ……..……..……..……..... 37
2 МОДЕЛЮВАННЯ ПРОЦЕСУ ЗАХИСТУ НА ОСНОВІ КОНТРОЛЮ ДОСТУПУ ДО РЕСУРСІВ..…..…….……..…….. 39
2.1 Оцінка актуальності загрози запуску ШПЗ...…..…… 39
2.2 Моделювання підходу до захисту від ШПЗ на основі контролю доступу до ресурсів…….…….…….…… 45
2.3 Модель захисту з використанням матриці розмежування доступу 48
2.4 Висновки ……..……..… 60
3 РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЙНИХ РЕСУРСІВ …… 61
3.1 Розробка вимог, для побудови безпечної системи …….. 61
3.2 Схема механізму забезпечення захисту ……..……... 63
3.3 Запровадження ПД для захисту від ШПЗ …….…….…….……... 65
3.4 Висновки ……..…… 68
4 ДОСЛіДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО МЕТОДУ…… 69
4.1 Реалізація способу підвищення безпеки ……..... 69
4.2 Оцінка ефективності запропонованого методу……... 74
4.3 Висновки ……,,…….……..... 80
ВИСНОВКИ…….…… 82
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 84

ДОДАТОК А Код (лістинг) програмного забезпечення виявлення загрозливих програм…… 89
ДОДАТОК Б Перелік наукових праць…….. 94
ДОДАТОК В Презентація…….. 100

ВСТУП

Актуальність роботи. Швидкий розвиток інформаційних систем та комп'ютерних технологій майже повністю змінив усі сфери сучасності. Завдяки постійній інтеграції інформаційних технологій досягається значне підвищення продуктивності праці та вдосконалення людської діяльності. Однак у результаті глобальної цифровізації кількість загроз цілісності інформаційних систем значно збільшується та диверсифікується. Це вимагає посиленого захисту кінцевих вузлів та винаходу нових адаптивних заходів безпеки.

Серед засобів захисту інформації (ЗЗІ) найпоширенішими є: апаратне, програмне та апаратно-програмне забезпечення. Останні є найбільш універсальними та гнучкими, оскільки дозволяють робити налаштування для будь-якого типу системи та архітектури, що актуально у зв'язку з активним розповсюдженням "Інтернету речей" (IoT). Цей сектор пристроїв є одним з найбільш вразливих і все частіше використовується для цілеспрямованих атак [19]. Однак настільні ПК все ще є лідерами за кількістю шкідливих програм, що розробляються.

Захист інформаційних ресурсів на сьогодні є важливим напрямком інформаційної безпеки підприємства. З кожним роком кількість інформаційних ресурсів збільшується, кількість конфіденційної інформації, яка локалізується на серверах віддаленого доступу також зростає.

Як результат, зростає не тільки кількість атак на веб-ресурси, але й економічні наслідки таких атак. Останнім часом вразливість інформаційних ресурсів до атак набула політичного виміру як через поширення гібридних війн у світі, так і через зростання терористичних загроз.

Таким чином, вдосконалення методів та систем захисту інформаційних ресурсів від атак залишається актуальною науковою проблемою, особливо з огляду на постійне вдосконалення методів та засобів атак та появу нових методів та засобів. Удосконалення методів захисту інформаційних ресурсів від нападів також є важливим завданням на практиці через зростаючі економічні, соціальні та політичні наслідки зловмисних дій.

Дослідженнями в цій галузі займалися зарубіжні і вітчизняні вчені, зокрема роботи Баришев А.Ф., Савенко О.С., Віртерс Дж., Віпперман К., , Котлер Ф., Ланкін

В.Є., Марков В.Д., Мазілкіна Є. І., Роджерса Л., Терьохіна К.І., Гольдштейн Г. Я та інших.

Інформаційна безпека оцінюється з початку інформаційних технологій. На цю тему існує багато робіт, але найбільш актуальними та фундаментальними роботами є нормативні документи, які внесли значний теоретичний та практичний внесок у вирішення проблем інформаційної безпеки, а саме: "Помаранчева книга" [12], яка викладає та систематизує критерії. оцінки захисту комп'ютерних систем; Європейські характеристики оцінки безпеки інформаційних технологій [23]; Федеральні критерії США [15] на замовлення уряду США, спрямовані на усунення обмежень, незручностей практичного застосування та недоліків "помаранчевої книги"; Міжнародний стандарт I80 / IEC 15408 - "Критерії оцінки безпеки інформаційних технологій" [16].

Проблема протидії загрозливому програмному забезпеченню, залишається досить гострою, навіть враховуючи появу більш ефективних механізмів його виявлення, аналізу, оновлення баз сигнатур і правил виявлення.

Об'єктом дослідження є процес впровадження та запуску шкідливого програмного забезпечення.

Предметом дослідження є методи та засоби захисту інформаційних ресурсів на основі розмежування доступу до виконуваних файлів.

Метою дослідження є розвиток методів захисту інформаційних ресурсів шляхом застосування розмежувальної політики доступу до виконуваних об'єктів.

Завдання дослідження. В магістерському дослідженні були поставлені і вирішені наступні завдання:

1. Проаналізовано основні типи загрозливого програмного забезпечення запропоновано їх класифікувати за способами завантаження і виконання шкідливих файлів.

2. Досліджено ефективність існуючих методів і засобів захисту, які базуються на сигнатурному і поведінковому аналізі.

3. Вдосконалено моделі і методів захисту від шкідливого програмного забезпечення на основі контролю доступу до файлів.

4. Сформульовано ряд вимог до побудови безпечної системи.

5. Розроблено функціональну схему, політику безпеки, направлених на захист інформаційної системи від шкідливого програмного забезпечення.

6. Проведено оцінку ефективності запропонованих методів захисту з врахуванням впливу на завантаження обчислювальних ресурсів інформаційної системи.

Наукова новизна дослідження:

1. Вдосконалено моделі системи захисту, яка базується на контролі доступу що дозволяї оцінити можливі способи витоку прав доступу і сформулювати вимоги, дотримання яких забезпечує ефективний захист.

2. Розроблено метод захисту інформаційних ресурсів, від завантаження і виконання бінарних і скриптових загрозливих файлів.

Практична цінність результатів роботи полягає в наступному:

Використовуючи розглянуту модель засобів захисту та сформульовані вимоги, до інтенсивності виявлення помилок засобами захисту, при виконанні яких досягається високий рівень ефективності застосування пропонованого рішення на практиці. Зокрема показано, що при інтенсивності виправлення помилок у засобі захисту, яка становить 3 - 7 днів (ці значення досяжні на практиці), величина ймовірності готовності засобів захисту до безпечної експлуатації становить 0,97 - 0,99, середній час безвідмовної роботи. збій інформаційної безпеки) засобів захисту в цьому випадку становить від шести місяців до двох років.

Методи дослідження. При вирішенні поставлених завдань використовувалися методи математичної статистики, теорії надійності, теорії графів, теорії масового обслуговування, теорії алгоритмів.

Апробація роботи. За темою магістерської роботи опубліковано 1 наукова стаття в нефаховому журналі і 1 теза доповідей на міжнародній конференції.

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ЗАХИСТУ ВІД ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Класифікація загрозливого програмного забезпечення

Програмне забезпечення - це сукупність програм, призначених для вирішення проблем на вашому комп’ютері. Програма - це впорядкований набір деяких команд. Програмне і апаратне забезпечення працюють взаємопов’язано та в постійній взаємодії. Будь-який апаратний пристрій контролюється програмним забезпеченням.

Програмне забезпечення умовно можна розділити на три класи: системне, прикладне і інструментальне. На теперішній час майже кожна програма має особливості кожного класу [21].

Системне програмне забезпечення призначене для керування комп’ютером, розподілу його ресурсів, створення діалогу з користувачами, сприяння обслуговуванню комп’ютера, і, також для автоматизації процесу створення нових програм. Системне програмне забезпечення - це набір застосунків, багато з яких постачаються разом з вашим комп'ютером та документацією на нього. Системне програмне забезпечення проводить контроль роботи комп’ютерної системи. Як правило, системні програми дозволяють взаємодіяти іншим програмам з апаратними компонентами, організовуючи інтерфейс користувача.

Термін шкідливе програмне забезпечення, зловмисне програмне забезпечення, означає програмне забезпечення, яке незаконно впроваджується у комп’ютерну систему та може спричинити порушення безпеки, пошкодити інформаційні ресурси, та в деяких випадках ще й ресурси комп'ютерного обладнання. системи [22].

Деякі програми можуть виконувати навіть певну руйнівну функцію, тому їх називають деструктивним програмним забезпеченням (російська - деструктивне програмне забезпечення, англійська - деструктивне програмне забезпечення). Однак шкідливе програмне забезпечення, що не має вбудованої деструктивної функції, також не є безпечним для комп’ютерної системи. По-перше, воно дарма витрачає ресурси системи, а по-друге, порушують політику безпеки. Загальною особливістю шкідливого програмного забезпечення є те, що воно призначене для порушення політики безпеки.

Шкідливе програмне забезпечення класифікується по різним критеріям. Наприклад, у [17] воно поділене на два види - перше виконує деструктивні функції, а інше - ні. В інших джерелах так звані програмні закладки виділяються як окремий видшкідливих програм. Іноді термін закладка програмного забезпечення відноситься майже до всіх шкідливих застосунків, крім комп'ютерних вірусів. На наш погляд, протиставлення вірусів та програмних закладок є неправильним. Будь-яке загрозливе програмне забезпечення може додавати закладки або ні. Програмна закладка програми буде працювати на комп'ютері деякий час, аж поки не буде виявлена відповідно до вбудованого алгоритму. Натомість деяке троянське програмне забезпечення може виконувати деструктивні дії з катастрофічними наслідками (наприклад, форматування диска) в момент активації, не намагаючись залишити їх компоненти у системі.

Тому ми вважаємо за доцільну класифікувати зловмисне програмне забезпечення за двома основними ознаками:

• спосіб розповсюдження інструменту - як інструмент потрапляє на комп’ютер і домагається його активації;

• призначення інструменту - які шкідливі дії він виконує.

Розглянемо основні типи загрозливих програм та принципи роботи [11, 21]. Вони поділяються на класи за такими основними характеристиками (рис. 1.1):

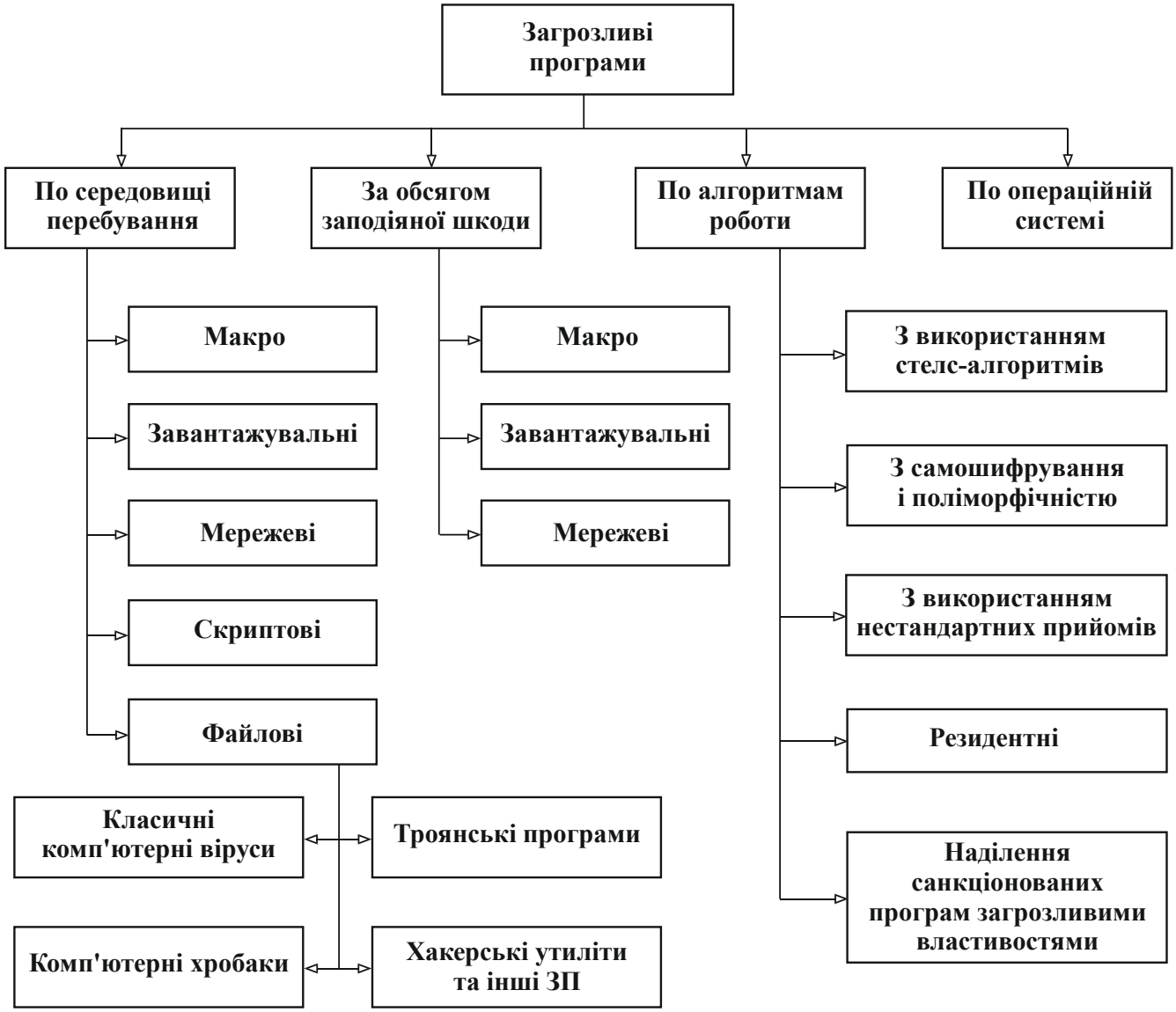


Рисунок 1.1 - Класифікація загрозливих програм

Через наявність великої кількості шкідливого програмного забезпечення та величезного спектру програм, кожного з видів ЗПЗ, можна провести його однозначний розподіли за класами. ЗПЗ включає: віруси, хробаків, троянських коней, руткіти, шпигунські програми, кейлоггери тощо. Відповідно до звіту [13], найбільш часто у 2020 році було виявлено рекламні віруси (близько 20000 виявлень у серпні 2020 року). Графік виявленого шкідливого програмного забезпечення за рік зображено на рис. 1.1.

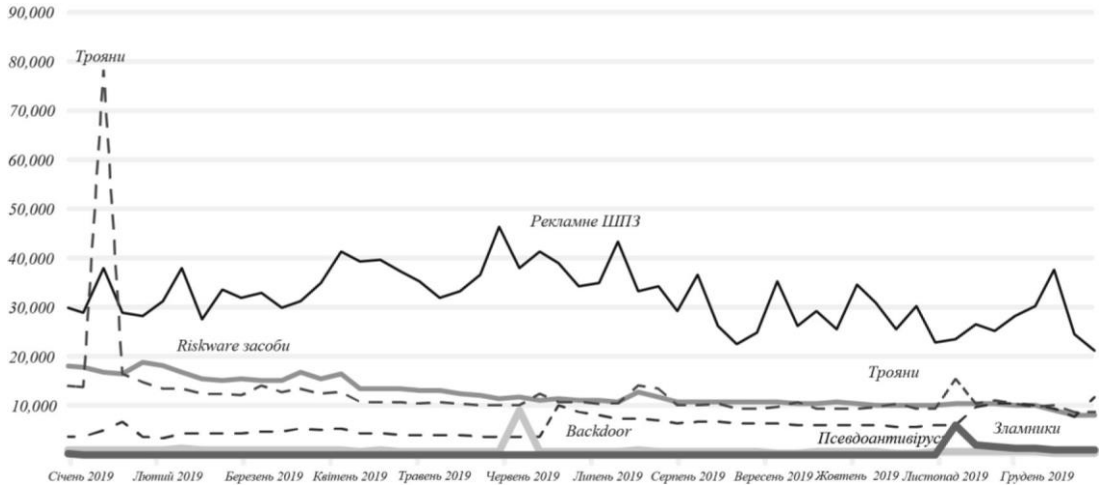


Рисунок 1.2 - Топ-10 категорій загроз для споживачів у 2020 році

Розкриємо детальніше особливості роботи кожного з видів шкідливого програмного забезпечення.

Комп'ютерні віруси - це програмні засоби, які здатні самовідтворюватися, тобто відтворюватися та використовувати як інший програмний код, який вони змінюють таким чином, щоб вбудувати в нього свою копію. В результаті замість коду програми, запущеного користувачем для виконання, виконується код вірусу. Детальніше про комп'ютерні віруси буде розглянуто далі в цьому розділі. Віруси - це зловмисне програмне забезпечення з механізмом самовідтворення. Вони існують як виконуваний файл і розповсюджуються шляхом копіювання в інші хост-системи. Оскільки це пасивний тип програмного забезпечення, зараження відбувається через файли, носії інформації або мережеві файли. Залежно від того, наскільки складним є програмний код, він може навіть модифікувати свої дублікати [36]. Віруси можна використовувати не тільки для пошкодження комп'ютерних вузлів і мереж, але й для крадіжки інформації, створення ботнетів, відображення реклами тощо.

За час свого існування типовий вірус проходить 4 стадії:

- фаза спокою. Вірус не працює, але чекає події, яка його активує. Ця подія може бути певною датою, наявністю іншого файлу або овердрайвом. Та не всі віруси дотримуються цієї стратегії;

- фаза розмноження. Вірус розміщує свою копію в інших програмах або у деякихсистемних областях на диску. Тоді кожна заражена програма містить копію вірусу, яка також колись почне розмножуватися;

- стадія запуску. Вірус активується, щоб мати можливість виконувати функції, для яких він був створений. Окрім виходу з фази спокою, перехід до фази запуску може бути визівний різними системними подіями (в тому числі - перевищення певної допустимої кількості нових копій вірусу);

- фаза виконання. ШПЗ виконує свої функції. Вони можуть бути безпечними (відображати повідомлення) або завдавати шкоди (видаляти файли з додатками та даними) [16].

Більшість вірусів виконують деструктивні дії, пристосовуючись до ОС, абодеяких випадках – до деякої апаратної платформи., тобто використовують особливості та слабкості операційних систем.

Хробаки також мають механізм реплікації, але є активним шкідливим програмним забезпеченням, яке поширюється по мережі за допомогою різних вразливостей в операційній системі або встановленому програмному забезпеченні. Класичні мережеві хробаки здатні поширюватися самостійно, без будь-якого втручання користувача, в комп'ютерну мережу, виконуючи принаймні дві функції: передаючи свій програмний код на інший комп'ютер і запускаючи свій програмний код на віддаленому комп'ютері [26]. Для цього хробаки використовують уразливості в комп’ютерних системах. На відміну від класичних вірусів, хробаки, як правило, не використовують код інших програм як носій інформації, оскільки вони не покликані змусити користувача запускати їх таким чином. Хробаками іноді називають програмами, що не можуть самостійно запускатись на віддаленому комп'ютері, а тому застосовують принцип "троянського коня".

Мережні хробаки використовують мережеві з'єднання для переходу від однієї системи до іншої. Після активації в системі хробак може поводитись як звичайний комп’ютерний вірус, породжувати троянських коней або виконувати інші руйнівні чи деструктивніі дії [21].

Хробак використовує деякі носії для самовідтворення:

- електронна пошта - він надсилає свою копію в інші системи;

- можливість віддаленого запуску програми - він запускає свою копію в іншій системі;

- можливості віддаленої реєстрації - черв'як заходить у віддалену систему під виглядом користувача, а далі використовує стандартні команди для копіювання із однієї системи в іншу.

Перш ніж скопіювати себе в систему, мережевий черв'як може спробувати ідентифікувати заражену систему. Крім того, у багатозадачній системі це можна приховати, називаючи системні процеси чи інші імена, які важко одразу помітити системному адміністратору.

Найвідоміший представник цього класу - вірус Морріса (вірніше, «черв'як Морріса»), який потрапив в Інтернет в 1988 р. Найпопулярніше середовище поширення глистів. Існує мережа, де всіх користувачів вважають друзями і довіряють один одному, а відсутність механізмів безпеки найкраще сприяє уразливості мережі.

Найкращий спосіб захисту від хробаків - це вжити певні заходи для запобігання несанкціонованому доступу до мережі.

Отже, як і віруси, троянські програми та хробаки є однією з найнебезпечніших загроз для комп'ютерної системи сьогодні. Для захисту від таких типів шкідливих програм потрібно створити закрите середовище для запуску програм, обмежити доступ до виконуваних файлів, контролювати цілісність виконуваних файлів і системних областей, протестувати придбане програмне забезпечення.

Троянський кінь - це програма, яка виконує додаткові дії, але не описані в документації, крім основних (розроблених та задокументованих). Троянський кінь - це корисна або, здавалося б, корисна програма чи процедура, що приховує код, який може активувати небажану або якусь шкідливу функцію при активації [24].

Таким чином, аналогія з давньогрецьким троянським конем виправдана - в обох випадках у шкаралупі є загроза, яка не викликає жодних підозр. Програми цього класу становлять серйозну загрозу для безпеки комп'ютерних систем.

Троянські програми можуть застосовуватися для виконання функцій, які неавторизований користувач не може безпосередньо виконувати.

За своєю природою троянський кінь є активною загрозою, що реалізується програмним забезпеченням та працює у пакетному режимі. Троянський кінь - це загроза будь-якому об'єкту комп'ютерної системи, і ця загрозлива дія може бути виражена будь-яким з можливих способів: прямий вплив на об'єкт атаки, вплив на систему дозволів,

непрямий вплив. Найнебезпечнішим є непрямий вплив, під яким троянський кінь діє в межах повноважень одного користувача, та в інтересах іншого користувача, ідентичність якого встановити практично неможливо.

Небезпека троянського коня полягає у додатковому блоці команд і його тілі, встановлених тим чи іншим способом у початковій нешкідливій програмі, яка, таким чином, пропонується (подарунок, заміна, продаж) користувачам комп'ютерної системи. Цей командний блок може працювати, при умові виконання певної умови (дата, час тощо, чи зовнішня команда). Той, хто запускає таку додатрк, представляє загрозу для своїх файлів, і також для всієї комп'ютерної системи. Таким чином, у деяких випадках певні логічні бомби також можна віднести до троянів.

Троянський кінь також може виконувати найнебезпечніші дії, при умові, що користувач, який його запустив, володіє розширеним набором привілеїв. В цьому випадку зловмисник, який створив і впровадив троянського коня, але він не має цих привілеїв, має можливість виконувати несанкціоновані привілейовані функції за чужий рахунок. Або, наприклад, зловмисник дуже цікавиться деякими наборами даних користувача, що запускав цю програму. Останні можуть навіть не володіти розширеним набором привілеїв - це не буде перешкодою для виконання несанкціонованих дій.

Прикладом троянського коня, якого дуже важко виявити, є компілятор, який був змінений для вставки додаткового коду у певні програми (наприклад, системні входи) в момент компіляції. За допомогою цього коду в програмі є можливість створити люк, який дозволяє автору входити в систему за допомогою спеціального пароля. Такого троянського коня не можна знайти у оригінальному тексті програми реєстрації. Тому, люки можна віднести до троянців.

Троянський кінь - одна з найнебезпечніших загроз для безпеки операційних систем. Радикальним сметодом захисту від цієї загрози є розроблення закритого програмного середовища. Бажано також, щоб привілейовані і непривілейовані користувачі працювали із різними екземплярами програм, які слід зберігати і захищати індивідуально. Якщо ці заходи будуть виконані, ймовірність виконання таких програм буде досить низькою.

Порівняно з вірусами, троянські коні не є такими поширеними з простих причин – зрозуміло, що вони знищать себе разом із іншими даними на диску, чи виявляють свою присутність та знищуються користувачем ,яким завдано шкоди.

До категорії троянців також належать програми вандалів. Ці програми зазвичай імітують роботу будь-якої корисної функції або маскуються під нову версію відомого програмного продукту. Як побічний ефект вони знищують файли, пошкоджують каталоги, форматують диски або виконують деякі інші руйнівні дії.

Шпигунське програмне забезпечення - це шкідлива програма, що використовує функції операційної системи для шпигунства за діяльністю користувачів [21]. Іноді таке програмне забезпечення ще має додаткові функції, такі як втручання у мережеве підключення чи навіть зміна параметрів захисту зараженої системи. Поширення відбувається шляхом підключення легітимного програмного забезпечення, троянських коней чи навіть відомих вразливостей. Шпигунське програмне забезпечення має можливість контролювати поведінку користувачів, наприклад, збираючи натискання на клавіші та надсилаючи інформацію на віддалений хост зловмиснику.

Люк - це прихована бездокументована точка входу у програмний модуль, яка дозволяє всім, хто знає про її існування, отримати доступ до програми, минаючи звичайні заходи, призначені для забезпечення безпеки КС. Люк інтегрується у програму в більшості випадків безпосередньо на етапі налагодження для полегшення роботи - цей модуль можна викликати у різних місцях, що дозволяє налагоджувати деякі його частини незалежно одна від іншої. Крім того, люк можна інегрувати на етапі розробки для подальшого підключення цього модуля до інших системних модулів, але тоді, через змінені умови, ця точка входу вже буде непотрібною.

Як правило, програміст проектує програмний застосунок, що передбачає процес реєстрації або налаштування якого триває довго, вводячи багато різних параметрів під час запуску. Можливо, розробник бажає надати програмі спеціальні привілеї чи мати можливість запобігти процесу встановлення і автентифікації, чи програміст повинен мати надійний метод, який дозволяє активувати програму в разі можливих збоїв.

Наявність бекдору дозволяє викликати програму нестандартно, що може серйозно вплинути на стан системи безпеки (невідомо, як в цьому випадку програма сприйме

інформацію, середовище системи тощо). Крім того, не завжди можливо вгадати її поведінку.

Бекдори можуть з'являтися в програмах з таких причин:

- вони забули усунути (ненавмисна помилка);

- для подальшого коригування;

- забезпечити підтримку готової програми;

- здійснювати секретний контроль доступу до цієї програми після її інсталяції (перший крок до навмисного проникнення на комп'ютер за допомогою цієї програми).

Помилки в програмному забезпечення не є люками. Бекдор - це механізм налагодження для обслуговування і налаштування програм. Якщо бекдори використовуються для отримання несанкціонованого доступу о системи, вони стають загрозою.

Їх створення можна запобігти, проаналізувавши оригінальні тексти застосунків, заходи безпеки повинні застосовуватися головним чином на стадії розробки та оновлення програм.

Руткіти застосовують набір інструментів, щоб уникнути виявлення у системі. Інструменти - це, зазвичай, вдосконалені та складні застосунки, призначені для маскування загальних процесів на зараженому сайті. Через своє маскування вони дуже інвазивні і їх складно видалити. Розроблений з метою повного контролю над системою і отримання найвищих привілеїв на зараженому сайті [19]. Через методи маскування, які використовуються руткітами, багато готових програмних рішень захисту вузлів не ефективні для їх безпосереднього виявлення та видалення. Засоби боротьби з руткітами супроводжуються спостереженням за поведінкою комп'ютерної системи на предмет виявлення нехарактерних дій, аналізом дампів пам'яті і скануванням підписів системних файлів.

Боти - це програми, призначені для виконання конкретних операцій. Вперше були розроблені боти для управління каналами чату. Деякі з них використовуються в законних цілях, але шкідливі боти призначені для формування ботнетів. Ботнет - це мережа хост-комп'ютерів (зомбі / ботів), керованих зловмисником. Боти заражають і керують іншим комп'ютером, який, у свою чергу, заражає ще й інші підключені комп'ютери, утворюючи деяку мережу скомпрометованих комп'ютерів, яка називається

ботнетом. Боти часто застосовуються як спам-боти, DDoS-атаки, веброзповсюджувачі для поширення зловмисного програмного забезпечення на сайтах обміну файлами, репозитаріях. Для контролю використовуються тести SARTSNA [29].

Здирник заражає комп'ютерні вузли або мережі та утримує систему закритою, що вимагає погашення від користувачів. Як правило, файли шифруються в зараженій системі або система блокується, щоб заборонити доступ користувачеві. Потім відображаються повідомлення, які змушують вас платити за доступ до даних. Використовує ті ж засоби розповсюдження, що і комп’ютерні хробаки.

Логічна бомба - це код, який розміщується в законній програмі. Він влаштований таким чином, що за виконання певних умов він «вибухає». Умовою запуску логічної бомби може бути присутністьь або відсутність певних файлів, якийсь день тижня або певна дата, а також запуск програми конкретним користувачем.

Кейлоггери - це різновид шпигунського програмного забезпечення. Існує невидимий для звичайного користувача запис клавіш, які він натискає на свому клавіатурі. Потім ці дані передаються зловмиснику через мережу Інтернет [21]. Ці програми використовуються для перехоплення паролів, наприклад, в паролях для Інтернет-банкінгу. Ці програми також можуть використовувати шпигунське програмне забезпечення для потреб викрадення іншої особистої інформації, наприклад документів, які зберігаються на вебсайті комп’ютера. Широкий набір методів використовується для завантаження шпигунського програмного забезпечення та реєстраторів ключів на віддалений вебресурс.

Backdoor - це таємний "вхід" для доступу. Іноді його планують створити розробники системи або постачальники послуг як віддалений інструмент для діагностики, усунення несправностей або інших системних тестів. Саме існування задніх дверей накладає великі ризики для безпеки, оскільки виявити їх важко. ШПЗ, такі як вірус "задніх дверей", поширені на основі безпечного руху назад. Це шкідливий код, що використовуючи системні вразливості та експлойти, використовується для спрощення віддаленого несанкціонованого доступу до комп’ютерної системи чи програми. Як і весь зловмисний код, він працює у фоновому режимі. Цей доступ забезпечує повноту для загрозливих операцій у системі. Комп’ютери дуже вразливі до

незаконного копіювання файлів, їх змін, крадіжки даних через здійснення маніпуляцій [18].

Програми, які загрожують сценарію (скрипту). Загрожуючі сценарії або виконувані сценарії - це клас СПЗ, для виконання яких має бути встановлений спеціальний інтерпретатор (віртуальна машина). Це відрізняє їх від виконуваних файлів. Скриптові як і файлові віруси класу файлів, також присутні в файловому середовищі будь-якої операційної системи. Є багато видів застосунків скриптових прогам, написаних з використанням різнич технологій: файли PHP, файли BAT, rhbgnb Visual Basic та Java.

1.2 Способи виявлення загроз

Для виявлення шкідливих програм використовуються різні методи аналізу коду. Одним з них є статичний аналіз. Статичний аналіз означає аналіз шкідливого програмного забезпечення без його виконання. Шаблони виявлення, що використовуються при статичному аналізі, включають підписи рядків, послідовності байтів, n-грами, виклики синтаксису бібліотеки, графіки потоків управління, аналіз частотного коду операційних кодів тощо [11]. Аналіз виконується шляхом попереднього розпакування та декодування виконуваного файлу. Інструменти для розбирання та налагодження, аналізу дампа пам'яті використовуються для перегляду принципу роботи. Процес реінжинірингу показаний на рис. 1.2

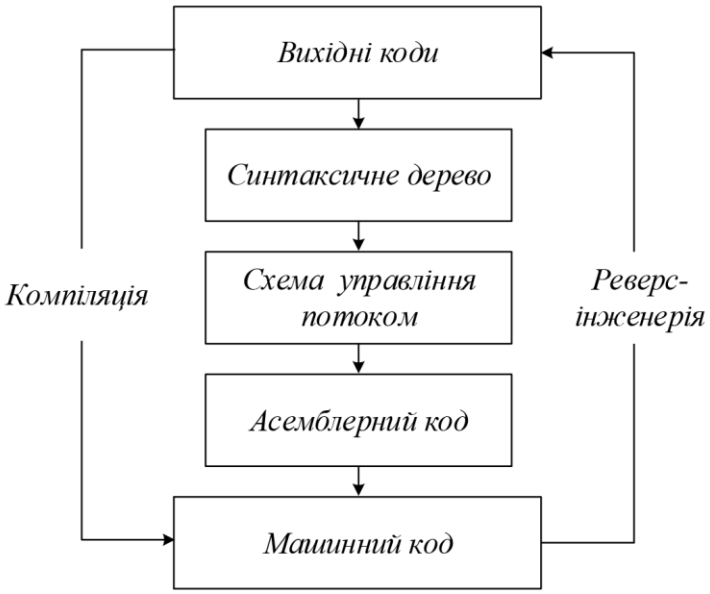


Рисунок 1.2 - Аналіз загрозливих програм за використання підходу реінжинірингу

Технічні методи ухилення від статичного аналізу зловмисниками призвели до розвитку динамічного аналізу. Мозер та ін. [13], досліджували недоліки методології статичного аналізу. Він представив схему, засновану на обфускації коду, яка доводить неадекватність статичного аналізу для виявлення або класифікації шкідливих програм. Згідно з дослідженнями, динамічний аналіз є необхідним доповненням до статичного аналізу, оскільки він менш вразливий до затуманення.

Дезасемблювання - це процес розробки набору специфікацій для складної апаратної системи шляхом упорядкованого вивчення зразків цієї системи. Ця робота насамперед стосується зворотніх інженерних двійкових файлів або зворотного розвитку коду. При програмуванні зворотного коду метою є розробка набору специфікацій, поведінка двійкового коду - зі складної програмної системи.

Зворотну інженерію коду можна розділити на два методи: моніторинг поведінки виконуваного файлу та розбирання або декомпіляція двійкового коду в зручний для читання формат. [31] Як правило, зворотний інженерний процес називається демонтажем, якщо кінцевий продукт являє собою представлення мови збірки та декомпіляції двійкового коду, або мову, що триває протягом усього життя.

Управляючи двійковим файлом, ми можемо відповісти на дуже складні запитання, наприклад "Які файли запускають конкретний двійковий файл?", "До яких хостів підключається конкретний двійковий файл?", "У яких портах цей двійковий файл відкритий і підключений?" Чи “Цей двійковий файл намагається перезапустити інші запущені процеси? "і т. д. Оскільки моніторинг виконання виконуваного файлу вимагає його виконання, це може становити ризик для безпеки скасування двійкового коду шляхом моніторингу без відповідних запобіжних заходів, таких як віртуалізація або ізольований комп'ютер, призначений для запуску потенційно шкідливого програмного забезпечення. Хоча моніторинг може бути потужним методом одержання аналізу двійкового коду, неможливо визначити, яким чином двійковий файл виконує всі свої вбудовані дії через вбудовані системні обмеження, такі як обмежений інтервал роботи або обмежене підключення до мережі Інтернет, або тому, що вони містять спеціальні

особливості. [23].

Динамічний аналіз шкідливих програм передбачає аналіз застосуну під час його роботи в системі [22]. Запуск загрозливого програмного забезпечення в захищеному та контрольованому середовищі для запобігання передачі загрозливого програмного забезпечення, яке досліджується, до інших систем чи мереж. Основний динамічний аналіз передбачає спостереження за зібраним зразком і його взаємодіюі з системою. Спочатку робиться знімок початкового стану віртуальної машини, далі шкідливе програмне забезпечення готується для запуску в тестовій системі. Вихідні та вхідні стани порівнюються для виявлення змін. Зміни, отримані в результаті спостережень, потім використовуються для видалення із заражених вузлів та / або для моделювання ефективних сигнатур. Як і базовий статичний аналіз загрозливого програмного забезпечення, динамічний аналіз є важливим початковим етапм в аналізі зловмисних програм, хоча він не надає вичерпної інформації користувачу про шкідливе програмне забезпечення [27].

Розширений динамічний аналіз передбачає використання інструментів для вивчення стану шкідливої програми під час її запуску. Наприклад, вивчається внутрішній стан шкідливого коду. Використання передових методів аналізу дає інформацію, яку неможливо зібрати за допомогою інших методів [40]. Динамічний аналіз завжди виконується в ізольованому середовищі, щоб переконатися, що всі входи та виходи системи відомі для подальшого розгляду.

Аналіз параметрів, що використовуються під час дзвінків, та функції API дозволяє групувати функції, що використовуються семантично, тоді як аналіз даних, що обробляються та розподіляються в системі, дозволяє зрозуміти файли, що використовуються та створюються шкідливим програмним забезпеченням. Вони ведуть до визначення мети, для якої було розроблено шкідливе програмне забезпечення [19]. Розширений динамічний аналіз шкідливого програмного забезпечення дуже корисний для виявлення варіантів шкідливого програмного забезпечення та прихованих методів.

Безпека віртуальних машин. Апаратні розширення у сучасних процесорах x86 забезпечують багато способів контролю продуктивності віртуальної машини. Хоча розширення апаратної віртуалізації пропонують кілька способів приховати існування віртуалізації [21], на жаль, цього не можна гарантувати в реальному світі, оскільки

процес апаратної віртуалізації є складним та схильним до помилок. Наприклад, уразливість програмного забезпечення для віртуалізації Xen (CVE-2015-5166 [30]), що збільшує привілеї гостя над хост-системою, або вразливість продуктів VMware (CVE-2012-2450 [22]), що призводить до відмови в обслуговуванні і, можливо, збільшення привілеїв гостя над господарем.

Завжди існує можливість уразливості програмного забезпечення; запуск ненадійного виконуваного файлу в ізольованій віртуальній машині безпечніший (принаймні на 2 порядки), ніж запуск без віртуалізації. Виходи з гостьового режиму, такі як два наведені вище приклади, в кінцевому рахунку надзвичайно рідкісні [38], а відмова в обслуговуванні гостьової системи не даєєє контролювати виконання двійкового коду без будь-яких проблем із безпекою.

Для зручності застосовуються автоматизовані засоби динамічного аналізу зловмисних програм, що надають звіти, які можна використовувати для групування за поведінкою шкідливого програмного забезпечення.

1.3 Підходи до виявлення загроз на інформаційні ресурси

Завданням антивірусних програм є пошук комп'ютерних вірусів та небажаних програм загалом та відновлення заражених файлів такими програмами, а також профілактика - запобігання модифікації файлів або операційної системи за допомогою шкідливих програм код.

Проаналізувавши методи пошуку ШПЗ за допомогою антивірусних програм, стає зрозумілим, що антивірусні програми не є повністю надійними. Кожен метод пошуку вірусів не є надійним, оскільки кожен метод має багато недоліків [3].

Були проаналізовані наступні методи виявлення вірусів, а саме:

- Визначення на основі підписів - Цей метод дозволяє ідентифікувати конкретну атаку з високою точністю та невеликою часткою помилкових дзвінків. Метод є беззахисним проти поліморфних вірусів та модифікованих версій того самого вірусу і вимагає регулярного та надзвичайно швидкого оновлення [13].

- Виявлення аномалій - на відміну від методу узгодження визначення вірусу у словнику, метод підозрілої поведінки забезпечує захист від абсолютно нових вірусів та

мережевих атак, яких ще немає в жодній базі вірусів чи атак. Однак програми, побудовані за цим методом, також можуть видавати велику кількість помилкових тривог, що робить користувача дуже чутливим до попереджень.

- Виявлення на основі емуляції - У деяких випадках емуляція є досить ефективною у протидії таким технологіям, як поліморфізм шкідливих програм, що досягається оцінкою вжитих дій, але не програмного коду. Безсумнівним недоліком процесу емуляції є велике споживання системних ресурсів, що негативно впливає на продуктивність комп'ютера.

На ранніх стадіях широко застосовувався метод виявлення, заснований на підписах. Цей метод працює швидко та ефективно для виявлення відомих типів шкідливих програм. Однак цей метод не може бути використаний для виявлення ІП, які використовують уразливості нульового дня [4]. Пізніше з’явились поведінкові, евристичні та інші методи.

Кожен із підходів до визначення ключових ознак відрізняється один від одного та має свої переваги і недоліки, тому не можна однозначно стверджувати, що той чи інший метод кращий.

Основою реалізації структурно незалежних механізмів ідентифікації шкідливих програм в захищених інформаційних системах є методи їх виявлення за допомогою професійних пакетів антивірусних інструментів. Основними методами є:

- сканування;
- евристичне сканування;
- сканування ВЯП;
- антивірусний моніторинг;
- імунізація.

Класифікація методів виявлення загрозливих програм та їх ознак зображена на рис. 1.3.

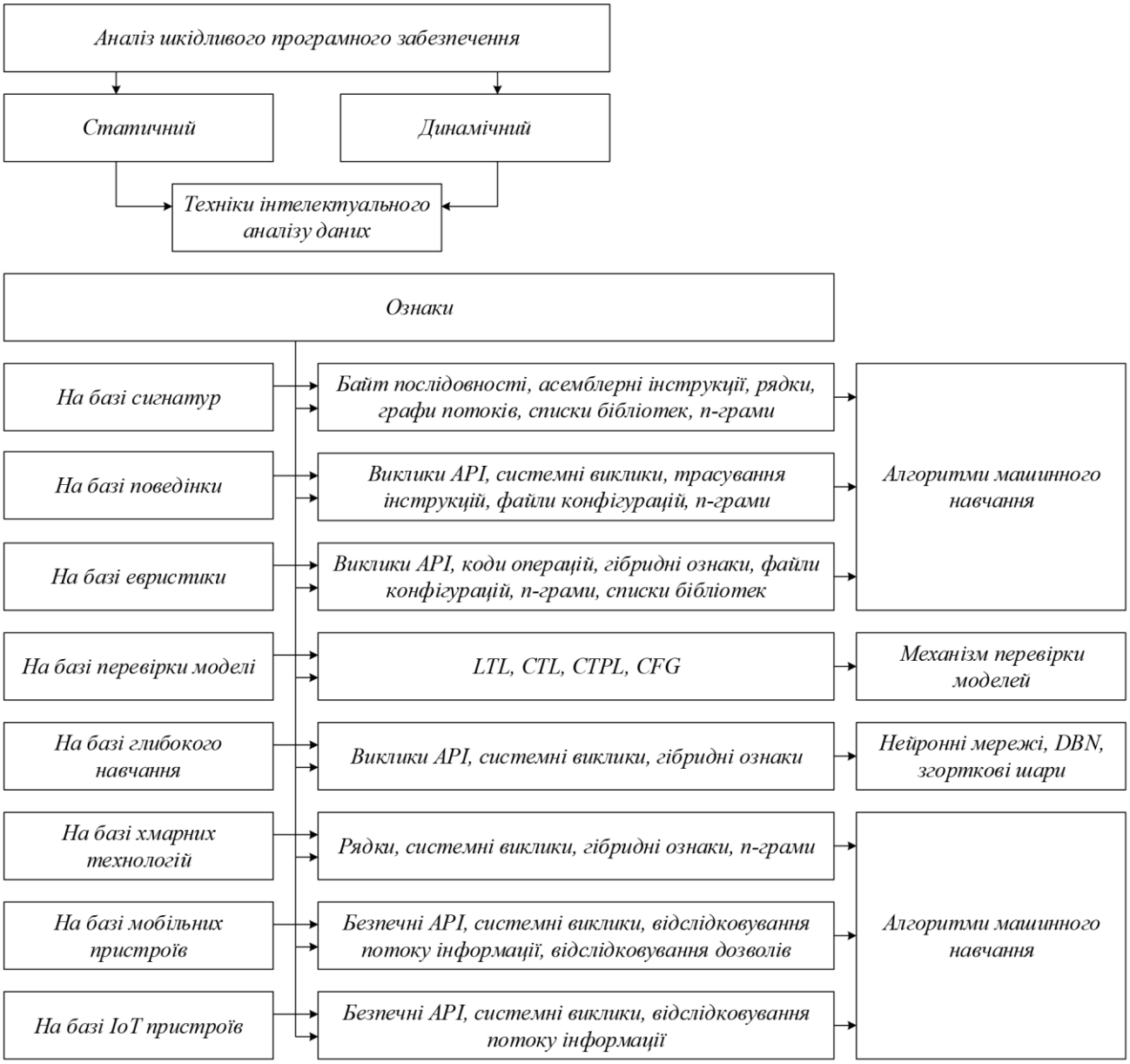


Рисунок 1.3 - Класифікація методів виявлення загрозливих програм

Використовуючи підходи до виявлення загроз на основі поведінкових зразків, евристики та моделювання, можна ідентифікувати велику кількість ШПЗ. Крім того, ці моделі також виявляють нові шкідливі програми. Однак вони не є універсальними і не можуть виявити всі шкідливі програми. Тому існує велика потреба у пошуку методу, який ефективно виявляє більш складні, невідомі програми.

Поведінковий підхід до виявлення шкідливого програмного забезпечення, дотримується поведінки програми за допомогою інструментів моніторингу та визначає, чи є програма шкідливою. Незважаючи на зміну програмного коду, поведінка

залишається подібною, що дозволяє застосовувати цей підхід для виявлення більшості нових шкідливих програм [12]. Однак деякі шкідливі програми не працюють належним чином у захищеному середовищі, тому зразок шкідливого програмного забезпечення може вважатися безпечним.

Схема визначення ЗП за поведінковим підходом наведена на рис. 1.4

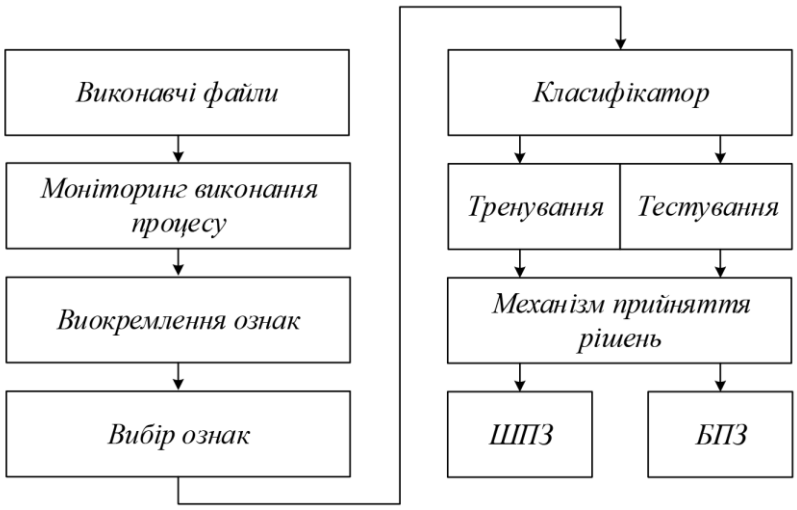


Рисунок 1.4 - Схема визначення ЗП за поведінковим підходом

Сучасні антивірусні програми застосовують еврістичний підхід до виявлення ЗПЗ. Це складний метод виявлення, який використовує досвід та різні методи, такі як правила та техніки машинного навчання [37]. Якщо сигнатурний метод базується на виділенні характерних ознак вірусу та пошуку цих ознак у файлах, які перевіряються, то евристичний аналіз базується на дуже правдоподібному припущенні, що нові віруси не рідко виявляються схожі на які-небудь із вже відомих. Постфактум таке припущення дійсно виправдовується наявністю у антивірусних базах сигнатур для визначення не одного, а певного типу вірусів. Евристичний метод, заснований на такому припущенні полягає в пошуку файлів, які не повністю, але дуже схожі на сигнатури відомих вірусів.

Позитивним ефектом використання цього методу є можливість виявлення нових вірусів до їх підписання. Недоліки:

• Ймовірність помилкового виявлення присутності вірусу у файлі, коли файл фактично чистий - ці події називаються помилковими спрацьовуваннями антивірусу.

• Неможливість лікування - саме через можливі помилкові спрацьовування ічерез можливе неточне визначення типу шкідливої програми, спроба лікування може призвести до більшої втрати інформації, ніж сам вірус, і це неприпустимо.

• Низька ефективність - проти справді інноваційних вірусів, які спричиняють найбільші епідемії, цей тип евристичного аналізу мало корисний.

Інший метод, заснований на евристиці, базується на припущенні, що загрозливе програмне забезпечення якимось чином намагається завдати шкоди вашому комп'ютеру. Метод заснований на виділенні основних загрозливих дій, таких як, наприклад:

• Видалити файл

• Записати у файл

• Запис у певних областях реєстру

• Відкриття порту прослуховування

• Перехоплення даних, введених з клавіатури

• Надсилання листів тощо.

Зрозуміло, що здійснення кожної такої дії окремо не є підставою вважати програму шкідливою. Але якщо програма виконує кілька таких дій, наприклад, записує сам запуск в ключ автозапуску реєстру, перехоплює дані, введені з клавіатури, і з деякою частотою надсилає ці дані на адресу в Інтернеті, це означає, що програма є принаймні підозрілим. Евристичний аналізатор, заснований на цьому принципі, повинен постійно контролювати дії, що виконуються програмами.

Перевагою цього методу є можливість виявлення раніше невідомих загрозливі програм, навіть якщо вони не сильно схожі на вже відомі. Наприклад, нове загрозливе програмне забезпечення може використовувати нову вразливість для проникнення на комп'ютер, але потім починає виконувати звичайні шкідливі дії. Таку програму має можливість пропустити евристичний аналізатор першого типу, але безпосередньо виявити може аналізатор другого типу.

Евристичний підхід до виявлення ШПЗ представлений на рис. 1.5

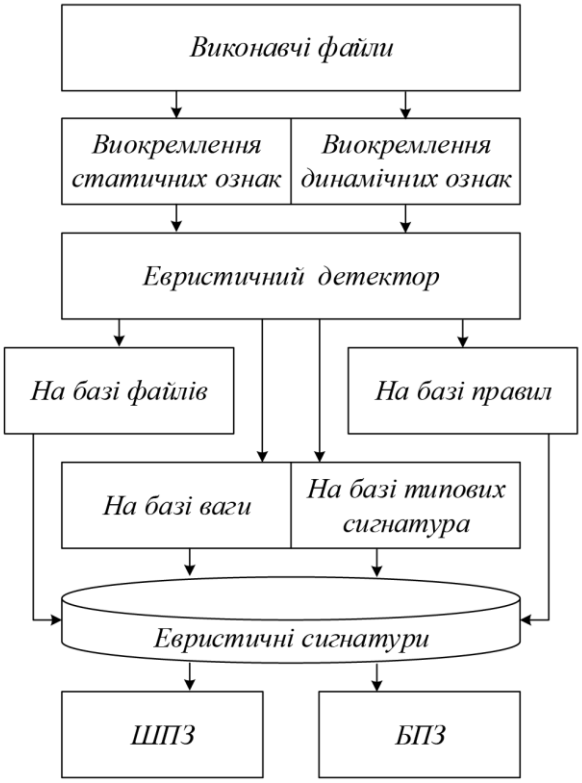


Рисунок 1.5 - Евристичний підхід до виявлення ШПЗ

Програмна поведінка створюється шляхом перегляду взаємозв'язків потоку одного або декількох системних викликів та визначення поведінки за допомогою таких властивостей, як приховування, розповсюдження та ін'єкція [37]. Порівнюючи цю поведінку, визначається, чи є програма загрозливою.

Виявлення загрозливого програмного забезпечення у хмарі підвищує ефективність виявлення для персональних комп'ютерів та мобільних пристроїв із набагато більшими базами даних, які потребують більших обчислювальних ресурсів.

Хмарне виявлення використовує різні типи агентів виявлення на хмарних серверах і пропонує безпеку як послугу. Користувач може завантажувати файли будь-якого типу та отримувати звіт про те, визнано завантажений файл загрозливим чи ні (рис 1.5).

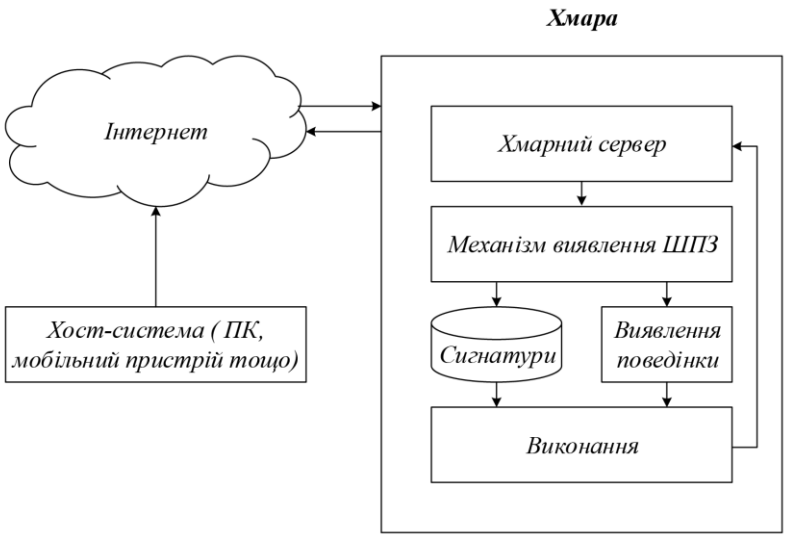


Рисунок 1.6 - Хмарна схема виявлення ШПЗ

Для платформи Android запропоновано численні методи виявлення загрозливого програмного забезпечення. Як правило, ці методи використовують алгоритми передачі даних та машинного навчання для виявлення шкідливих програм. Використовується низка різних функцій, таких як системні дзвінки, API, що чутливі до безпеки, інформаційні потоки та структури потоків управління.

Хоча сучасні дослідження покращили виявлення традиційних шкідливих програм наступного покоління, виявлення складних шкідливих програм залишається проблемою.

1.4 Класифікація загрозливих програм за способами виконання шкідливих файлів

На підставі проведених досліджень усіх типів ШПЗ пропонується провести їх класифікувати за підходами до виконання загроливих файлів. Така класифікація дасть можливість проаналізувати найбільш поширенні способи реалізації ШПЗ в інформаційній системі. Їх можна розділити на виконувані та макропрограми, у свою чергу виконувані програми поділяються на двійкові, що включають завантажувальні програми, класичні комп'ютерні віруси, троянські програми, комп'ютерні хробаки, хакерські утиліти, потенційно небажане програмне забезпечення та загрозливі програми які використовують сценарії виконання.(рис. 1.7).

Відповідно до звіту Cisco, основними ШПЗ, які використовуються, є троянські програми, які є частиною двійкових виконуваних загроз.

Згідно з графіком виявлених загрозливих програм за 2020 рік, наведеного на рис. 1.1, ми можемо зробити висновок, що найважливішим завданням є захист від загрозливих програм, що представляють собою двійкові файли та виконувані файли скриптів.

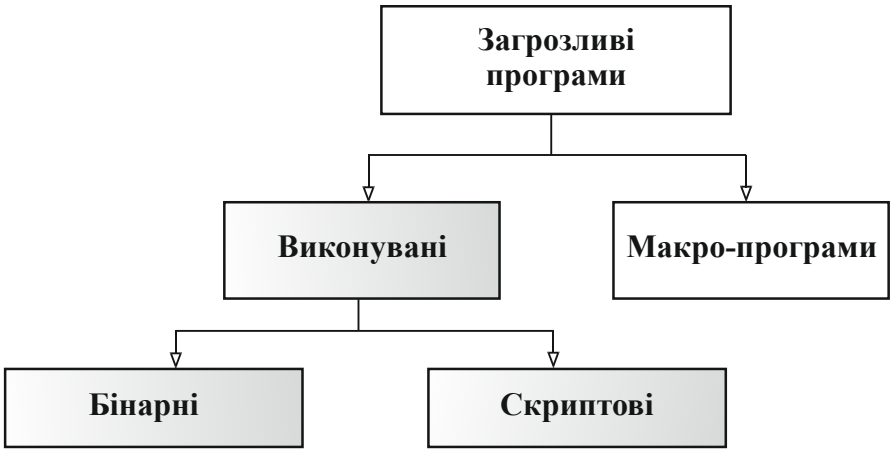


Рисунок 1.7 - Запропонована класифікація загрозливих програм

Необхідним для вірусів і кібер-злочинців завданням є впровадження вірусу, хробака або троянської програми в комп'ютер-жертву або мобільний телефон. Досягається ця мета різними способами, які діляться на дві основні категорії:

• соціальна інженерія (також вживається термін «соціальний інжиніринг» - калька з англійського «social engineering»);

• технічні прийоми впровадження шкідливого коду в заражає систему без відома користувача.

Часто ці способи використовуються одночасно. При цьому так само часто використовуються спеціальні заходи з протидії антивірусним програмам.

Соціальна інженерія [32]. Методи соціальної інженерії тим чи іншим способом змушують користувача запустити заражений файл або відкрити посилання на заражений веб-сайт. Ці методи застосовуються не тільки численними поштовими хробаками, а й іншими видами шкідливого програмного забезпечення.

Завдання хакерів і вірусів - привернути увагу користувача до зараженому файлу (або HTTP-посилання на заражений файл), зацікавити користувача, змусити його клікнути по файлу (або за посиланням на файл).

Використовуються також можливості файлообмінних мереж (P2P-мережі). Черв'як або троянська програма викладається в P2P-мережу під різноманітними «смачними» назвами, наприклад:

• AIM & AOL Password Hacker.exe

• PornStar3D.exe

• Microsoft CD Key Generator.exe

У пошуку нових програм користувачі P2P-мереж натикаються на ці імена, скачують файли і запускають їх на виконання.

Технології впровадження [29]. Ці технології використовуються зловмисниками для впровадження в систему шкідливого коду таємно, не привертаючи уваги власника комп'ютера. Здійснюється це через уразливості в системі безпеки операційних систем і в програмному забезпеченні. Наявність вразливостей дозволяє виготовленому зловмисником мережевого хробака або троянської програми проникнути в комп'ютер-жертву і самостійно запустити себе на виконання.

Уразливості є, по суті, помилками в коді або в логіці роботи різних програм. Сучасні операційні системи і додатки мають складну структуру і великий функціонал, і уникнути помилок при їх проектуванні та розробці просто неможливо. Цим і користуються комп'ютерні зловмисники.

Уразливими в поштових клієнтах Outlook користувалися поштові черв'яки Nimda і Aliz [34]. Для того щоб запустити файл хробака, досить було відкрити заражене лист або просто навести на нього курсор у вікні попереднього перегляду.

Також шкідливі програми активно використовували уразливості в мережевих компонентах операційних систем. Для свого поширення такими уразливими користувалися черви CodeRed, Sasser, Slammer, Lovesan (Blaster) і багато інших черви, що працюють під ОС Windows. Під удар потрапили і Linux-системи - черви Ramen і Slapper проникали на комп'ютери через уразливості в операційному середовищі та додатках для неї.

В останні роки одним з найбільш популярних способів зараження стало впровадження шкідливого коду через вебресурси. При цьому часто використовуються уразливості в інтернет-браузерах. На вебсторінку поміщається заражений файл і скрипт-програма, яка використовує уразливість в браузері. При заході користувача на заражену сторінку спрацьовує скрипт-програма, яка через уразливість закачує заражений файл на комп'ютер і запускає його там на виконання. В результаті для зараження великої кількості комп'ютерів досить заманити якомога більше число користувачів на таку веб-сторінку. Це досягається різними способами, наприклад, розсилкою спаму із зазначенням адреси сторінки, розсилкою аналогічних повідомлень через інтернет-пейджери, іноді для цього використовують навіть пошукові машини. На зараженій сторінці розміщується різноманітний текст, який рано чи пізно обраховується пошуковими машинами - і посилання на цю сторінку виявляється в списку інших сторінок в результатах пошуку.

Окремим класом є троянські програми, які призначені для завантаження та запуску інших троянських програм. Зазвичай ці дуже незначні за розміром трояни так чи інакше (наприклад, використовуючи іншу вразливість в системі) «проскакують» на комп’ютері жертви, а потім завантажують з Інтернету та встановлюють інші шкідливі компоненти. Часто такі трояни змінюють налаштування браузера на найбільш небезпечні, щоб «полегшити» роботу інших троянських програм.

Вразливості, які стають відомими, досить швидко виправляються компаніями-розробниками, але інформація про нові вразливості постійно з’являється, якими негайно користуються багато хакери. Багато троянських ботів використовують нові уразливості, щоб збільшити свою кількість, і нові помилки в Microsoft Office негайно починають використовуватися для впровадження звичайних троянських програм в комп'ютери. У той же час, на жаль, спостерігається тенденція до зменшення часового інтервалу між появою інформації про наступну вразливість та початком її використання хробаками та троянами. Як результат, вразливі компанії-розробники програмного забезпечення та розробники антивірусного програмного забезпечення потрапляють у ситуацію, що обмежується часом. Перший - це якомога швидше виправити помилку, протестувати результат (зазвичай його називають «патч», «патч») і надіслати його користувачам, а

другий - негайно випустити інструмент для виявлення та блокування об’єктів (файлів, мережі пакети), які використовують уразливість.

Одночасне використання технологій впровадження та методів соціальної інженерії [15]. Досить часто комп’ютерні зловмисники використовують одночасно обидва методи. Метод соціальної інженерії - для привернення уваги потенційної жертви, а технічний - для збільшення ймовірності проникнення зараженого об’єкта в систему.

Наприклад, поштовий черв'як Mimail поширювався як вкладення електронної пошти. Для того, щоб користувач звернув увагу на лист, в нього був вставлений спеціально відформатований текст, а для запуску копії хробака з прикріпленого до листа ZIP-архіву - уразливість в браузері Internet Explorer. В результаті, під час відкриття файлу з архіву, черв’як створив на диску власну копію та запустив її на виконання без будь-яких системних попереджень та додаткових дій користувача. До речі, цей хробак був одним із перших, призначених для викрадення особистої інформації користувачів електронних гаманців e-gold.

Інший приклад - надсилання спаму з темою "Привіт" та текстом "Подивіться, що про вас написано". За текстом було посилання на веб-сторінку. Аналіз показав, що ця веб-сторінка містить програму-сценарій, яка, використовуючи чергову уразливість в Internet Explorer, завантажує на комп'ютер користувача троянський LdPinch, призначений для викрадення різних паролів.

1.5 Постановка задачі

В першому розділі здійснено дослідження основних типів загрозливих програм, та запропоновано класифікацію ШПЗ за способом їх виконання. Враховуючи аналіз існуючої статистики зроблено висновок, що найбільш актуальними для захисту є виконувані двійкові і файли сценаріїв.

Можна виділити два найбільш поширених способи зараження: соціальна інженерія; технічні прийоми впровадження ШПЗ, що заражається без відома користувача.

Ці види ШПЗ передбачають обов'язкове збереження файлу на вінчестері перед виконанням.

Тому можна зробити висновок що застосування розмежувальної політики доступу до виконуваних об'єктів, дозволяє мінімізувати загрози.

В ході дослідження необхідно виконати наступні заддачі:

1. Проаналізовати основні типів шкідливого програмного забезпечення запропоновано їх класифікувати за способами завантаження і виконання шкідливих файлів.

2. Дослідити ефективність існуючих методів і засобів захисту, які базуються на сигнатурному і поведінковому аналізі.

3. Вдосконалити моделі і методи захисту від шкідливого програмного забезпечення на основі контролю доступу до файлів.

4. Зформулювати ряд вимог до побудови безпечної системи. які реалізують розроблені методи, виконання яких дозволяє будувати безпечну систему.

5. Розробити політики безпеки - варіанти налаштування засобів захисту від ШПЗ

6. Розробити функціональну схему, політику безпеки, направлених на захист інформаційної системи від шкідливого програмного забезпечення.

7. Провести оцінку ефективності запропонованих методів захисту з врахуванням впливу на завантаження обчислювальних ресурсів інформаційної системи.

2 МОДЕЛЮВАННЯ ПРОЦЕСУ ЗАХИСТУ НА ОСНОВІ КОНТРОЛЮ ДОСТУПУ ДО РЕСУРСІВ

2.1 Оцінка актуальності загрози запуску ШПЗ

Використання сучасних підходів до інформаційної безпеки вимагає, з одного боку, відстеження швидких змін у інформаційних технологіях та нових загроз, а з іншого - з урахуванням реальних характеристик апаратного та програмного забезпечення корпоративних мереж та систем. Процедура придбання пристроїв захисту інформації проста. Набагато складніше вирішити проблему - як захистити і які заходи безпеки застосовувати, мінімізуючи витрати. Впроваджуючи різні засоби захисту, необхідно врахувати баланс між можливим збитком від несанкціонованого витоку інформації і обсягом інвестицій, витрачених на забезпечення безпеки інформаційних ресурсів. З метою підвищення ефективності захисту інформаційних ресурсів необхідно дослідити підходи до оцінки рівня їх захисту та систем захисту. Ця оцінка індивідуальна для кожного конкретного випадку і залежить від багатьох факторів (вартості інформації, статусу організації, важливості інформації, рівня технічного та програмного забезпечення тощо).

Як правило, для оцінки рівня захисту спочатку необхідно визначити поточний стан інформаційної безпеки. Сьогодні існує два підходи до оцінки сучасного стану інформаційної безпеки, а саме "дослідження знизу вгору" та "дослідження зверху вниз".

Використовуючи перший підхід, адміністратори починають перевіряти систему безпеки на всі відомі типи атак. Таким чином, адміністратори виступають як зловмисники, які намагаються порушити захист інформаційних ресурсів.

Підхід зверху вниз ґрунтується на детальному аналізі всіх відомих схем зберігання та обробки даних. Спочатку визначаються інформаційні об’єкти та потоки захисту, а потім досліджується сучасний стан систем захисту інформації з метою визначення впроваджених методів захисту інформаційних ресурсів, а також їх стану та рівня. Тоді всі інформаційні об'єкти класифікуються за класами відповідно до їх вимог щодо конфіденційності, доступності та цілісності.

Останнім кроком є "оцінка ризику", яка полягає у визначенні розміру збитків фірми внаслідок порушення захисту кожного конкретного інформаційного ресурсу. Приблизний ризик - це результат "можливого збитку від нападу" до "ймовірності цього нападу". Як правило, оцінка ризику складається з аналізу ризику та оцінки збитків.

Всі сучасні стандарти в галузі безпеки відображають загальний підхід до організації управління ризиками, сформований у міжнародній практиці. Управління ризиками розглядається як основна частина системи управління якістю організації. Стандарти є відверто концептуальними, що дозволяє експертам з інформаційної безпеки використовувати будь-які методи, засоби та технології для оцінки, тестування та управління ризиками. Різні стандарти дозволяють використовувати якісні та кількісні методи оцінки ризиків інформаційної безпеки, але немає рекомендацій та обґрунтування щодо вибору математичного та методологічного апарату. Додаток до стандарту [13] наводить приклад якісного методу оцінки, а саме використання три- та п’ятибальної рейтингової шкали. За п’ятибальною шкалою рівні витрат ідентифікованого ресурсу оцінюються як: „незначний”, „низький”, „середній”, „високий”, „дуже високий”. За племінною шкалою - як «низький», «середній», «високий».

Спільні критерії оцінки безпеки повинні застосовуватися на єдиній загальній методологічній основі, заснованій на синтезі заходів безпеки, інструментів та послуг для мінімізації інформаційних ризиків. Експерти, розробники та клієнти використовують загальну методологію оцінки інформаційної безпеки для оцінки та моніторингу ресурсів інформаційної безпеки [9].

З точки зору розробника профілю захисту, застосування загальної методології дає можливість проводити її незалежну та послідовну оцінку та обґрунтування.

Застосування загальної методології дозволить розробнику:

• самостійно обґрунтувати та перевірити показники захисту кожного інформаційного ресурсу, задокументованого у профілі та проекті;

• переконати споживача в тому, що об'єкт оцінки відповідає необхідним показникам безпеки;

• ефективно використовувати результати, отримані під час оцінки інших продуктів та систем, для побудови систем безпеки;

• зменшити витрати часу та матеріальних ресурсів в процесі оцінки безпеки системи.

Відповідно до загальної методології оцінки інформаційної безпеки вона повинна проводитися у три етапи: підготовчий, базовий, заключний.

На підготовчому етапі головними дійовими особами є замовник оцінки і експерт. Замовник інформує всі сторони про необхідність оцінки профілю захисту або об'єкта оцінки, надає експерту необхідну документацію, матеріали щодо профілю захисту та об'єкта оцінки. Завдання експерта - визначити можливість успішної оцінки на основі отриманих матеріалів, а при необхідності вимагати додаткові матеріали у замовника або розробника. Результатом підготовчого етапу є укладення між замовником та експертом договору про виконання робіт з оцінки об'єкта або профілю охорони.

Результатом основного етапу є розробка та подання експертом звіту про технічну оцінку, який містить обґрунтування рішення. На основному етапі експерт вивчає подані йому матеріали, профіль захисту або об’єкт оцінки. Експерт готує низку звітів з вимогами для надання пояснень щодо вимог контролюючого органу, виявлених недоліків та іншої інформації про хід оцінки. Контролюючий орган здійснює постійний моніторинг процесу оцінювання відповідно до схеми оцінки.

Заключним кроком є всебічний аналіз звіту про технічну оцінку інспекційним органом на відповідність загальним критеріям загальної методології і вимогам схем оцінки безпеки. На підставі технічного звіту формується остаточний звіт про оцінку з рішенням про відповідність необхідним вимогам. Усі сторони, які беруть участь у процесі оцінки, вивчають остаточний звіт і мають право вимагати відповідних пояснень.

Багато організацій з різних причин не здатні повністю оцінити захист інформаційних ресурсів, тому пропонується використовувати кількісну оцінку рівня безпеки. Його використання можливо на етапі реалізації. В результаті застосування кількісної оцінки можна більш точно порівняти кілька варіантів захисту, що дозволяє вибрати найбільш ефективний. Для його використання визначають ймовірність загроз та вразливості інформаційних ресурсів, вартість захищених ресурсів (оцінка збитків у разі виходу з ладу інформаційного ресурсу) і частоту загроз кожного типу в загальному потоці загроз. Обов’язковим є встановлення обмежень на вартість системи захисту інформації і зменшення рівня продуктивності системи.

Щоб провести оцінку безпеки, ми рекомендуємо вам виконати наведені нижче дії.

• на першому етапі скласти перелік загроз з позицій інформаційної безпеки, визначити ймовірність загроз та ймовірність їх відображення за системою захисту, вартість інформаційних ресурсів;

• на другому етапі накладаються обмеження на вартість системи захисту інформації і на зниження рівня швидкодії комп'ютерної інформаційної системи;

• на третьому етапі розраховується оцінка за математичними формулами загального рівня безпеки комп'ютерної інформаційної системи обраними засобами;

• На четвертому етапі з розглянутих та оцінених варіантів виберіть той, який найкраще відповідає вимогам і не перевищує встановлених меж.

Насправді рівень захисту визначається як відношення ризиків в захищеній системі до ризиків в незахищеній системі. Цей підхід дозволяє точніше описати інформаційні ресурси через властиві їм вразливості, вартість самих ресурсів, ранжувати ризики і, відповідно, інформаційні ресурси відповідно до ступеня критичності для організації.

Для отримання оцінки актуальності впровадження та запуску ШПЗ в інформаційній системі, застосуємо математичну модель, яка базується на зображенні атаки як послідовності загроз на орієнтованому графі (рис. 2.1).

Використаємо наступні позначення: *З* - зловмисник; *Зі*, де *і* від 1 до *n* – імовірність реалізація загрози; $Ц_1$ – втрата конфіденційності; $Ц_2$ – втрата цілісності; $Ц_3$ – втрата доступності.

Приймемо імовірність захищенності від загрози за $p_{0Зi}$. Тоді імовірність реалізації відповідної загрози 1- $p_{0Зi}$.

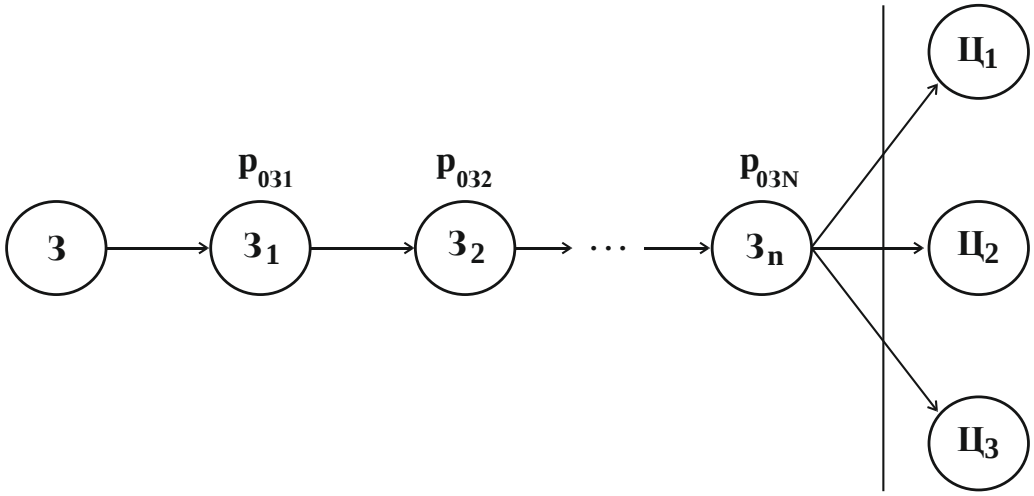


Рисунок 2.1 - Представлення атаки як послідовності загроз

Згідно теорії надійності, враховуючи необхідність реалізації всієї послідовність загроз, схема прийме вигляд (рис. 2.2).

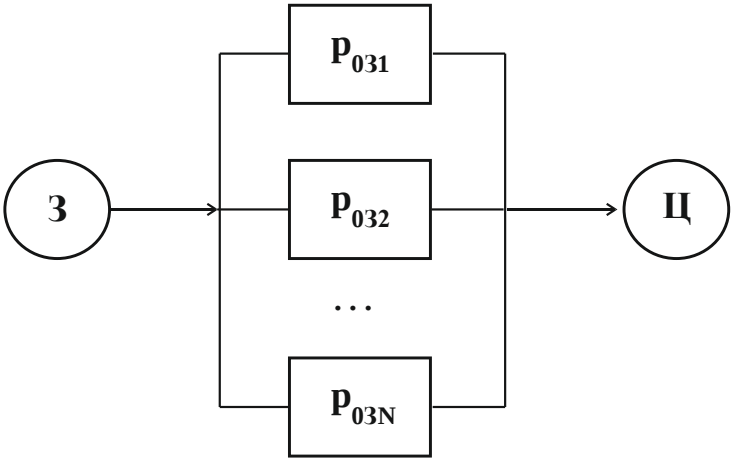


Рисунок 2.2 - Паралельне резервування

Тоді імовірність здійснення атаки розрахуємо по формулі:

$$p_{0здійсн} = \prod_{i=1}^{n} (1 - p_{0i}) \qquad (2.1)$$

Модель захищеної системи, з доданим засобом захисту зображено на рисунку 2.3.

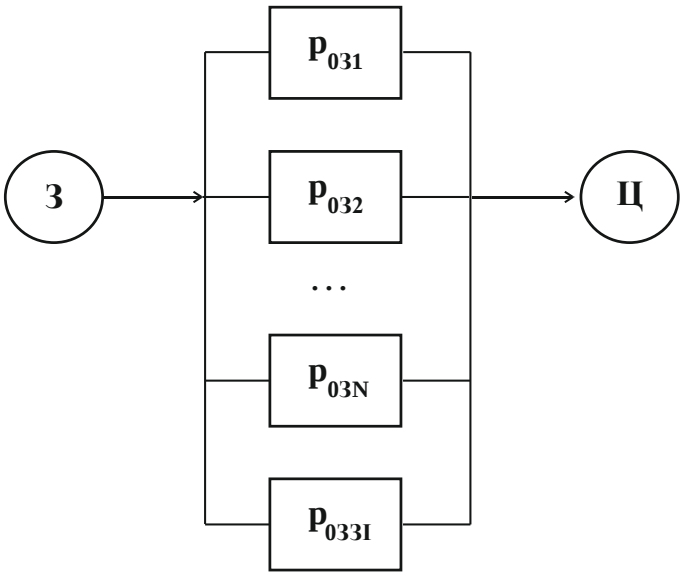


Рисунок 2.3 - Паралельного резервування з включенням засобу захисту

Тоді імовірність здійснення атаки розрахуємо по формулі:

$$p_{0здійсн} = (1 - p_{0ЗЗІ}) \cdot \prod_{i=1}^{n} (1 - p_{0i}) \qquad (2.2)$$

Бачимо, що місце розташування ЗЗІ не впливає на захищеність інформаційної системи. Запропонуємо критерій актуальності загрози, який обраховується по формулі:

$$K_a = U \cdot S \qquad (2.3)$$

де, U - число атак, які входять в вершину; S - число атак, що виходять з вершини на графі. Враховуючи статистику провідних компаній в сфері ІБ ESET, Cisco, найчастіше при атаках використовувалося ШПЗ. Сформуємо відповідний графі (рис. 2.4).

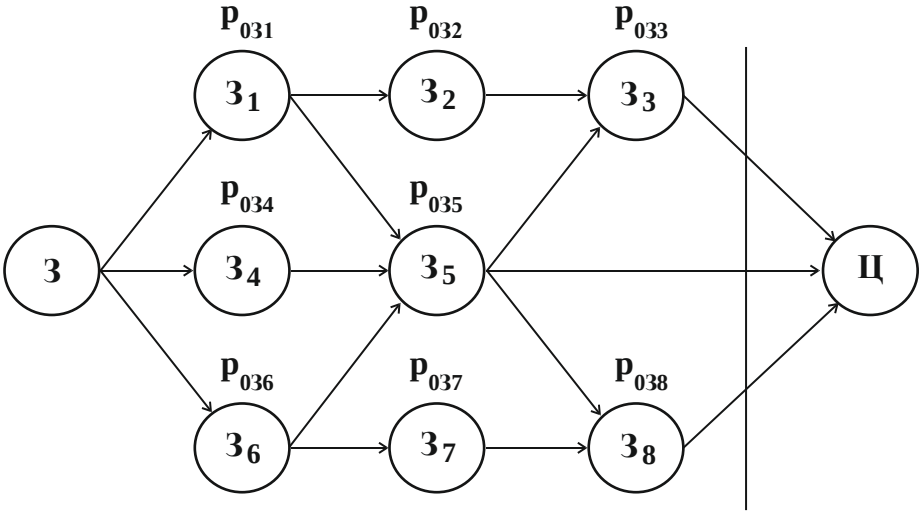


Рисунок 2.4 – Кількість атак, які використовували однакове ШПЗ

де, *З* - зловмисник; *Зі*, де *і* від 1 до *n* – імовірність реалізація загрози;; $З_1$ - небезпека взлому ресурсу; $З_2$ - небезпека взлому паролів; $З_3$ – небезпека підміни файлів ОС; $З_4$ – небезпека, пов'язана з соціальною інженерією; $З_5$ - небезпека застосування ШПЗ; $З_6$ - зміна привілеїв; $З_7$ – небезпека взлому протоколу мережевої взаємодії; $З_8$ - небезпека вразливості в мережі; *Ц* - цілі: втрата конфіденційності; втрата цілісності; втрата доступності. Згідно формули 2.3 найбільш актуальною є загроза, яка має найбільшу кількість переходів і з поточного стану.

2.2 Моделювання підходу до захисту від ШПЗ на основі контролю доступу до ресурсів

На основі запропонованої в першому розділі класифікації ШПЗ, було зроблено висновок про можливість реалізації захисту на основі розмежування доступу до ресурсів, зокрема, до виконуваних файлів., відповідно до політики безпеки. Політика інформаційної безпеки - це сукупність вимог, правил, обмежень, рекомендацій, що регулюють порядок інформаційної діяльності в організації та спрямовані на досягнення та підтримання стану інформаційної безпеки організації.

Основною причиною запровадження політики безпеки є, як правило, вимога мати такий документ від регулятора - організації, яка визначає правила роботи підприємств цієї галузі. У цьому випадку відсутність політики може призвести до репресивних дій проти компанії або навіть до повного припинення її діяльності.

Крім того, певні вимоги (рекомендації) встановлюються галузевими або загальними, місцевими або міжнародними стандартами. Зазвичай це виражається у формі коментарів зовнішніх аудиторів, які проводять перевірки підприємства. Відсутність політики викликає негативну оцінку, що, в свою чергу, впливає на результати діяльності компанії в суспільстві - позицію в рейтингу, рівень надійності тощо.

Цікаво, що згідно з дослідженням безпеки, проведеним Deloitte в 2006 році, компанії з офіційною політикою інформаційної безпеки набагато рідше піддаються злому. Це наводить на думку про те, що наявність політики - це ознака зрілості підприємства у питаннях інформаційної безпеки. Той факт, що компанія чітко сформулювала свої принципи та підходи до інформаційної безпеки, означає, що в цьому напрямку була проведена серйозна робота.

Розглядаючи інформаційну безпеку в автоматизованих системах, завжди кажуть, що існують деякі "бажані" стани системи. Ці бажані стани (які зазвичай представлені з точки зору моделі самого динаміка) описують "безпеку" системи. Поняття "безпека" принципово не відрізняється від інших особливостей технічної системи, таких як "надійна робота". Особливістю поняття "безпека" є його тісний зв'язок з поняттям "загроза" (наслідком чого може бути виведення системи із захищеного стану).

Отже, існує три компоненти, пов’язані з порушенням безпеки системи:

• "загроза" - зовнішнє порушення властивості "безпеки" щодо вихідної системи;

• "об'єкт атаки" - частина системи, на яку впливає загроза;

• "канал дії" - засіб передачі шкідливих дій.

Невід'ємною характеристикою, яка враховує всі ці компоненти, є політика безпеки - якісне (або якісно-кількісне) вираження властивостей безпеки в термінах, що представляють систему. Опис політики безпеки повинен включати або враховувати властивості загрози, об'єкт атаки та канал дії.

Розробка і підтримка політики безпеки практично завжди означає досягнення компромісу між альтернативами, що обирають власники цінної інформації для її захисту. Тому, будучи результатом компромісу, політики безпеки ніколи не задовольнить усі сторони, які беруть участь у захисті інформації. Але вибір політики безпеки - це остаточне рішення: що погано й що добре в поводженні з цінною інформацією. Після прийняття такого рішення є можливість будувати захист, тобто певну систему підтримки виконання правил ПБ.

В роботі пропонується використати дискреційну модель розмежування доступу. Основою *дискреційної політики безпеки* (ДПБ) є дискреційне керування доступом (Discretionary Access Control - DAC), яке визначається такими властивостями [24]:

- усі об'єкти й суб'єкти мають бути однозначно ідентифіковані;
- рівні доступу суб'єкта до об'єкта системи мають визначаютися на основі певних зовнішніх правил.

ДПБ реалізується за рахунок матриці доступу, яка регламентує множину об'єктів та суб'єктів, наданих кожному суб'єкту.

Наведемо приклади варіантів створення матриці доступу.

1. Листи можливостей: для кожного користувача створюється лист (файл) усіх об'єктів, до яких йому надано доступ.
2. Листи контролю доступу: для кожного користувача створюється список усіх суб'єктів, які мають доступи до цього користувача.

До переваг такої політики можна віднести відносно просту реалізацію необхідних механізмів захисту. Саме цим зумовлено те, що переважна більшість поширених нині захищених систем забезпечують виконання положень саме ДПБ.

Модель Харрісона-Руззо-Ульмана є класичною дискреційною моделлю, реалізує довільне керування доступом суб'єктів до об'єктів та контроль за розподіл прав доступу в рамках цієї моделі.

В рамках запропонованого підходу модель складається з наступних елементів:

- кінцева множина вихідних суб'єктів (сукупності активних сутностей) $S=\{S_1, \dots, S_n\}$;

- кінцева множина вихідних об'єктів (пасивних сутностей) $O=\{O_1, \dots, O_n\}$;

- кінцевої множини прав доступу $R-\{r_1, \dots, r_n\}$

- матриця доступу *M,* в якій записані права і правила доступу, в кожній комірці міститься набір прав суб'єкта *S* до об'єкта *O*

Модель безпеки HRU (Харрісона-Руззо-Ульмана) реалізує довільне управління доступом суб'єктів до об'єктів та контроль за поширенням прав доступу.

позначимо:

S - множина суб'єктів системи (активні сутності);

R - кінцева множина прав доступу суб'єктів до об'єктів;

M = {r1, ..., rn} - повноваження на виконання відповідних дій

Щоб включити в область дії моделі і відносини між суб'єктами, прийнято вважати, що всі суб'єкти є об'єктами.

Стан системи характеризується трійкою (S, O, M), де S - множина суб'єктів, O - множина об'єктів (в Про можуть входити і суб'єкти), M - матриця доступу. Матриця M включає по одному рядку для кожного суб'єкта і по одному стовпцю для кожного об'єкта Осередок такої матриці M [s, o] містить права доступу суб'єкта s до об'єкта o. Права доступу входять в кінцевий набір прав R.

2.3 Модель захисту з використанням матриці розмежування доступу

Бінарні та скриптові виконувані файли будуть діяти як об'єкти доступу. Розглянемо варіант, коли всі користувачі мають однакові права доступу.

Перш ніж стверджувати, що виконуваний файл програми може мати розширення лише одного конкретного типу, необхідно зрозуміти, як відрізнити такий об’єкт від інших. Основні відмінності між виконуваними файлами та іншими інформаційними даними включають такі фактори: саме розширення, яке вказує вміст файлу чи машини, або байт-код віртуальної машини, підпис, атрибути у файловій системі..

Що стосується побудови виконуваних файлів, вони повинні містити заголовки (передбачуване виконання інструкцій, параметрів та форматів коду) та самі інструкції (джерело, машина або байт-коди). У деяких випадках структура може включати дані налагодження, опис середовища, вимоги до операційної системи, списки відповідних бібліотек, звук, графіку, зображення, піктограми ярликів тощо [31]. Багато з вас, напевно, помічали, що здебільшого кожен такий файл в операційній системі спочатку має піктограму.

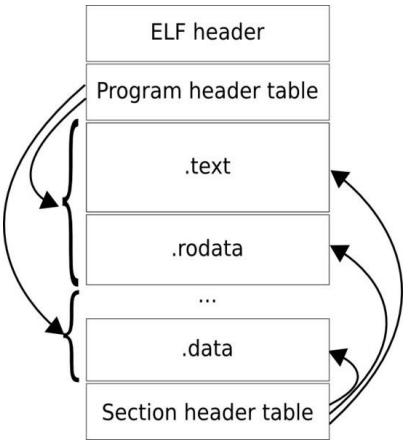


Рисунок 2.5 – Структура виконуваного файлу

Хоча виконувані файли можуть мати розширення різних типів, вони працюють за одним принципом. Виконавчий файл завантажується в пам'ять комп'ютера під час запуску. Також на цьому етапі відбувається налаштування деяких додаткових операцій та виконання інструкцій тими методами, які реєструються безпосередньо у файлі.

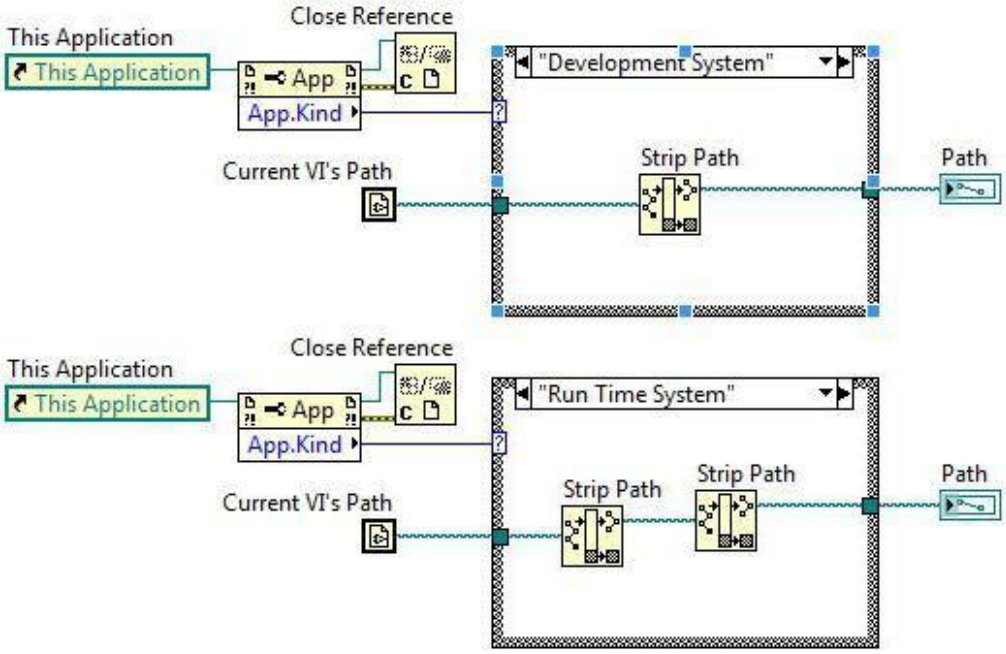


Рисунок 2.6 – Структура виконуваного файлу

У Windows найпоширеніші двійкові виконувані файли. Найпоширеніший їх тип - аплікація. Додатки мають розширення EXE і можуть працювати самостійно. Крім того, існують динамічні бібліотеки (їх розширення - DLL), які містять загальні функції для різних додатків. Існують також драйвери (DRV або VXD) - спеціальні програми, необхідні для того, щоб система могла взаємодіяти з конкретними моделями певних пристроїв. Виконувані файли (особливо в Windows) можуть залежати один від одного: наприклад, для запуску будь-якої програми потрібні певні системні динамічні бібліотеки, а вони, в свою чергу, потребують драйверів.

Слід зазначити, що виконувані файли містять не тільки самі програми, але і різні додаткові дані. Це можуть бути різні графічні ресурси, що відображаються програмою, тексти написів, описи діалогових вікон тощо. Яскравим прикладом цього є архіви, які містять великі обсяги упаковки з метою зменшення її обсягу при передачі або зберіганні інформації.

Отже суб'єктом доступу є будь-який користувач системи $S_i : S = \{S_1, \dots, S_k\}$. Об'єкти доступу поділяються по їх розширенням на виконувані, системні та інформаційні: $O = \{O_{вик1}, \dots, O_{викq}, O_{сист1}, \dots, O_{системт}, O_{інф1}, \dots, O_{інфп}\}$.

Системні файли мають розширення: *.config, *.manifest, *.fon, *.ttf, *.log. Множина прав доступу містить операції читання, виконання і запис:

$$R = \{Чт, В, Зп\}.$$

Враховуючи, що використовується дозвільна політика доступу, тобто не дозволено все що явно не дозволено. Згідно з основною ідеєю пропонованого підходу захисту від загрозливого ПЗ, пропонований метод полягає в наступному:

- виконувані файли мають атрибути читання і виконання;
- системні мають атрибути тільки читання;
- інформаційних мають атрибути читання і запис;
- все інше забороняється.

Враховуючи відповідні дозволи отримаємо наступну матрицю доступу:

$$M=\begin{array}{c} \\ S_1 \\ \dots \\ \dots \\ S_k \end{array}\begin{array}{c} \begin{array}{ccc} O_{вик1}, \dots, O_{викq} & O_{сист1}, \dots, O_{систm} & O_{інф1}, \dots, O_{інфn} \end{array} \\ \begin{bmatrix} Чт, В & Чт & Чт, Зп \\ \dots & \dots & \\ \dots & \dots & \\ Чт, В & Чт & Чт, Зп \end{bmatrix} \end{array}$$

Оскільки послідовність дій, в результаті яких відбудеться витік повноважень «Запис» і «Виконання» генерує користувач, то його дії не зможуть привести до витоку права доступу. В тому числі «Власник», який створив новий об'єкт, не зможе наділити правом доступу «Запис» інших користувачів, бо в даній політиці виключена сама сутність «Власник». Перевагою даного підходу є те, що неможливо зберегти або запустити ШПЗ.

Недоліком данної політики є можливість запускати всі виконувані файли, незалежно від їх розташування, і Адміністратор системи не зможе інсталювати програмне забезпечення, адже він має права, однакові з іншими користувачами.

Введемо обмеження для розташування виконуваних файлів:

- системні каталоги «%ProgramFiles%» і «%windir%»;

- системний диск, наприклад, «C:\».

Для врахування зазначенних недоліків побудуємо матрицю доступу з врахуванням повноважень Адміністратора.

Наділимо його правами:

- читання, інсталювання і запуск виконуваних файлів;

- читання, інсталювання системних файлів;

- читання і запис інформаційних файлів;

- все інше забороняти.

Опишемо права інших користувачів:

- читання і запуск виконуваних файлів, які вже знаходяться на системному диску;

- читання системних файлів;

- читання і запис інформаційних файлів;

- все інше забороняти.

Матриця доступу буде мати вигляд:

$$M = \begin{array}{c} \\ S_1 \\ \dots \\ \dots \\ S_k \\ A \end{array} \begin{array}{c} \begin{array}{ccc} O_{вик1}, \dots, O_{викq} & O_{сист1}, \dots, O_{системm} & O_{інф1}, \dots, O_{інфn} \end{array} \\ \begin{bmatrix} Чт, В & Чт & Чт, Зп \\ \dots & \dots & \\ \dots & \dots & \\ Чт, В & Чт & Чт, Зп \\ Чт, Зп, В & Чт, Зп & Чт, Зп \end{bmatrix} \end{array}$$

Зі збільшенням повноважень звичайного користувача до Адміністратора можливий витік права щодо суб'єкта доступу.

Але ця матриця доступу враховує можливість адміністратора інсталювати нове ПЗ.

Недоліком цього підходу є можливість отримання прав адміністратора, що викликає необхідність додаткового захисту від підвищення повноважень до адміністративних прав.

Ідентифікатор безпеки (*Security Identifier*, SID) у операційних системах Windows) — це унікальний незмінний ідентифікатор об’єкту, групи користувачів або інших захищених елементів [16]. Вони мають один SID протягом всього циклу функціонування, і всі властивості цього елемента, включаючи також його назву (ім'я), пов'язані з SID. Такий підхід дозволяє, наприклад, перейменовувати елементи без впливу на атрибути безпеки об'єктів, що відносяться до нього.

Windows використовує SID для здійснення ідентифікації машин та всіх інших елементів системи безпеки: користувачів і груп, облікових записів домену. Імена таких об'єктів є тільки зрозумілішими для користувачів формами подання SID.

Коли Windows посилається на загальну назву, як ми робимо, замість SID, тоді все, що пов'язано з цим ім'ям, буде недійсним або недоступним, коли ім'я було змінено будь-яким способом.

Тому замість того, щоб змінити ім’я вашого облікового запису було неможливо, акаунт користувача посилається до незмінного рядка (SID), що дозволяє змінювати назву, не впливаючи на жодні налаштування системи.

Хоча ім'я користувача може бути змінено багато разів, ви не можете змінити SID, який пов'язаний з обліковим записом Тому не потрібно вручну оновлювати всі

налаштування безпеки, пов'язані саме з цим користувачем, щоб відновити його ідентифікацію.

Облікові записи SID починаються з S-1-5-21 а все інше буде унікальним. Кілька SID можуть бути декодовані без інструкцій. Для прикладу, SID для облікового запису Адміністратор у Windows завжди закінчується 500. SID. Обліковий запис Гість завжди закінчується 501.

Ви також визначите SID на кожній установці Windows, що відповідає певним вбудованим обліковим записам.

Наприклад, SID S-1-5-18 є в будь-якій копії Windows, які існують і відповідає LocalSystem обліковий запис, тобто системному обліковому запису, завантаженому в Windows перед входом користувача.

Хоча більшість розмов про SID відбуваються в контексті підвищення інформаційної безпеки, більшість згадок стосуються реєстру Windows і як інформація про конфігурацію користувача зберігаються в певних розділах реєстру, що називаються такими самими, як і SID конкретного користувача.

Отже суб'єкт доступу потрібно задавати виходячи з двох сутностей: ефективного та первинного користувача, і заборони надавати користувачу прав Адміністратора.

Для вирішення проблеми проникнення ШПЗ під виглядом санкціонованого типу файлів пропонується розділяти способи модифікації об'єктів. Наступний метод враховує способи модифікації ОД.

Відмінними моментами від попереднього методу є:

- врахування заборони модифікації об'єктів шляхом перейменування (позначимо «*~~П~~*»);

- врахування заборони модифікації об'єктів шляхом видалення виконуваного файлу (позначимо «*~~Д~~*»)

Заборону запису позначимо «*~~Зп~~*».

Тоді в моделі будемо використовувати такий набір прав:

$$R = \{Чт, В, Зп, \sout{Зп}, \sout{П}, \sout{Д}\}.$$

Для врахування зазначених недоліків побудуємо матрицю доступу з врахуванням повноважень Адміністратора.

Наділимо його правами:

- читання, інсталювання і запуск виконуваних файлів;
- читання, інсталювання системних файлів;
- читання і запис інформаційних файлів;
- все інше забороняти.

Опишемо права інших користувачів:

- читання і запуск виконуваних файлів, які вже знаходяться на системному диску;
- читання системних файлів;
- читання і запис інформаційних файлів;
- заборонити запис, перейменування і видалення системних файлі;
- заборонити перейменування і видалення інформаційних файлів;
- все інше заборонити.

Матриця доступу буде мати вигляд:

$$M = \begin{array}{c} \\ S_1 \\ \dots \\ \cdots \\ S_k \\ A \end{array} \begin{array}{c} \begin{array}{ccc} O_{вик1}, \dots, O_{викq} & O_{сист1}, \dots, O_{систm} & O_{інф1}, \dots, O_{інфn} \end{array} \\ \begin{bmatrix} Чт, В, \sout{Зп}, \sout{П}, \sout{Д} & Чт, \sout{Зп}, \sout{П}, \sout{Д} & Чт, Зп, \sout{П}, \sout{Д} \\ \dots & \dots & \\ \cdots & \cdots & \\ Чт, В, \sout{Зп}, \sout{П}, \sout{Д} & Чт, \sout{Зп}, \sout{П}, \sout{Д} & Чт, Зп, \sout{П}, \sout{Д} \\ Чт, Зп, В & Чт, Зп & Чт, Зп \end{bmatrix} \end{array}$$

Перевагою даного підходу є захист від впровадження шкідливої програми під виглядом санкціонованої.

Цей метод не враховується різниці між перейменуванням існуючого виконуваного файлу і інформаційного файлу. Для усунення даного недоліку в ще додамо заборону перейменування існуючого виконуваного файлу («~~Пв~~»), заборона перейменування в виконуваний файл («~~ПвВ~~»).

Тоді в моделі будемо використовувати таку множину прав:

$$R = \{Чт, В, Зп, \sout{Зп}, \sout{Пв}, \sout{ПвВ}, \sout{Д}\}.$$

Права адміністратора:

- читання, інсталювання і запуск виконуваних файлів;

- читання, інсталювання системних файлів;

- читання і запис інформаційних файлів;

- все інше забороняти.

Опишемо права інших користувачів:

- читання і запуск виконуваних файлів, які вже знаходяться на системному диску;

- читання системних файлів;

- читання і запис інформаційних файлів;

- заборонити запис, перейменування і видалення системних файлі;

- заборонити перейменування і видалення інформаційних файлів;

- заборонити для інформаційних файлів перейменування існуючого файлу, видалення;

- все інше заборонити.

Матриця доступу буде мати вигляд:

$$M = \begin{array}{c} \\ S_1 \\ \dots \\ \dots \\ S_k \\ A \end{array} \begin{array}{c} \begin{array}{ccc} O_{вик1}, \dots, O_{викq} & O_{сист1}, \dots, O_{систm} & O_{інф1}, \dots, O_{інфn} \end{array} \\ \begin{bmatrix} Чт,В,\sout{Зп,Пв,ПвВ,Д} & Чт,\sout{Зп,Пв,ПвВ,Д} & Чт,Зп,\sout{Пв,ПвВ,Д} \\ & \dots & \\ & \dots & \\ Чт,В,\sout{Зп,Пв,ПвВ,Д} & Чт,\sout{Зп,Пв,ПвВ,Д} & Чт,Зп,\sout{Пв,ПвВ,Д} \\ Чт,Зп,В & Чт,Зп & Чт,Зп \end{bmatrix} \end{array}$$

При такому підході також неможливий витік повноважень доступу «Запис», так як розмежування застосовні до всіх користувачів і не мають значення ні його ім'я, ні його SID. Також передбачається додатковий контроль перейменування об'єктів доступу.

Недоліком даного підходу ж те, що не враховується особливість впровадження шкідливого файлу-сценарію.

Вид виконуваного файлу залежить від вибраної мови [29]:

- виконуваний файл з інтегрованим інтерпретатором скриптової мови;
- бібліотека, функції C/C++ якої безпосередньо стають доступні з іншої мови;

- бібліотека функцій C/C++ і бібліотека функцій - надбудов над функціями C/C++ (наприклад, JNI для Java та інші).

Двигуни скриптових мов інтегруються в програми C/C++ через наступні причинами:

- швидкість проектування з використанням скриптової мови вище, ніж проектування на C/C++;
- користувачі мають можливість автоматизації певних дій завдяки сценаріям. Наприклад, в комп'ютерних іграх сценарії використовуються для написання сюжету та рівнів додатку;
- розробники мають можливість автоматизації та тестування ПО на етапі розробки. Готове програмне забезпечення може не включати скриптовий двигун.

Причини створення бібліотек модулів C/C++, доступних інтерпретаторам інших мов:

- забезпечення функціональності, яка відсутня в скриптових мовах;
- оптимізація найбільш виконуваних ділянок коду для підвищення продуктивності.

Ми додамо захист від виконуваних файлів сценарію. Для їх виконання потрібно встановити інший інтерпретатор, в якому запуститься виконуваний файл. Основою моделі захисту від виконуваних файлів сценарію є захист від витоку повноважень на зміну функцій дозволеної програми. Особливістю методу є поділ методу доступу "Запис" для створення нового об'єкта доступу («*~~ЗпН~~*») та зміни існуючого об'єкта доступу шляхом перейменування («*~~ЗпІ~~*»).

Тоді в моделі будемо використовувати таку множину прав:

$$R = \{Чт,\ В,\ Зп,\ \sout{ЗпН},\ \sout{ЗпІ},\ \sout{Пв},\ \sout{ПвВ},\ \sout{Д}\}.$$

Права адміністратора:

- читання, інсталювання і запуск виконуваних файлів;
- читання, інсталювання системних файлів;
- читання і запис інформаційних файлів;
- все інше забороняти.

Опишемо права інших користувачів:

- читання і запуск виконуваних файлів, які вже знаходяться на системному диску;

- заборонити створення нового файлу, зміни існуючого, перейменування існуючого виконуваного файлу, видалення існуючих виконуваних файлів на системному диску;

- читання системних файлів;

- заборонити для системних файлів створення нового, зміну та перейменування існуючого файлу;

- читання і запис інформаційних файлів;

- заборонити запис, перейменування і видалення системних файлі;

- заборонити перейменування і видалення інформаційних файлів;

- заборонити для інформаційних файлів перейменування існуючого файлу, видалення;

- все інше заборонити.

-

Матриця доступу буде мати вигляд:

	$O_{вик1}, \dots, O_{викq}$	$O_{сист1}, \dots, O_{системm}$	$O_{інф1}, \dots, O_{інфn}$
S_1	*Чт, В, ~~ЗпН~~, ~~ЗпІ~~,* *~~Пв~~, ~~ПвВ~~, ~~Д~~*	*Чт, ЗпН, ЗпІ,* *~~Пв~~, ~~ПвВ~~, ~~Д~~*	*Чт, ЗпН, ЗпІ,* *~~Пв~~, ~~ПвВ~~, ~~Д~~*
…	…	…	…
S_k	*Чт, В, ~~ЗпН~~, ~~ЗпІ~~,* *~~Пв~~, ~~ПвВ~~, ~~Д~~*	*Чт, ЗпН, ЗпІ,* *~~Пв~~, ~~ПвВ~~, ~~Д~~*	*Чт, ЗпН, ЗпІ,* *~~Пв~~, ~~ПвВ~~, ~~Д~~*
А	*Чт,Зп, В*	*Чт,Зп*	*Чт, Зп*

M=

У цій моделі дозволена програма наділяється загрозливими можливостями, такими як витік повноважень модифікації функцій в процесі запуску невиконуваних файлів («*~~ЗпН~~*» і «*~~ЗпІ~~»). Перевагою даного підходу є контроль впровадження загрозливих виконуваних файлів-сценаріїв. Але відсутній контроль роботи браузера з інтерпретаторами (віртуальними машинами).

Введемо додатково право доступу - заборона читання («*~~Чт~~*»).

Тоді в моделі будемо використовувати таку множину прав:

$$R = \{Чт,\ \sout{Чт}\ ,\ В,\ Зп,\ \sout{ЗпН},\ \sout{ЗпІ},\ \sout{Пв},\ \sout{ПвВ},\ \sout{Д}\}.$$

Веедемо права доступу для суб'єкту інтерпретатор:

- читання і запуск виконуваних файлів, які вже знаходяться на системному диску;

- заборонити читання, запис, перейменування існуючого виконуваного файлу, видалення існуючих виконуваних ОД на системному диску;

- дозволити тільки читання для системних файлів;

- -дозволити тільки читання і запис інформаційних файлів;

- - все інше заборонити.

Матриця доступу буде мати вигляд:

		$O_{вик1}, \dots, O_{викq}$	$O_{сист1}, \dots, O_{систm}$	$O_{інф1}, \dots, O_{інфn}$
M =	S_1	*Чт, В, ~~ЗпН~~, ~~ЗпІ~~,*	*Чт, ~~ЗпН~~, ~~ЗпІ~~,*	*Чт, Зп,*
	…	*~~Пв~~, ~~ПвВ~~, ~~Д~~*	*~~Пв~~, ~~ПвВ~~, ~~Д~~*	*~~Пв~~, ~~ПвВ~~, ~~Д~~*
	…		…	
	S_k	*Чт, В, ~~ЗпН~~, ~~ЗпІ~~,*	*Чт, ~~ЗпН~~, ~~ЗпІ~~,*	*Чт, Зп,*
		~~Пв~~, ~~ПвВ~~, ~~Д~~	*~~Пв~~, ~~ПвВ~~, ~~Д~~*	*~~Пв~~, ~~ПвВ~~, ~~Д~~*
	VM_1	*~~Чт~~, В, ~~ЗпН~~, ~~ЗпІ~~,*	*Чт, ~~ЗпН~~, ~~ЗпІ~~,*	*Чт, Зп,*
	…	*~~Пв~~, ~~ПвВ~~, ~~Д~~*	*~~Пв~~, ~~ПвВ~~, ~~Д~~*	*~~Пв~~, ~~ПвВ~~, ~~Д~~*
	…		…	
	VM_j	*~~Чт~~, В, ~~ЗпН~~, ~~ЗпІ~~,*	*Чт, ~~ЗпН~~, ~~ЗпІ~~,*	*Чт, Зп,*
		~~Пв~~, ~~ПвВ~~, ~~Д~~	*~~Пв~~, ~~ПвВ~~, ~~Д~~*	*~~Пв~~, ~~ПвВ~~, ~~Д~~*
	A	*Чт,Зп, В*	*Чт,Зп*	*Чт, Зп*

Такий підхід дає нам впевненість що такі права доступу не дадуть змогу бінарним і скриптовим виконуваним загрозливим файлам зашкодити безпеці системи.

Але потрібно забороняти співпадання будь-якого користувача з Адміністратором.

2.4 Висновки

У розділі запропоновано підхід до захисту від ШПЗ, який базується на контролі доступу до ресурсів за допомогою розширень та типів файлів.

Розроблено підходи до захисту від бінарних файлів та файлів, які містять сценарії.

Запропоновано модель захищеної системи, яка дає можливість сформувати вимоги з точки зору запобігання витоку прав доступу.

Реалізація запронованих моделей дозволить побудувати захищену систему.

3 РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЙ-НИХ РЕСУРСІВ

3.1 Розробка вимог, для побудови безпечної системи

DBFA від Lengyel, Maresca та інших. [27] - це інструмент для автоматичної автентифікації віртуальної машини, розроблений спеціально для аналізу шкідливих програм Windows. Технічно DBFA базується на широкому застосуванні LibVMI і вимагає використання гіпервізора Xen для віртуалізації. DBFA має безліч функцій з дистрибутива, і їх можна доповнити плагінами, два з яких особливо корисні для нашого інструменту: плагін системного виклику-трасування та функціональність введення процесу.

Для відстеження всіх системних викликів DBFA розміщує точки зупинку INT3 на початку всіх необхідних функцій ядра. Це має очевидну перевагу, що вся діяльність, враховуючи внутрішню активність ядра, буде безпосередньо залежати від DBFA порівняно з руткітом у режимі ядра, уникаючи методів підключення системних викликів, що покладаються на фактичні механізми системних викликів, тобто перехід із призначеного для користувацького інтерфейсу в ядрі режим виконання. Функції ядра розміщуються за допомогою інструментарію криміналістики Rekall [6] у поєднанні з інформацією про символи налагодження ядра, наданою корпорацією Майкрософт. Розташування самого ядра у пам'яті базується на Windows 7, використовуючи регістри центрального процесора FS та GS для зберігання віртуальної адреси ядра із певним символом, що також може бути використано для визначення базової адреси ядра.

Цікавою особливістю DBFA є її здатність реалізовувати процеси в клієнтській області віртуальної машини без допоміжного процесу, що працює у віртуальній машині (часто її називають агентом). Цей процес введення заснований на подіях захоплення в ЦП реєстру CR3, який використовується для віртуальної адресації та буде записаний до кожного перемикача контексту. Введення процесу вимагає цільового процесу, що використовується як тимчасовий сурогат для виклику Windows CreateProcessA, що є стандартним методом для запуску процесу в операційній системі. Коли здійснюється контекстний перехід до цільового процесу, DBFA починає виконувати інструкції

віртуальної машини один раз, поки не буде виконана перша інструкція користувача режиму. На цьому етапі необхідні аргументи системного виклику CreateProcessA розміщуються в стеку, зберігаються вихідні значення необхідних регістрів ЦП для системного виклику, а регістри, включаючи вказівник інструкцій, змінюються для системного виклику. Після того, як система повернеться до початкового стану, визначеного іншою точкою зупинки, стан виклику можна нормально читати відповідно до угоди про виклик, тоді інформація про процес буде зберігатися у DBFA перед тим, як повертати стек та регістри ЦП до очікуваного значення для цільового процесу. Вона також вживає додаткових заходів для того, щоб перемикач контексту не спричиняв нестабільність системи під час процесу введення. [27]

Метод відстеження системних викликів - або, вірніше, викликів функцій ядра - має ту перевагу, що працює з певністю для всіх поточних методів системних викликів та навіть будь-яких майбутніх. DBFA є досить ефективним, особливо з точки зору пам'яті, завдяки можливості використовувати власну семантику пам'яті Copy-on-Write Xen. [27] Крім того, ви можете використовувати подібні копії для запису в логічний том Linux, щоб зменшити простір і особливо час, необхідний для клонування віртуальної машини, про що йтиметься далі в цій дисертації. Додаток А на плакаті 8 містить огляд технічного рівня DBFA на високому рівні. Подібність з архітектурою Ether відразу стає зрозумілою. Додаток А на плакаті 9 представляє процес самоаналізу DBFA. Додаткове введення процесу є особливістю системи DBFA.

Базуючись на розроблених моделях та механізмах захисту сформулюємо основні вимоги до системи захисту:

1. Необхідно реалізувати дискреційний механізм політики безпеки.

2. Забезпечити розмежування доступу.

3. Запровадити керування потоками для виключення сутності «Власник».

4. Запровадити дозвільну (дискреційну) політику доступу.

5. Запровадити механізм контролю уособлення, який забороняє отримання прав інших користувачів.

Для правильної реалізації політики розмежування права доступу повинні бути присвоєні суб'єктам, а не діяти як атрибути для об'єктів доступу. Для об’єкта доступу, зазначеного розширенням, слід використовувати наступний стандартний набір правил

доступу: читання; запис; реалізація; видалення; перейменування. Встановивши для перейменованого об’єкта, зазначеного розширенням, це правило повинно забороняти перейменування будь-якого іншого об’єкта файлу в цей об’єкт файлу та створення нового об’єкта для доступу до вказаного розширення.

3.2 Схема механізму забезпечення захисту

Виконання сформульованих вимог забезпечує побудову захищеної системи з точки зору запобігання витоку прав доступу під час функціонування системи та змінює відомі підходи до здійснення контролю доступу.

MS Windows використовує дискреційний контроль доступу, а саме метод, заснований на встановленні атрибутів доступу. Контроль доступу (розмежування) здійснюється центральним елементом системи безпеки - менеджером доступу, перехоплює всі запити до об'єктів, ідентифікуючи суб'єкти, об'єкти доступу та параметри запитуваного доступу суб'єкта до об'єкта (читання, запис, виконання та ін.), надання на основі спочатку встановлених правил суб’єкту, в якому вимагається доступ, або відмова в ньому. Для встановлення правил доступу кожному об’єкту присвоюються атрибути доступу, які вказують, які сутності та яким (з якими параметрами) дозволено / не дозволено доступ до них. Запитуючи доступ, менеджер, отримуючи необхідні облікові дані із запиту, зчитує атрибути, що належать об’єкту, до якого пропонується ввести відповідний доступ, на основі їх аналізу визначає правильність обробленого запиту. Одночасно вносить інформацію до журналів аудиту про доступ

Схема розмежувальної політики зображена на рис. 3.1.

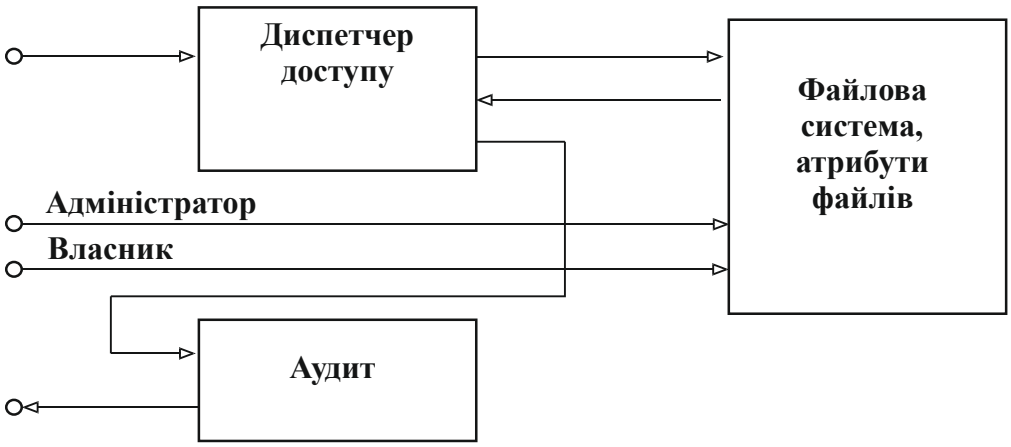


Рисунок 3.1 - Реалізація політики доступу в ОС Windows

Використовуючи наведену вище схему, неможливо встановити політику розмежування для доступу до файлових об'єктів, визначених масками (включаючи розширення), оскільки права доступу встановлюються як атрибути доступу певного об'єкта, що, звичайно, не передбачає маску завдання .

Враховуючи отримані результати та сформульовані вимоги, що дозволяють побудувати безпечну систему, ми формулюємо вимоги до вирішення проблеми. Пропоноване рішення передбачає використання методу, заснованого на задачі матриці доступу. Цей метод контролю доступу кардинально відрізняється тим, що правила доступу формуються вже не для об’єкта, а для суб’єкта доступу. Для суб'єктів доступу встановлено, до яких об'єктів і за якими правилами (з якими правами) їм дозволений доступ, тобто правила доступу тут вже належать не об'єкту, а суб'єкту доступу. Запитуючи доступ, менеджер доступу отримує доступ до матриці доступу (таблиці правил) і зчитує з неї дозволені правила доступу суб’єкта до об’єкта, що з’являються в запиті, на основі чого аналізується правильність запиту. Крім того, необхідна інформація вноситься до журналу аудиту.

Загальна схема реалізації ПД зображена на рис.3.2.

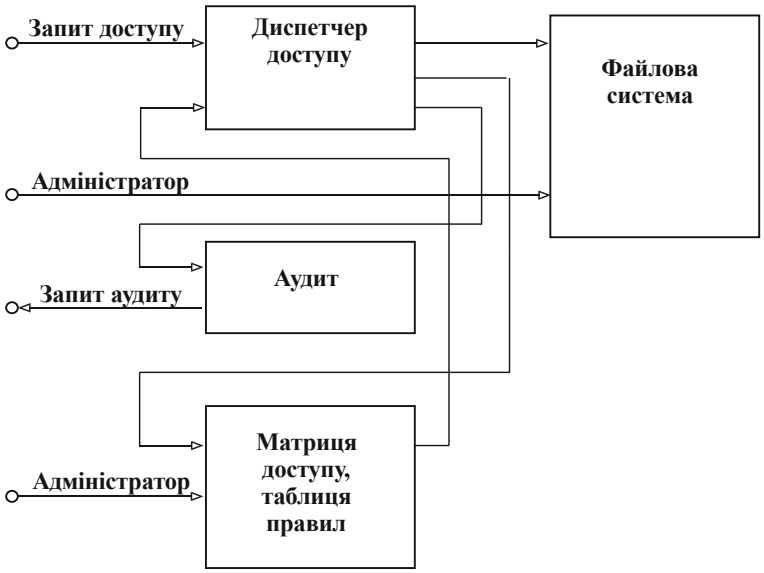


Рисунок 3.2 - Реалізація запропонованого рішення

3.3 Запровадження ПД для захисту від ШПЗ

Запропонований метод захисту від запуску загрозливих програм, реалізує такі етапи:

1. Створіть нову сутність доступу, оскільки сутностями доступу є користувачі, системи та процеси.

2. Сутність доступу визначається трьома сутностями: ефективним, основним користувачем та процесом.

3. Створення теми доступу та профілю захисту "Звичайний користувач".

4. Тема доступу "Адміністратор" створюється окремо, щоб усунути можливість встановлення нового програмного забезпечення або оновлення існуючого.

5. Для суб'єкта доступу "Адміністратор" як ефективного та основного користувача встановлюється права Адміністратор.

6. Призначення об'єктів доступу, які є виконуваними. Вони мають наступні розширення: *.cmd ; *.exe; *.bat; *.vbe; *.vbs ; *.js; *.jse; *.wsf; і т.д.

7. Системні файли мають розширення: *.config, *.manifest, *.drv, *.dll, *.log, *.sys. Системні та виконувані файли повинні призначатися з урахуванням усунення недоліку, пов'язаного з неможливістю диференціювати доступ до тих ресурсів, зміна яких ми не можемо контролювати.

В таблиці 3.1 представлено запропоновану політику доступу, відповідно до матриці доступуФайли поділяться на виконувані, системні та інформаційні.

Таблиця 3.1 - Розмежувальна політика згідно матриць доступу

Суб'єкт доступу	Об'єкт доступу	Правило розмежування доступу
1	2	3
Будь-який ефективний і первинний користу-вач і будь-який процес	Виконуваний файл	Читання дозволено; Запис не дозволено; Виконання дозволено; Перейменування не дозволено; Видалення не дозволено.
	Системний файл	Читання дозволено; Запис не дозволено; Виконання не дозволено; Перейменування не дозволено; Видалення не дозволено.
		Читання дозволено; Запис дозволено;

	інформаційний файл	Виконання не дозволено; Перейменування не дозволено; Видалення не дозволено.
	виконуваний файл	Читання не дозволено; Запис заборонено; Виконання дозволено; Перейменування не дозволено; Видалення не дозволено.

Кінець таблиці 3.1

1	2	3
Будь-який ефективний і первинний користувач і процес віртуальної машини	системний файл	Читання дозволено; Запис не дозволено; Виконання не дозволено; Перейменування не дозволено; Видалення не дозволено.
	інформаційний файл	Читання дозволено; Запис дозволена; Виконання не дозволено; Перейменування не дозволено; Видалення не дозволено.
В якості ефективного і первинного виступає Адміністратор і будь-який процес	виконуваний файл	Читання дозволено; Запис дозволено; Виконання дозволено; Перейменування дозволено; Видалення дозволено.
	системний файл	Читання дозволено; Запис дозволено; Виконання не дозволено; Перейменування дозволено; Видалення дозволено.
	інформаційний файл	Читання дозволено; Запис дозволено; Виконання не дозволено; Перейменування дозволено; Видалення дозволено.

3.4 Висновки

1. В розділі пропонується реалізувати розроблені методи захисту від загрозливих програм, враховуючи дотримання сформульованих вимог до побудови захищеної системи з точки зору запобігання витоку прав доступу під час роботи системи. Показано, що практична реалізація сформульованих вимог до побудови безпечної системи принципово змінює підходи до здійснення контролю доступу до ресурсів.

2. Доведено, що всі розроблені методи можуть бути реалізовані одним із засобів захисту при використанні різних політик безпеки (різних налаштувань механізму захисту).

3. Розроблено політику безпеки, яка підтверджує можливість практичної реалізації розроблених методів та ілюструє простоту адміністрування та функціонування захисту.

4 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО МЕТОДУ

4.1 Реалізація способу підвищення безпеки

Блок аудиту одним із перших завантажується під час запуску системи, щоб не втратити важливі дані, зібрані під час запуску системи. Блок перехоплює всі виклики, тому, відповідно до необхідної політики безпеки на даному вузлі, вона налаштовується шляхом вказівки набору правил для зменшення навантаження. Під час здійснення системного виклику з простору користувача дані надходять безпосередньо до блоку аудиту та передаються через систему, що складається з п’яти фільтруючих елементів, як показано на рисунку 4.1. Найважливішим з точки зору аналізу є вихідний фільтр, оскільки він містить всю інформацію про системний виклик.

Схема реалізації викликів з простору користувача наведена на рис. 4.1.

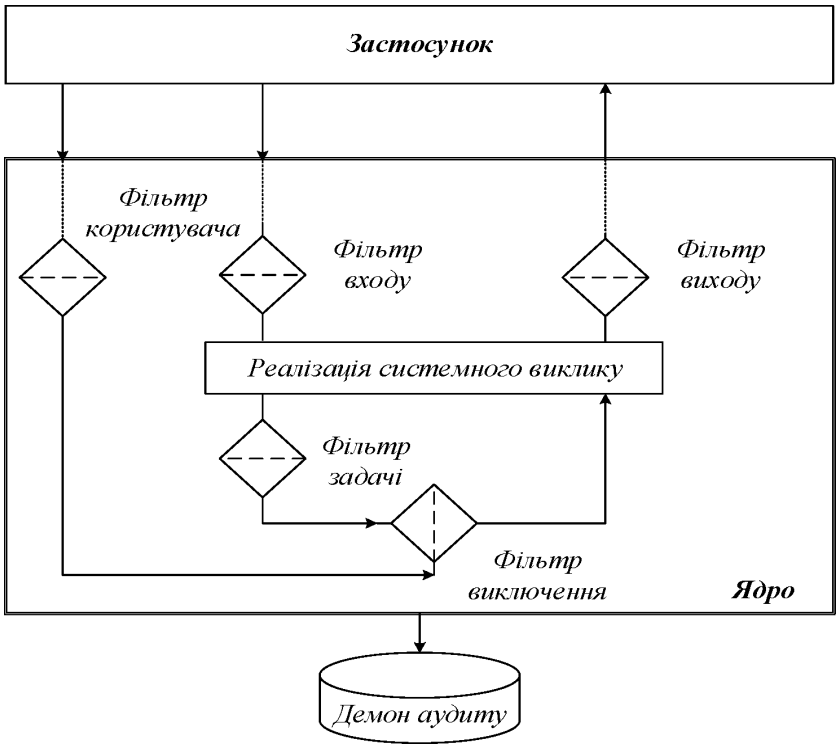


Рисунок 4.1 - Схема реалізації викликів з простору користувача

Завданням блоку аудиту є отримання детальної інформації про стан системних процесів, що відповідають певній політиці, та реєстрація будь-яких типів подій у режимі

реального часу. Дані отримують шляхом проходження кожного процесу через систему фільтрів блоку аудиту.

Наявність рівня у вигляді модуля ядра аудиту дозволяє побудувати універсальний інструмент для перехоплення та управління системними викликами та процесами, а також забезпечує можливість збору даних, необхідних для подальшого використання при аналізі системної діяльності за зловмисну діяльність. Відповідно, обов'язковим є надання підсистемі аудиту можливості записувати такі дані: дату та час події, тип події, ідентифікатор сутності, результат дій; асоціація події з ідентифікацією користувача, який її спричинив; всі спроби модифікувати файли конфігурації підсистеми аудиту та отримати доступ до журналів; використання механізмів автентифікації; зміни у надійних джерелах та базах даних; спроби імпорту та експорту інформації. Також важливо мати можливість увімкнути або виключити події на основі ідентифікатора користувача, міток об'єктів та інших атрибутів.

Для проведення аналізу системи на основі отриманих даних пропонується побудувати аналізатор з використанням засобів машинного навчання, а саме - нейронних мереж. Цей підхід створить систему, здатну до самонавчання та підвищення якості роботи в процесі нових загроз, за допомогою використання даних з баз знань та досвіду. Перевагою цього підходу є відносна стійкість до вразливостей нульових днів.

Перехоплення всіх системних викликів, що відбуваються в ОС, може мати значний вплив на продуктивність системи. Тому для поліпшення використання системних ресурсів пропонується виконувати подальшу трасування лише для викликів, які збуджують тригери аналізатора, як показано на рисунку 4.2. Для цього всі точки входу в будь-яке програмне забезпечення повинні містити особистий ідентифікатор для входу. Також необхідно створити список надійних програм, які матимуть доступ до файлів бази знань та журналів підсистеми аудиту для їх оновлення та модифікації (серед них слід також виділити сервісні програми, які включені до списку утиліти та демони, безпосередньо необхідні для нормальної роботи системи).

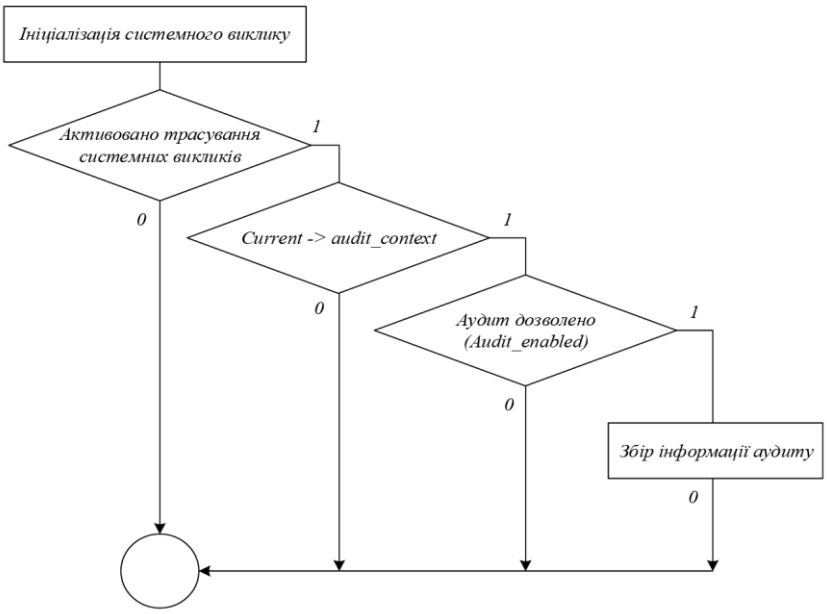


Рисунок 4.2 - Схема прийняття рішень щодо збору даних підсистемою аудиту

Згідно з наведеною вище каскадною схемою, необхідність ведення журналу визначається лише за умови, що поточний суб'єкт відповідає політиці аудиту.

Щоб обмежити вплив сторонніх суб’єктів програмного забезпечення, пропонується ізолювати їх за допомогою віртуальної файлової системи. Цей крок необхідний для визначення шкідливого програмного забезпечення. Наступна схема (рис. 4.3) демонструє принцип взаємодії простору користувача з різними типами файлових систем.

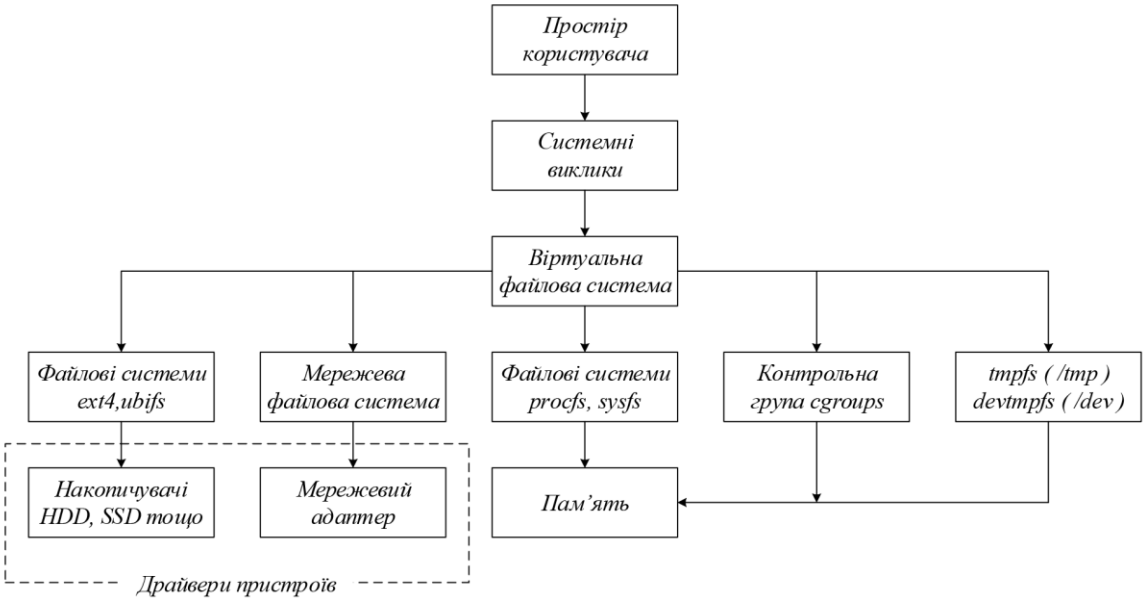


Рисунок 4.3 - Принцип взаємодії простору користувача з різними типами ФС

Абстракція у вигляді віртуальної файлової системи дозволяє користувачеві не тільки безпечно взаємодіяти з третіми сторонами, але також дозволяє використовувати структури даних інших файлових систем. Через ізоляцію на основі часткової віртуалізації файлової системи, користувацький простір відокремлений. Це запобігає втратам при спробі модифікувати ці об’єкти, оскільки змінюватимуться лише дублікати об’єктів файлової системи в обмеженому середовищі. Обмеження прав у такій системі не дозволяє атакувати основну систему зсередини віртуальної FS через інтерфейси операційної системи [14].

Структурна схема алгоритму методу підвищення безпеки комп'ютерного вузла на основі підсистем аудиту операційної системи представлена на рис. 4.4

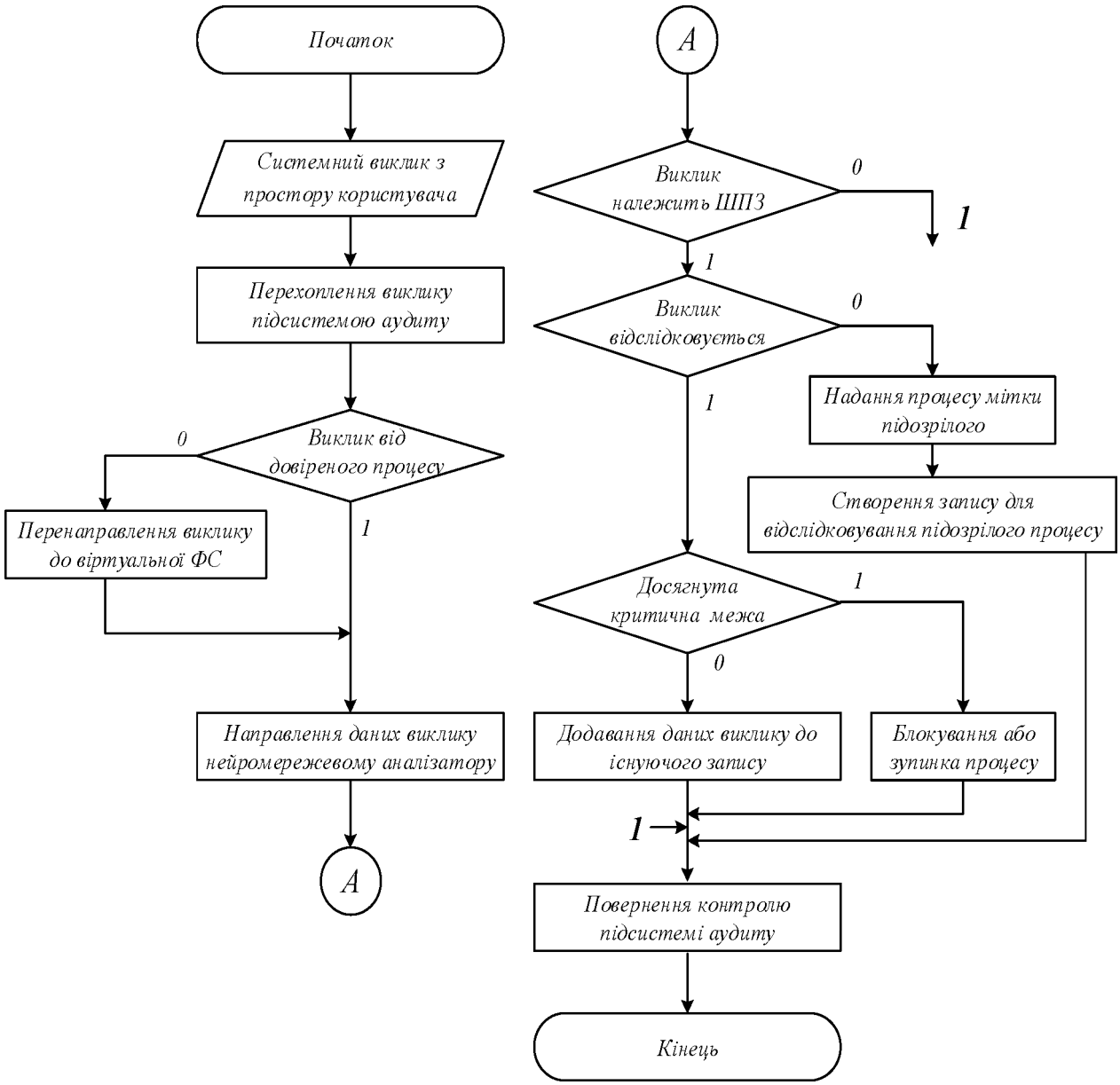


Рисунок 4.4 – Блок-схема алгоритму методу підвищення безпеки

Аналіз структури підсистеми аудиту з метою використання її як основи для розробки засобів інформаційної безпеки виявив недоліки, які необхідно усунути, а також надав можливість розробити алгоритм з урахуванням реалізації підсистеми аудиту.

4.2 Оцінка ефективності запропонованого методу

Одним із найпоширеніших підходів до оцінки якості інформаційної безпеки є певний розподіл реалізованих функцій та завдань, характеристик роботи та вимог відповідно до технічних цілей створення системи безпеки. Іншим методом, що застосовується у вітчизняній та зарубіжній практиці, є аналіз функціональної надійності системи, який також характеризує рівень якості системи захисту інформації.

Абсолютні кількісні показники можна систематизувати за такими різновидами:

1. Технічна. До цієї групи належать:

- Кількість виявлених загроз визначає кількість загроз, які можна ідентифікувати та вирішити. Загроза вважається визнаною, якщо її характеристики збігаються з описами в системі інформаційної безпеки;

- якість реагування на загрози - визначається здатністю системи адекватно реагувати на виявлені загрози. У реальному житті існують загрози, на які важко реагувати за допомогою автоматизованих інформаційних систем. У такій ситуації бажано відзначити в протоколі ті дії, які здійснюються загрозою;

- зниження продуктивності системи в цілому - відображає зниження продуктивності системи через необхідність реалізації дій, передбачених політикою безпеки. Прикладами є картки шифрування, які зменшують швидкість передачі даних через необхідність шифрування при передачі та дешифруванні при отриманні даних тощо [4].

2. Організаційні. Цей тип показників характеризує кількість додаткового персоналу, залученого до служби. Для реалізації функцій безпеки задіяний додатковий персонал - інженери, програмісти, системні адміністратори, менеджери з безпеки.

3. Економічний. Цей різновид включає наступні показники:

- витрати на створення, впровадження, експлуатацію та навчання користувачів та підтримку (усі витрати на всіх етапах життєвого циклу, включаючи витрати на дослідження, придбання технології ноу-хау, спеціального обладнання та програмного забезпечення тощо). Вони також включають заробітну плату працівників, які виконують роботу;

- витрати на конкретні матеріали. Передбачає використання спеціальних витратних матеріалів. Як приклад можна розглянути додаткові магнітні носії, необхідні для резервного копіювання;

- витрати на відновлення нормальної роботи після загрози. Вони включають витрати на інформаційні, технічні, трудові та інші ресурси для відновлення нормальної роботи системи;

- коефіцієнт зменшення потенційних збитків, що характеризує взаємозв'язок між швидкістю зменшення втрат і величиною можливих втрат.

Кількісний рівень захисту автоматизованих інформаційних систем характеризується двома основними групами показників:

1. Відносна кількісна оцінка, яка являє собою число (клас, категорію, нормоване значення), яке вимагає порівняння з іншими числами, прийнятими як стандарт. Для їх визначення використовуються експертні оцінки. Найважливішим пунктом якісної оцінки є питання виправлення та виправлення помилок, що виникають через суб'єктів незалежних оцінок - експертів. Найпопулярнішим методом обстеження є метод Дельфі та його модифікація. Експертиза може бути спрямована на оцінку ефективності системи захисту, рівня допустимого ризику, рівня захисту окремих підсистем тощо.

2. Абсолютна кількісна оцінка інформаційної безпеки в АІС може характеризувати витрати, виражені у грошовому еквіваленті, частоту несприятливих подій або інші показники, які є значущими з точки зору захисту інформації.

Для оцінки розроблених методів ми визначимо потенційний рівень безпеки та умови, за яких він буде забезпечуватися, крім того, визначимо, як впровадження

розроблених методів в інформаційній системі вплине на навантаження обчислювальних ресурсів.

Щоб оцінити ефективність, розгляньте графік цілеспрямованих атак, на якому будуть представлені варіанти встановлення системи захисту інформації. У цьому стовпці вершини зважуються, дуги вказують послідовність атак на інформаційну систему, іншими словами, показують процес використання вразливостей та подолання захисту. Розглянемо три можливих варіанти установки СЗІ:

а) система захисту знешкоджує небезпеку (рис. 4.5, а);

б) система захисту заважає реалізації наслідків загрози (рис. 4.5, б);

в) система захисту знешкоджує небезпеку, і запобігає наслідкам реалізації загрози (рис. 4.5, в).

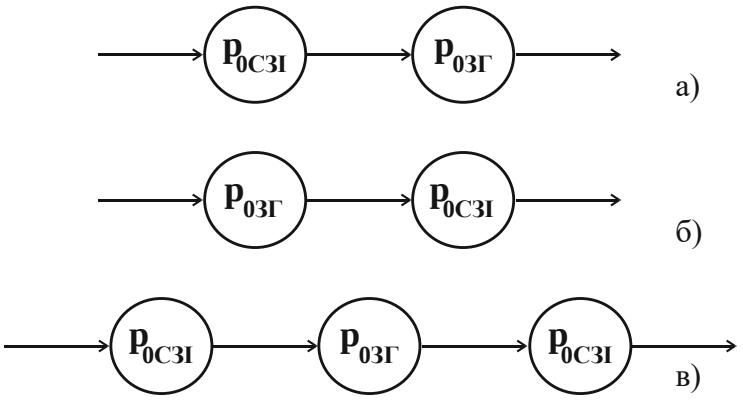
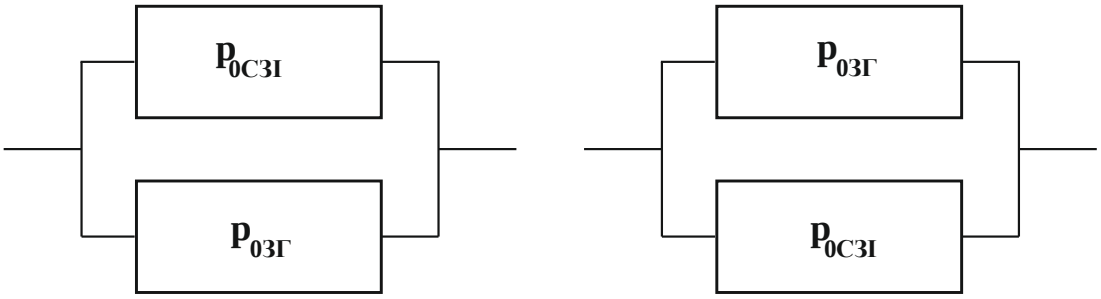


Рисунок 4.5 - Моделі захисту

$P_{0СЗІ}$ - вірогідність того, що в СЗІ немає вразливості, $P_{03Г}$ - вірогідність того, що в системі немає загрозливих програм. На рис. 4.1 *б* зображео функціонування антивірусної програми, оскільки її функціонквання заключається в перевірці файлу на шкідливість після потрапляння його в систему. На рис. 4.1 *в* можливе декількох різних СЗІ. У нашому випадку функціональність одного із засобів захисту інформації включає як можливість запобігання реалізації та запуску загрозливих програм, так і можливість запобігання діям, які будуть вжиті після реалізації загрози.

Промодедюємо засіб захисту у вигляді схеми паралельного резервування. Схема зображена на рис. 4.6 (*а* і *б*)



а) б)

Рисунок 4.6 - Схема паралельного резервування

Базуючись на схемі, зображеній на рис 4.2, отримаємо формулу для обрахування вірогідності здійснення атаки:

$$p_{0зд} = (1 - p_{0ЗГ}) \times (1 - p_{0СЗІ}) \tag{4.1}$$

Тоді вірогідність захищеності інформаційної системи обрахуємо по формулі:

$$p_{0ЗІС} = 1 - p_{0зд} \tag{4.2}$$

$$p_{0ЗІС} = 1 - (1 - p_{0ЗГ}) \times (1 - p_{0СЗІ}) \tag{4.3}$$

Обрахуємо потенційну ефективність ЗЗІ ($p_{0СЗІ}$) і визначимо, за яких умов вона досяжна. Використовуючи формулу (4.3), отримаємо нижню межу значення імовірності відсутності помилок ($p_{0СЗІ}$) програмування або некоректності реалізації, Відмітимоі, що, на відміну від антивірусних програм, засоби контролю доступу можуть повністю запобігти впровадженню та виконанню загрозливих програм.

Вірогідність відсутності помилок для багатоканальної системи розраховується за формулою:

$$p_{0СЗІ} = \left(\sum_{m=0}^{C} \frac{\left(\frac{\lambda}{\mu}\right)^m}{m!} + \frac{\left(\frac{\lambda}{\mu}\right)^{C+1}}{C! \cdot \left(C - \frac{\lambda}{\mu}\right)} \right)^{-1}, \tag{4.4}$$

де λ - це інтенсивність виявлення вразливостей,

μ - інтенсивність виправлення помилок, яка обернено пропорційна часу виправлення помилок, C - кількість обслуговуючих приладів [21].

Для розрахунку ефективності системи захисту важливі два параметра $\lambda_{СЗІ}$ і $\mu_{СЗІ}$. $\lambda_{СЗІ}$ відповідає за помилками програмування і помилковість реалізації механізмів захисту. $\mu_{СЗІ}$ пропорційний виправленню помилок, або доопрацюванню механізмів.

Висока ефективність системи захисту досягається в разі, коли вірогідність $p_{0СЗІ}$ близька до 1, відповідно перевищує значення 0,9.

При заданих $p_{0СЗІ}$ і $\lambda_{СЗІ}$, розрахуємо необхідну інтенсивність виправлення помилок системи захисту інформації (таб. 4.1):

Таблиця 4.1 - Інтенсивність виправлення помилок системи захисту

$p_{0СЗІ}$ / $\lambda_{СЗІ}$ раз / рік	0,9	0,92	0,95	0,98	0,99
0,6	6	6,25	10	25	50
1	10	12,6	20	50	100
3	30	37,6	60	150	300
12	120	150	240	600	1200
48	480	600	960	2400	4800

На основі розрахованої інтенсивності корекції помилок ми можемо визначити час, необхідний у днях для досягнення заданої ефективності, для якого помилку слід виправити. (таб. 4.2):

Таблиця 4.2 - Розрахунковий час в днях для забезпечення заданої ефективності

$p_{0СЗІ}$ / $\lambda_{СЗІ}$ раз / рік	0,9	0,92	0,95	0,96	0,98	0,99
0,5	73	58,3	36,5	29,1	14,6	7,3
1	36,6	29,3	18,25	14,7	7,3	3,65
3	12,2	9,8	6,09	4,87	2,45	1,22
12	3	2,5	1,5	1,22	0,61	0,32
48	0,76	0,59	0,38	0,31	0,16	0,08

Розрахований час безперервної роботи між відмовами наведено в табл. 4.3.

Таблиця 4.3 – Термін бевідмовної роботи

$\lambda_{СЗІ}$	$p_{0СЗІ}$											
	0.9		0.92		0.95		0.96		0.98		0.99	
	$\mu_{СЗІ}$	T_0	$\mu_{СЗІ}$	T_0	$\mu_{СЗІ}$	T_0	$\mu_{СЗІ}$	T_0	$\mu_{СЗІ}$	T_0	$\mu_{СЗІ}$	T_0
0,5	5	1,8	6,24	1,84	10	1,9	12,5	1,92	25	1,96	50	1,92
1	10	0,9	12,5	0,92	20	0,95	25	0,96	50	0,98	100	0,99
3	31	0,3	37,6	0,31	60	0,33	75	0,32	150	0,32	300	0,32
12	120	0,07	150	0,07	240	0,07	300	0,07	600	0,07	1200	0,07
48	480	0,03	600	0,03	960	0,03	1200	0,03	2400	0,03	4800	0,03

Отримані результати завантаження процесора з СЗІ і без СЗІ представлені на рис. 4.3

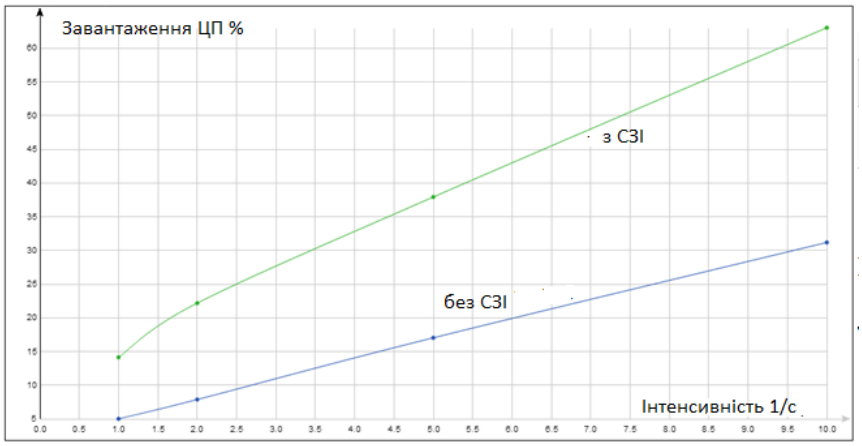


Рисунок 4.7 - Різниця між завантаженням процесора з систеиою захисту і без неї

Аналізуючи графік, зображений на рис. 4.4 можна прогнозувати, як поведе система при різних інтенсивностях. З урахуванням отриманих результатіввидно, що завантаження ЦП в штатному режимі не перевищить 18%, а в режимі запуску системи захисту буде 24%.

4.3 Висновки

1. Аналіз структури підсистеми аудиту з метою використання її як основи для розвитку інформаційної безпеки виявив недоліки, які потребують усунення, а також надав можливість розробити алгоритм, заснований на реалізації підсистеми аудиту.

2. Розглянута модель засобів захисту дозволяє сформулювати вимоги до експлуатаційних параметрів засобів захисту - до інтенсивності виявлення помилок засобами захисту, що дозволяє здійснити успішну атаку, та інтенсивності їх виправлення, виконання яких забезпечує необхідні значення експлуатаційних характеристик засобів захисту.

3. Використовуючи розглянуту модель засобів захисту та сформульовані вимоги, до інтенсивніості виявлення помилок засобами захисту, при виконанні яких досягається високий рівень ефективності застосування пропонованого рішення на практиці. Зокрема показано, що при інтенсивності виправлення помилок у засобі захисту, яка становить 3 - 7 днів (ці значення досяжні на практиці), величина ймовірності готовності засобів захисту до безпечної експлуатації становить 0,97 - 0,99, середній час безвідмовної роботи. збій інформаційної безпеки) засобів захисту в цьому випадку становить від шести місяців до двох років.

4. Показано, що додаткове навантаження процесора в режимі запуску програми не перевищуватиме 22%, а в звичайному режимі роботи програми не перевищуватиме 16%, тоді як час запуску програми збільшується не більше ніж на 3 секунди , що є прийнятним. Адекватність отриманих результатів підтверджується статистичною обробкою експерименту.

5. В результаті досліджень було зроблено висновок, що запропоновані методи захисту від загрозливих програм, засновані на реалізації контролю доступу до файлових об'єктів за їх типами та розширеннями, що підвищує ефективність захисту при цьому завантаження обчислювальних ресурсів інформаційної системи майже не змінюється.

ВИСНОВКИ

Дослідження принципу роботи сучасних засобів захисту та особливостей реалізації шкідливого програмного забезпечення надали можливість виокремити ключові недоліки сучасних способів виявлення загроз

Проведення аналізу структури підсистеми аудиту з метою застосування її як основи для розробки засобів захисту інформації дозволило виявити недоліки, які мають бути попередньо усунені, а також надало можливість розробити алгоритм способу з урахуванням особливостей реалізації взаємодії підсистеми аудиту з ядром операційної системи

В ході вирішення поставлених завдань було:

1. Використовуючи модель атаки, показано, що найбільш небезпечними загрозами для сучасних інформаційних ресурсів є двійкові файли та шкідливі файли. Зазначається, що ці класи шкідливих програм передбачають обов'язкове збереження загрозливого файлу перед його виконанням (читанням). Це дозволило зробити висновок, що захист від ШПЗ може здійснюватися шляхом контролю (розмежування прав) доступу до файлів за розширеннями та типами файлів. Можливість використання такого підходу обґрунтована вивченням засобів правового захисту.

2. Представлено можливість практичної реалізації розроблених методів захисту від загрозливих програм з урахуванням політики безпеки та обмеження доступу.

3. Використовуючи розглянуту модель засобів захисту та сформульовані вимоги, до інтенсивності виявлення помилок засобами захисту, при виконанні яких досягається високий рівень ефективності застосування пропонованого рішення на практиці. Зокрема показано, що при інтенсивності виправлення помилок у засобі захисту, яка становить 3 - 7 днів (ці значення досяжні на практиці), величина ймовірності готовності засобів захисту до безпечної експлуатації становить 0,97 - 0,99, середній час безвідмовної роботи. збій інформаційної безпеки) засобів захисту в цьому випадку становить від шести місяців до двох років.

4. Вдосконалено метод побудови системи захисту від реалізації та запуску загрозливих програм на основі контролю доступу, моделі та методи захисту

інформаційної системи від бінарних та скриптових файлів, на основі реалізації політики розмежування доступу до файлових об'єктів.

5. Проведено випробування та оцінено вплив засобів захисту, що реалізують розроблені методи, на навантаження обчислювального ресурсу, в результаті чого показано, що це набагато менше, ніж вплив на завантаження обчислювального ресурсу антивірусним засобами, зокрема, додаткове навантаження процесора в системі не перевищує 22%, а в звичайному режимі програми не перевищує 16%, тоді як час запуску додатків збільшується не більше ніж на 3 секунди, що майже не впливає на роботу користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Аласенко А.В. Разработка и системный анализ математической модели угроз, модели нарушителя, процедур защиты WEB-приложений на всех этапах функционирования / А.В. Аласенко, П.И. Дзьобай // Научный журнал КубГАУ. — 2014. — № 101(07). — С. 1-11.

2 Борисов М. А. Основы программно-аппаратной защиты информации. / М. А. Борисов, И. В. Заводцев, И. В. Чижов. – М.: УРСС: Либроком, 2013. – 370 с.

3 Жуков Ю.В. Основы веб-хакинга: нападение и защита / Ю.В. Жуков. — СПб.: Питер, 2011. — 176 с.

4 Експерти з кібербезпеки зазначають, що на розшифровування вірусу можуть піти тижні. [Електронний ресурс] // ТСН.ua. - 2017. - Режим доступу до ресурсу:https://tsn.ua/ukrayina/u-antivirusnoyi-kompaniyi-rozpovili-hto-mozhe-stoyati-za-hakerskoyu-atakoyu-petya-a-i-chim-ce-zagrozhuye-952457.html.

5 Казіміров В.О. Метод захисту відзагрозливих програм, заснований на реалізації контролю доступу до файлових об'єктів / В.О. Казіміров, С.В. Мостовий, В.С. Орленко // «Інтелектуальний потенціал – 2020» - збірник наукових праць молодих науковців і студентів / Колектив авторів – Хмельницький: ПВНЗ УЕП, 2020. – Частина 2. С. 45-49

6 Лугановська Є. GDPR в Україні: стратегічний план чи необдумане рішення? [Електронний ресурс] / Є. Лугановська // delo.ua. - 2018. - Режим доступу до ресурсу: https://delo.ua/business/gdpr-v-ukraiini- strategichnii-plan-chi-neobduman-348025/ .

7 Михайлов А. В. Компьютерные вирусы и борьба с ними. / А.В. Михайлов. – М.: Диалог-МИФИ, 2012. – 148 с.

8Нова хвиля кібератак в Україні може поширитись через сайт розробника ПЗ для бухобліку - експерти Детальніше читайте на УНІАН: https://www.unian.ua/science/2095776-nova-hvilya-kiberatak-v-ukrajini-moje- poshiritis-cherez-sayt-rozrobnika-pz-dlya-buhobliku-e [Електронний ресурс] // УНІАН. - 2017.- Режим доступу до ресурсу: https://www.unian.ua/science/2095776-nova-hvilya-kiberatak-v-ukraiini-moie-poshiritis-cherez-sayt-rozrobnika-pz-dlya-buhobliku-eksperti.html.

9 Партыка Т. Л. Информационная безопасность учебное пособие / Т. Л. Партыка, И. И. Попов. – М.: ФОРУМ, 2011. – 432 с.

10 Поворознюк А.И. Совершенствование защиты Web-приложений от вторжений на основе эвристического похода / А.И. Поворознюк, М.Н. Шкарупа: сб. науч. тр. «Вестник НТУ «ХПИ». Інформатика і моделювання. — 2007. — Вип. 19. — С. 145-154.

11 Проскурин В. Г. Защита программ и данных: учебное пособие / В. Г. Проскурин, С. В. Крутов, И. В. Мацкевич. – М.: Академия, 2011. – 198 с.

12 Про захист інформації в інформаційно-телекомунікаційних системах [Електронний ресурс] // Законодавство України. - 2014. - Режим доступу до ресурсу: https://zakon2 .rada.gov.ua/laws/show/80/94-%D0%B2%D 1 %80.

13 Про затвердження Інструкції про порядок обліку, зберігання і використання документів, справ, видань та інших матеріальних носіїв інформації, які містять службову інформацію [Електронний ресурс] // Законодавство України. - 2016. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/1893-98-%D0%BF.

14 Сорокин С.Н. Метод обнаружения атак типа «отказ в обслуживании» на WEB-приложения / С.Н. Сорокин // Прикладная дискретная математика. — 2014. — № 1(23). — С. 55-64.

15 Сердюк В. А. Организация и технологии защиты информации / В. А. Сердюк. – М.: Издательский дом Государственного университета – Высшей школы экономики, 2011. – 571 с.

16 Точно в цель - атака на Target [Електронний ресурс] // Infowatch. - 2014. - Режим доступу до ресурсу: https://www.infowatch.ru/analytics/leaks monitoring/5196.

17 Фаткиева Р.Р. Разработка метрик для обнаружения атак на основе анализа сетевого трафика / Р.Р. Фаткиева // Вестник Бурятского государственного университета. — 2013. — Vol. 9.— С. 81-86.

18 Шаньгин В. Ф. Защита информации в компьютерных системах и сетях. / В. Ф. Шаньгин. - М.: ДМК Пресс, 2012. – 576 с.

19 Bhavani A.B. Cross-site Scripting Attacks on Android WebView / A.B. Bhavani // International Journal of Computer Science and Network. — 2013. — Vol. 2, Issue 2. — 5 p. — Режим доступу в Інтернет: http://ijcsn.org/IJCSN-2013/2-2/IJCSN-2013-2-2-03.pdf

20 Blackmer M. Attacking the Weakest Link in the Supply Chain [Електронний ресурс] / Marc Blackmer // Cisco Blog. - 2017. - Режим доступу до ресурсу: https://blogs.cisco.com/security/attacking-the-weakest-link-in-the-supply-chain?dtid=osscdc000283.

21 Cuff P. Distributed channel synthesis / P. Cuff // IEEE. Trans. Inf. Theory. — 2013. — Vol. 59(11). — P. 7071-7096

22 Handbook on Ontologies / eds. S. Staab and R. Studer. — International Handbooks on Information Systems. — Berlin: Springer, 2009. — 832 p.

23 Sahin C.S. General Framework for Evaluating Password Complexity and Strength / C.S. Sahin, R. Lychev, N. Wagner. — 11 p. — Режим доступу в Інтернет: http://arxiv.org/abs/1512.05814

24 Sen J. A Robust Mechanism for Defending Distributed Denial OF Service Attacks on Web Servers / J. Sen // International Journal of Network Security & Its Applications (IJNSA). — 2011, March. — Vol. 3, N 2. — P. 162-179.

25 Schieler C. Rate-distortion theory forsecrecy systems / C. Schieler, P. Cuff // IEEE Trans. on Inf. Theory. — 2014. — Vol. 66(12). — P.7584-7605.

26 Website Security Statistics Report: 2019. — WhiteHat Security, 2019. — 30 p. — Режим доступу в Інтернет: https://info.whitehatsec.com/Website-Stats-Report-2019.html

27 Pankov N. Не станьте звеном в атаке через цепочку поставок [Електронний ресурс] / Nikolay Pankov // Kaspersky Lab. - 2018. - Режим доступу до ресурсу: https://www.kaspersky.ru/blog/ccleaner-supply-chain/20045/ .

28 CCleanup: A Vast Number of Machines at Risk [Електронний ресурс] / Edmund Brumaghin, Ross Gibb, Warren Mercer та ін.] // Cisco Talos. - 2017. - Режим доступу до ресурсу:https://blog.talosintelligence.com/2017/09/avast-distributes-malware.html .

29 New Ransomware Variant "Nyetya" Compromises Systems Worldwide [Електронний ресурс] // Cisco Talos. - 2017. - Режим доступу до ресурсу: https://blog.talosintelligence.com/2017/06/worldwide-ransomware- variant.html.

30 Target Hackers Broke in Via HVAC Company [Електронний ресурс] Режим доступу до ресурсу:https://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/.

31 BPatrick Moorhead. That Time Of Year Again: Cisco Systems Releases Its Annual Cybersecurity Report [Електронний ресурс] / Patrick Moorhead // Forbes. -2018.- Режим доступу до ресурсу:https://www.forbes.com/sites/patrickmoorhead/2018/03/05/that-time-of-year-again-cisco-systems-releases-its-annual-cybersecurity-report/#3e35b42518ec.

32 Information security breaches survey 2013: technical report [Електронний ресурс] // Gov.uk. - 2013. - Режим доступу до ресурсу: https://www.gov.uk/government/publications/information-security-breaches-survey-2013 - technical-report.

33 Managing Cyber Supply Chain Risks [Електронний ресурс] // Advisen Insurance Intelligence. - 2013. - Режим доступу до ресурсу: http://www.advisenltd.com/wp-content/uploads/2013_OBPI_SupplyChain CyberRM_Whitepaper.pdf.

34 Cyber-security risks in the supply chain [Електронний ресурс] // Cert -uk. -2015. - Режим доступу до ресурсу: https://www.cert.gov.uk/wp-content/uploads/2015/02/Cyber-security-risks-in-the-supply-chain.pdf.

35 ROB WAUGH. The terrifying rise of cyber crime: Your computer is currently being targeted by criminal gangs looking to harvest your personal details and steal your money [Електронний ресурс] / ROB WAUGH // Mail Online. - 2013. -Режим доступу до ресурсу: https://www.dailymail.co.uk/home/moslive/article-2260221/Cyber-crime-Your-currently-targeted-criminal-gangs-looking-steal-monev.html.

36 Supply Chain Risk - the “Cyber Attack” [Електронний ресурс] // ISG – Режим доступу до ресурсу: https://www.isg-one.com/industries/consumer-goods/articles/supply-chain-risk-the-cyber-attack.

37 Elia Florio. Attack inception: Compromised supply chain within a supply chain poses new risks [Електронний ресурс] / Elia Florio, Lior Ben Porat // Microsoft. - 2018. - Режим доступу до ресурсу: https://www.microsoft.com/security/blog/2018/07/26/attack-inception- compromised-supply-chain-within-a-supply-chain-poses-new-risks/.

38 ISO/IEC 28001:2007, Security management systems for the supply chain — Best practices for implementing supply chain security, assessments and plans — Requirements and guidance.

ДОДАТОК А

(обовязковий)

Код (лістинг) програмного забезпечення виявлення загрозливих програм

```
ProgramANTIVIRUS;
Uses dos,crt,printer;
Type St80 = String[80];
Var
R:Registers;
FileInfection:File Of Byte;
SearchFile:SearchRec;
Mas:Array[0..80] of St80;
MasByte:Array[1..3] of Byte;
Position,I,J,K:Byte;
Num,NumberOfFile,NumberOfInfFile:Word;
St:St80;
Flag,NextDisk,Error:Boolean;
Dt:DateTime;
Key1,Key2,Key3,NumError:Byte;
MasScreen:Array[0..24,0..159] Of Byte Absolute $B800:0000;
Ch,Disk:Char;
Procedure Cure(St : St80);
Var
I: Byte; MasCure: Array[1..24] Of Byte;
Begin
Assign(FileInfection,St); Reset(FileInfection);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Seek(FileInfection,FileSize(FileInfection) - ($0C1F - $0C1A));
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Read(FileInfection,Key1);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Read(FileInfection,Key2);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Seek(FileInfection,FileSize(FileInfection) - ($0C1F - $0BAA));
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
For I:=1 to 24 do
Begin
Read(FileInfection,MasCure[i]);
NumError:=IOResult;
```

```
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Key3:=MasCure[i];
InLine($50/
$8A/$26/KEY1/
$30/$26/KEY3/
$A0/KEY2/
$00/$C4/
$88/$26/KEY1/
$58);
MasCure[i]:=Key3;
End;
Seek(FileInfection,0);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
For I:=1 to 24 do Write(FileInfection,MasCure[i]);
Seek(FileInfection,FileSize(FileInfection) - $0C1F);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Truncate(FileInfection);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Close(FileInfection); NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Num:=Num+1;
End;
Procedure F1(St : St80);
Begin
FindFirst(St + '*.*', $3F, SearchFile);
While (SearchFile.Attr = $10) And (DosError = 0) And
((SearchFile.Name = '.') Or (SearchFile.Name = '..')) Do
Begin
FindNext(SearchFile);
End;
While (DosError = 0) Do
Begin
If KeyPressed Then
If (Ord(ReadKey) = 27) Then Halt;
If (SearchFile.Attr = $10) Then
Begin
Mas[k]:=St + SearchFile.Name + '\';
K:=K+1;
End;
If (SearchFile.Attr <> $10) Then
Begin
NumberOfFile:=NumberOfFile + 1;
```

```
UnpackTime(SearchFile.Time, DT);
For I:=18 to 70 do MasScreen[6,2*i]:=$20;
GoToXY(18,7);
Write(St + SearchFile.Name,' ');
If (Dt.Sec = 60) Then
Begin
Assign(FileInfection,St + SearchFile.Name);
Reset(FileInfection);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
Seek(FileInfection,FileSize(FileInfection) - $8A);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
For I:=1 to 3 do Read(FileInfection,MasByte[i]);
Close(FileInfection);
NumError:=IOResult;
If (NumError <> 0) Then Begin Error:=True; Exit; End;
If (MasByte[1] = $35) And (MasByte[2] = $2E) And
(MasByte[3] = $30) Then
Begin
NumberOfInfFile:=NumberOfInfFile + 1;
GoToXY(1,8);
Write( St + SearchFile.Name,' inficirovan. ',
'Udalit? [Y/N] ');
Repeat
Ch:=ReadKey;
If (Ord(Ch) = 27) Then Exit;
Until (Ch = 'Y') Or (Ch = 'y') Or (Ch = 'N')
Or (Ch = 'n');
If (Ch = 'Y') Or (Ch = 'y') Then
Begin
Cure(St + SearchFile.Name);
If (NumError <> 0) Then Exit;
End;
For I:=0 to 79 do MasScreen[7,2*i]:=$20;
End; End; End;
FindNext(SearchFile);
End;
End;
Begin
Repeat
Flag:=True;
TextAttr:=$1F;
Repeat
ClrScr;
```

```
GoToXY(29,1); TextAttr:=$1E; GoToXY(20,2); TextAttr:=$17;
Writeln('Programma dlya poiska i lecheniya fajlov,');
GoToXY(28,3);
Writeln('zaragennih SVC50.');
TextAttr:=$4F; GoToXY(1,25);
Write(' ESC - exit ');
TextAttr:=$1F; GoToXY(1,6);
Write('Kakoj disk proverit? ');
Disk:=ReadKey;
If (Ord(Disk) = 27) Then Exit;
R.Ah:=$0E; R.Dl:=Ord(UpCase(Disk))-65;
Intr($21,R); R.Ah:=$19; Intr($21,R);
Flag:=(R.Al = (Ord(UpCase(Disk))-65));
Until Flag;
NextDisk:=True;
Error:=False;
Num:=0;
K:=0;
St:=UpCase(Disk) + ':\';
GoToXY(1,6);
Writeln('Testiruetsya disk ',St,' ');
Writeln('Testiruetsya fajl ');
NumberOfFile:=0;
NumberOfInfFile:=0;
F1(St);
If (k = 0) Or Error Then Flag:=False;
If (k > 0) Then K:=K-1;
While Flag Do
Begin
If (k=0) Then Flag:=False;
F1(Mas[k]);
If (k > 0) Then K:=K-1;
End;
GoToXY(1,10);
Writeln('Provereno fajlov - ',NumberOfFile);
Writeln('Zarageno fajlov - ',NumberOfInfFile);
Writeln('Izlecheno fajlov - ',Num);
Write('Proverit drugoj disk? [Y/N]');
Repeat
Ch:=ReadKey;
If (Ord(Ch) = 27) Then Exit;
Until (Ch = 'Y') Or (Ch = 'y') Or (Ch = 'N') Or (Ch = 'n');
If (Ch = 'N') Or (Ch = 'n') Then NextDisk:=False;
Until Not(NextDisk);
End.
```

ДОДАТОК Б
(обовязковий)
Копії наукових праць

к.т.н., доц. Орленко В.С. *(ХмНУ)*
к.т.н. доц. Муляр І.В. *(ХмНУ),*
Казіміров В.О. *(ХмНУ)*

ЗАХИСТ ВІД ЗАГРОЗЛИВИХ ПРОГРАМ, ЗАСНОВАНИЙ НА КОНТРОЛІ ДОСТУПУ ДО РЕСУРСІВ

Однією з важливих задач забезпечення комп'ютерної безпеки є необхідність ефективної протидії загрозливим програмам. У загальному випадку атаки подібних програм можуть бути націлені, як на розкрадання даних, так і на виведення з ладу комп'ютерних ресурсів, як наслідок, об'єктами захисту, стосовно до даних загроз, повинні бути, як інформаційні, так і системні комп'ютерні ресурси. Існуюча статистика зростання загрозливих програм дозволяє припустити про низьку ефективність методів вирішення найбільш актуальних сучасних завдань захисту інформації. Незалежно від типу, загрозливі програми здатні завдавати значної шкоди, реалізуючи будь-які загрози інформації - порушення цілісності, конфіденційності, доступності [1].

В рамках проведеного дослідження ставиться задача моделювання системи антивірусного захисту з використанням математичного апарату теорії масового обслуговування, визначення і розрахунку основних характеристик з подальшою оцінкою реальної ефективності відомих методів захисту від загрозливих програм.

Проведено дослідження основних типів загрозливих програм, на підставі якого запропоновано класифікацію загрозливих програм за способом їх виконання. На підставі існуючої статистики зроблено висновок, що найбільш актуальними для захисту є виконувані бінарні і скриптові файли. Проведено дослідження способів впровадження загрозливих програм, в результаті якого дійшли висновку - класи загрозливих програм, що розглядаються передбачають обов'язкове збереження файлу на жорсткому диску перед виконанням (читанням).На основі проведених досліджень всіх типів загрозливих програм пропонується провести їх класифікацію за способами виконання загрозливих файлів. У загальному вигляді загрозливі програми слід ділити на виконувані і макро-програми, в свою чергу виконувані діляться на бінарні, мережні загрозливі програми, класичні комп'ютерні віруси, троянські програми, комп'ютерні черв'яки, хакерські утиліти, потенційно небажане програмне забезпечення, і скриптові загрозливі програми [2].

Запропоновано загальний підхід до захисту від загрозливих програм, заснований на контролі доступу до ресурсів по розширенням і типам файлів. Дослідження актуальності захисту від загрозливих програм і ефективності існуючих методів захисту, показало, що навіть при такому підході до оцінювання можна зробити висновок, що

завдання захисту від загрозливих програм актуальне, а ефективність існуючих засобів захисту низька.

ЛІТЕРАТУРА:

1. Борисов М. А. Основы программно-аппаратной защиты информации. / М. А. Борисов, И. В. Заводцев, И. В. Чижов. – М.: УРСС: Либроком, 2016. – 370 с.
2. Кушнерик О.О. Аналіз методів і засобів захисту від загрозливих програм / О.О.Кушнерик, І.В. Гурман. - Тези доповідей Всеукраїнської науково-практичної конференції молодих вчених, ад'юнктів, слухачів, курсантів і студентів "Молодіжна військова наука у Київському національному університеті імені Тараса Шевченка" [Текст] / за заг. редакцією І.В. Толока. – К. : ВІКНУ, 2018. – С. 121 - 122.

Метод захисту від загрозливих програм, заснований на реалізації контролю доступу до файлових об'єктів

Казіміров В.О., Мостовий С.В., Орленко В.С.
Хмельницький національний університет

Використання сучасних систем інформаційної безпеки вимагає, з одного боку, відстеження швидких змін в інформаційних технологіях та нових загроз, а з іншого - з урахуванням реальних характеристик апаратного та програмного забезпечення корпоративних мереж та систем. Процедура придбання пристроїв захисту інформації проста. Набагато складніше вирішити проблему - як захистити і які заходи безпеки застосовувати, мінімізуючи витрати. Впроваджуючи різні засоби захисту, необхідно визначити баланс між можливим збитком від несанкціонованого витоку інформації та обсягом інвестицій, які витрачаються на забезпечення безпеки інформаційних ресурсів. З метою підвищення ефективності захисту інформаційних ресурсів необхідно дослідити підходи до оцінки рівня їх захисту та систем захисту. Ця оцінка для кожного випадку індивідуальна і залежить від багатьох факторів (вартості інформації, статусу організації, важливості інформації, рівня технічного та програмного забезпечення тощо).

В роботі здійснено дослідження основних типів загрозливих програм, та запропоновано класифікацію шкідливого програмного забезпеченні (ШПЗ) за способом їх виконання. Враховуючи аналіз існуючої статистики зроблено висновок, що найбільш актуальними для захисту є виконувані двійкові і файли сценаріїв.

Можна виділити два найбільш поширених способи зараження: соціальна інженерія; технічні прийоми впровадження ШПЗ, що заражається без відома користувачf [1].

Ці види ШПЗ передбачають обов'язкове збереження файлу на вінчестері перед виконанням.

Тому можна зробити висновок що застосування розмежувальної політики доступу до виконуваних об'єктів, дозволяє мінімізувати загрози.

Проведено дослідження існуючих підходів до оцінки ефективності методів і засобів захисту від загрозливих програм, в результаті якого зроблені

висновки про неможливість з використанням відомих підходів ні кількісно оцінити актуальність окремої загрози для інформаційної системи в цілому (з урахуванням безлічі інших потенційних загроз), в тому числі загрози занесення і запуску загрозливих програм, ні кількісно оцінити основні стохастичні характеристики безпеки системи від загрозливих програм. В результаті чого сформульована задача розробки відповідних математичних моделей і наступного проведення на них досліджень, що дозволяють отримати необхідні кількісні оцінки.

Бінарні та скриптові виконувані файли будуть діяти як об'єкти доступу. Розглянемо варіант, коли всі користувачі мають однакові права доступу.

У Windows найпоширеніші двійкові виконувані файли. Найпоширеніший їх тип - аплікація. Додатки мають розширення EXE і можуть працювати самостійно. Крім того, існують динамічні бібліотеки (їх розширення - DLL), які містять загальні функції для різних додатків. Існують також драйвери (DRV або VXD) - спеціальні програми, необхідні для того, щоб система могла взаємодіяти з конкретними моделями певних пристроїв. Виконувані файли (особливо в Windows) можуть залежати один від одного: наприклад, для запуску будь-якої програми потрібні певні системні динамічні бібліотеки, а вони, в свою чергу, потребують драйверів [2].

Слід зазначити, що виконувані файли містять не тільки самі програми, але і різні додаткові дані. Це можуть бути різні графічні ресурси, що відображаються програмою, тексти написів, описи діалогових вікон тощо. Яскравим прикладом цього є архіви, які містять великі обсяги упаковки з метою зменшення її обсягу при передачі або зберіганні інформації [3].

Отже суб'єктом доступу є будь-який користувач системи $S_i : S = \{S_1, \ldots, S_k\}$. Об'єкти доступу поділяються на виконувані, системні та інформаційні: $O = \{O_{вик1}, \cdots, O_{викq}, O_{сист1}, \cdots, O_{системт}, O_{інф1}, \cdots, O_{інфп}\}$.

Системні файли мають розширення: *.config, *.manifest, *.fon, *.ttf, *.log.

Ми додамо захист від виконуваних файлів сценарію. Для їх виконання потрібно встановити інший інтерпретатор, в якому запуститься виконуваний файл. Основою моделі захисту від виконуваних файлів сценарію є захист від витоку повноважень на зміну функцій дозволеної програми. Особливістю методу є поділ методу доступу "Запис" для створення нового об'єкта доступу («*Зп/І*») та зміни існуючого об'єкта доступу шляхом перейменування («*Зп/*»). Введемо додатково право доступу - заборона читання («*Чт*») [4].

Тоді в моделі будемо використовувати таку множину прав:

$$R = \{Чт, \overline{Чт}, В, Зп, Зп/І, Зп/, Пн, Пн/В, Д\}.$$

Права адміністратора:

- читання, інсталювання і запуск виконуваних файлів;
- читання, інсталювання системних файлів;
- читання і запис інформаційних файлів;
- все інше заборонять.

Опишемо права інших користувачів:

- читання і запуск виконуваних файлів, які вже знаходяться на системному диску;
- заборонити створення нового файлу, зміни існуючого, перейменування існуючого виконуваного файлу, видалення існуючих виконуваних файлів на системному диску;
- читання системних файлів;
- заборонити для системних файлів створення нового, зміну та перейменування існуючого файлу;
- читання і запис інформаційних файлів;
- заборонити запис, перейменування і видалення системних файлі;
- заборонити перейменування і видалення інформаційних файлів;
- заборонити для інформаційних файлів перейменування існуючого файлу, видалення;
- все інше заборонити.

$$M = \begin{array}{c} \\ S_1 \\ \cdots \\ S_k \\ \\ VM_1 \\ \cdots \\ VM_j \\ A \end{array} \begin{array}{c} O_{кат1}, \ldots, O_{катq} \\ \hline Чт, В, ЗпН, ЗпІ, \\ Не, НеВ, Д \\ \cdots \\ Чт, В, ЗпН, ЗпІ, \\ Не, НеВ, Д \\ Чт, В, ЗпН, ЗпІ, \\ Не, НеВ, Д \\ \cdots \\ Чт, В, ЗпН, ЗпІ, \\ Не, НеВ, Д \\ Чт, Зп, В \end{array} \begin{array}{c} O_{сист1}, \ldots, O_{системр} \\ \hline Чт, ЗпН, ЗпІ, \\ Не, НеВ, Д \\ \cdots \\ Чт, ЗпН, ЗпІ, \\ Не, НеВ, Д \\ Чт, ЗпН, ЗпІ, \\ Не, НеВ, Д \\ \cdots \\ Чт, ЗпН, ЗпІ, \\ Не, НеВ, Д \\ Чт, Зп \end{array} \begin{array}{c} O_{інф1}, \ldots, O_{інфn} \\ \hline Чт, Зп, \\ Не, НеВ, Д \\ \\ Чт, Зп, \\ Не, НеВ, Д \\ Чт, Зп, \\ Не, НеВ, Д \\ \\ Чт, Зп, \\ Не, НеВ, Д \\ Чт, Зп \end{array}$$

Запропоновано модель захищеної системи, яка дає можливість сформувати вимоги з точки зору запобігання витоку прав доступу.

Такий підхід дає нам впевненість що такі права доступу не дадуть змогу бінарним і скриптовим виконуваним загрозливим файлам зашкодити

безпеці системи.

Але потрібно забороняти співпадання будь-якого користувача з Адміністратором [5] .

Отже в роботі досліджено існуючі способи впровадження та запуску загрозливих програм, в результаті чого зроблено висновок про те, що найбільш актуальними для захисту є виконувані бінарні і скриптові файли і про те, що дані класи загрозливих програм передбачають обов'язкове збереження загрозливого файлу на жорсткому диску перед його виконанням (читанням). Це дозволило зробити висновок щодо того, що захист від загрозливих програм може будуватися реалізацією контролю (розмежування прав) доступу до файлів.

Запропоновано загальний підхід до реалізації захисту від загрозливих програм, заснований на реалізації контролю доступу до файлів по їх типам, які можуть бути ідентифіковані розширеннями файлів. Можливість використання подібного підходу обгрунтована проведеним дослідженням засобів захисту.

Розглянута модель засобів захисту дозволяє сформулювати вимоги до експлуатаційних параметрів засобів захисту - до інтенсивності виявлення помилок засобами захисту, що дозволяє здійснити успішну атаку, та інтенсивності їх виправлення, виконання яких забезпечує необхідні значення експлуатаційних характеристик засобів захисту.

Перелік посилань

1. Лебедев А. Защита компьютера от вирусов, хакеров и сбоев / Алексей Лебедев. – М.: Питер, 2013. – 157 с.

2. Исполняемые файлы: расширения, форматы. // Справочник типов файлов. [Електронний ресурс]. Режим доступу: http://open-file.ru/types/executable/, свободный (10.10.2020).

3. *Cuff P.* Distributed channel synthesis / P. Cuff // IEEE. Trans. Inf. Theory. – 2013. – Vol. 59(11). – P. 7071-7096.

4. Джулій В.М. Оцінка актуальності загрози впровадження загрозливих програм для інформаційної системи / В.М. Джулій, В.О. Бойчук, О.О. Кушнерик, -Хмельницький: Наука й економіка, 2018. - Вип. № 2. – С.107-115

5. Schieler C. Rate-distortion theory forsecrecy systems / C. Schieler, P. Cuff // IEEE Trans. on Inf. Theory. – 2014. – Vol. 66(12). – P.7584-7605.

6. Тарнавський Ю. А. Технології захисту інформації: підручник / Ю. А. Тарнавський ; КПІ ім. Ігоря Сікорського. – Київ : КПІ ім. Ігоря Сікорського, 2018. – 162 с.

7. Остапов С. Е. Технології захисту інформації : навчальний посібник / С. Е. Остапов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2013. – 476 с.

ДОДАТОК В
(обовязковий)
Презентація

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

КАЗІМІРОВ ВОЛОДИМИР ОЛЕГОВИЧ

УДОСКОНАЛЕННЯ МЕТОДУ ЗАХИСТУ ІНФОРМАЦІЙНИХ РЕСУРСІВ ВІД ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Науковий керівник
к.т.н., доцент Орленко В.С.

кафедра кібербезпеки та комп'ютерних систем і мереж

Тема: Удосконалення методу захисту інформаційних ресурсів від шкідливого програмного забезпечення

Метою магістерської роботи є розвиток методів захисту інформаційних ресурсів шляхом застосування розмежувальної політики доступу до виконуваних об'єктів

Об'єкт дослідження: процес впровадження та запуску шкідливого програмного забезпечення

Предмет дослідження: методи та засоби захисту інформаційних ресурсів на основі розмежування доступу до виконуваних файлів

Задачі досліджень у роботі формулюються наступним чином:

1. Проаналізовано основні типи шкідливого програмного забезпечення запропоновано їх класифікувати за способами завантаження і виконання шкідливих файлів.
2. Досліджено ефективність існуючих методів і засобів захисту, які базуються на сигнатурному і поведінковому аналізі.
3. Вдосконалено моделі і методів захисту від шкідливого програмного забезпечення на основі контролю доступу до файлів.
4. Сформульовано ряд вимог до побудови безпечної системи.
5. Розроблено функціональну схему, політику безпеки, направлених на захист інформаційної системи від шкідливого програмного забезпечення.
6. Проведено оцінку ефективності запропонованих методів захисту з врахуванням впливу на завантаження обчислювальних ресурсів інформаційної системи.

Наукова новизна роботи полягає в:

1. Вдосконалено моделі системи захисту, яка базується на контролі доступу що дозволяє оцінити можливі способи витоку прав доступу і сформулювати вимоги, дотримання яких забезпечує ефективний захист.
2. Розроблено метод захисту інформаційних ресурсів, від завантаження і виконання бінарних і скриптових загрозливих файлів.

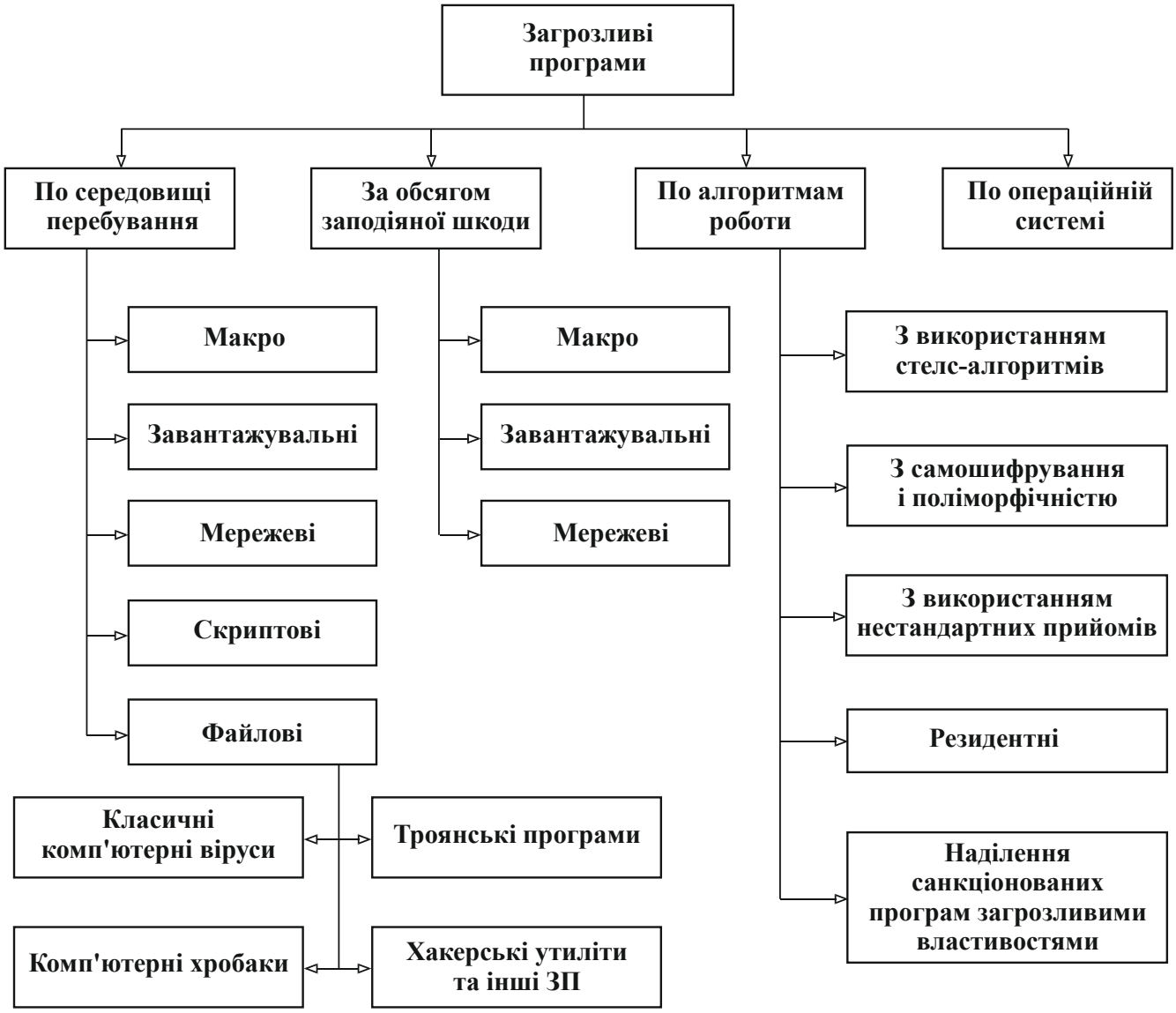
Методика проведення досліджень. При вирішенні поставлених завдань використовувалися методи математичної статистики, теорії масового обслуговування, теорії надійності, теорії графів.

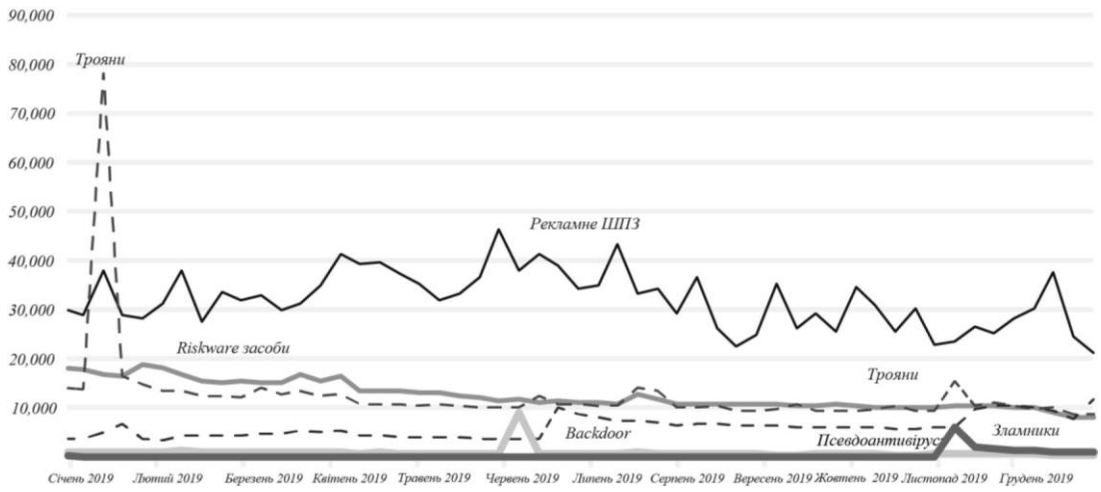
Практична цінність результатів роботи полягає в наступному:

Використовуючи розглянуту модель засобів захисту та сформульовані вимоги, до інтенсивності виявлення помилок засобами захисту, при виконанні яких досягається високий рівень ефективності застосування пропонованого рішення на практиці. Зокрема показано, що при інтенсивності виправлення помилок у засобі захисту, яка становить 3 - 7 днів (ці значення досяжні на практиці), величина ймовірності готовності засобів захисту до безпечної експлуатації становить 0,97 - 0,99, середній час безвідмовної роботи. збій інформаційної безпеки) засобів захисту в цьому випадку становить від шести місяців до двох років.

Публікації. По темі магістерської роботи опубліковано 1 стаття у нефаховому журналі (збірник НПК МНІС ІП-2020), 1 - теза доповідей на всеукраїнській конференції (Тези доповідей XVI Міжнародної науково-практичної конференції "Військоваосвіта і наука: сьогодення та майбутнє" [Текст] / за заг. редакцією Ігоря Толока.– К. : ВІКНУ, 2020)

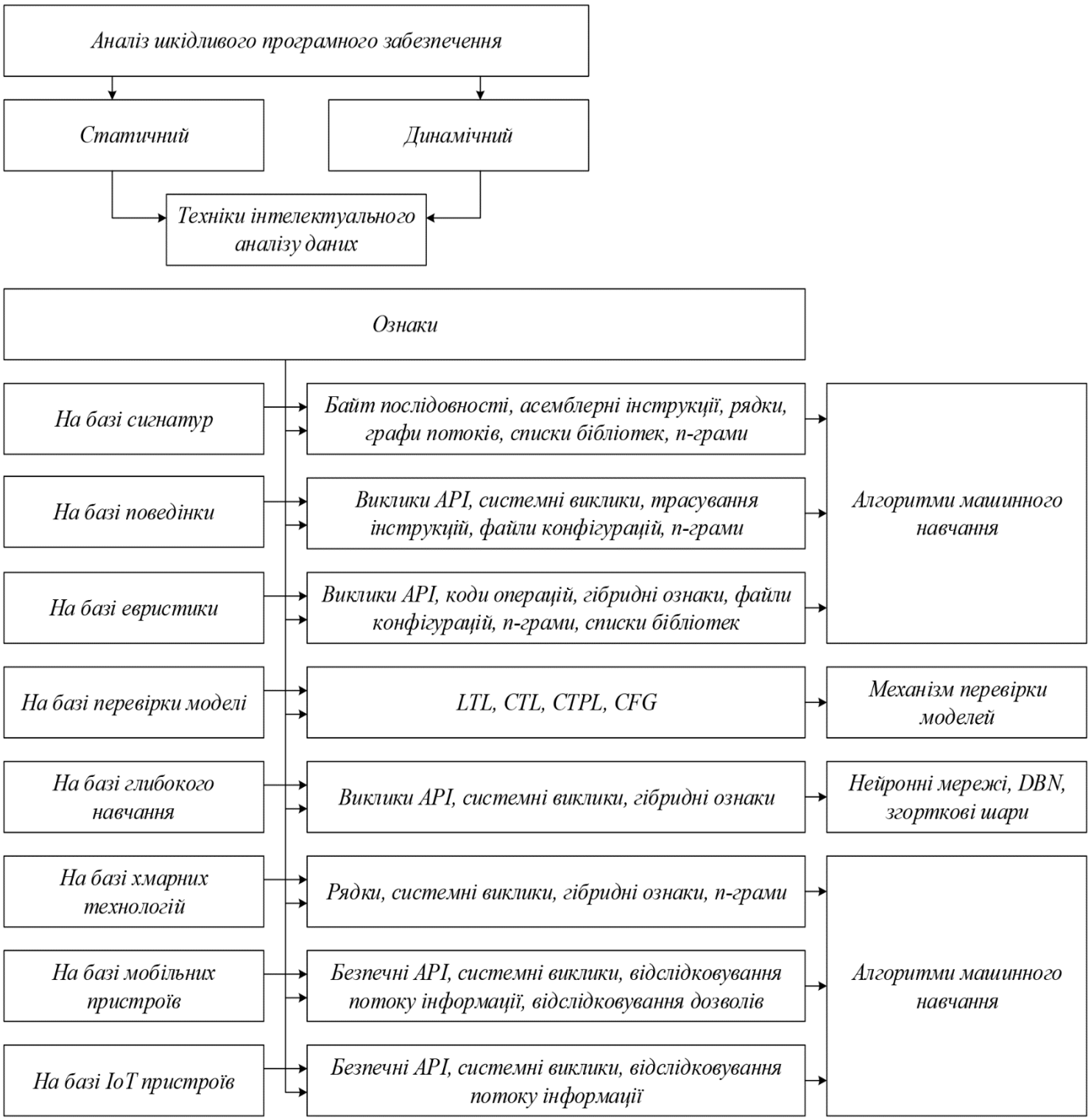
Класифікація шкідливого програмного забезпечення





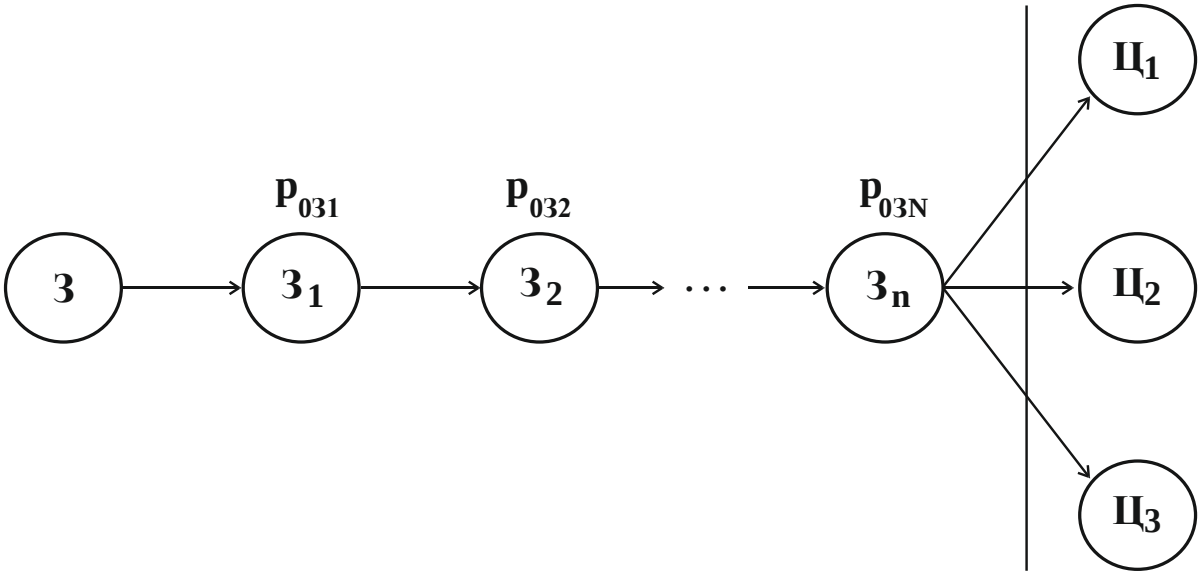
Топ-10 категорій загроз для споживачів у 2020 році

Класифікація методів виявлення шкідливого програмного забезпечення

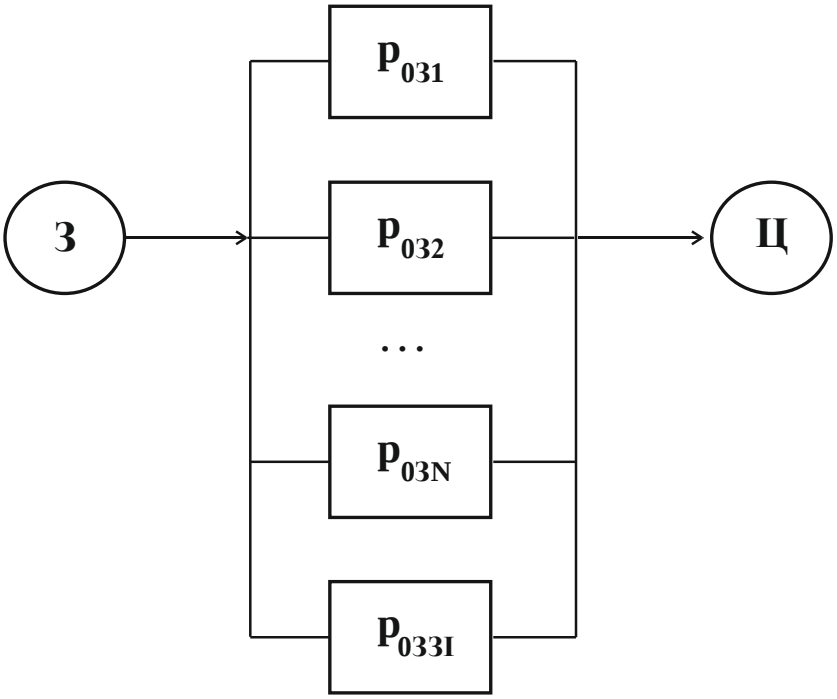


Модель оцінки ймовірності здійснення атаки

Представлення атаки як послідовності загроз



Модель захищеної системи, з доданим засобом захисту



Вірогідність здійснення атаки:

$$p_{0\,здійсн} = \left(1 - p_{0СЗІ}\right) \cdot \prod_{i=1}^{n} \left(1 - p_{0i}\right)$$

Перший науковий результат

Модель системи захисту, яка базується на контролі доступу що дозволяє оцінити можливі способи витоку прав доступу

Формальний опис моделі складається з наступних елементів:

- множина вихідних суб'єктів $S=\{S_1, \dots, S_n\}$
- множина вихідних об'єктів $O=\{O_1, \dots, O_n\}$
- множина прав доступу $R-\{r_1, \dots, r_n\}$
- вихідна матриця доступу M, що містить права доступу суб'єктів до об'єктів.

$S_i : S=\{S_1, \dots, S_k\}$ - користувачі системи

$O=\{ O_{вик}1, \dots, O_{викq}, O_{сист}1, \dots, O_{систm}, O_{інф}1, \dots, O_{інфn}\}$ - об'єкти доступу – поділяються по їх розширенням навиконувані, системні або інформаційні);

Матриця доступу:

$$M= \begin{array}{c} \\ S_1 \\ \dots \\ \dots \\ S_k \end{array} \begin{array}{c} \begin{array}{ccc} O_{вик1}, \dots, O_{викq} & O_{сист1}, \dots, O_{систm} & O_{інф1}, \dots, O_{інфn} \end{array} \\ \begin{bmatrix} Чт, В & Чт & Чт, Зп \\ & \dots & \\ & \dots & \\ Чт, В & Чт & Чт, Зп \end{bmatrix} \end{array}$$

Другий науковий результат

Метод захисту інформаційних ресурсів, від завантаження і виконання бінарних і скриптових загрозливих файлів

Матриця доступу Адміністратора

	$O_{вик1}, \dots, O_{викq}$	$O_{сист1}, \dots, O_{систm}$	$O_{інф1}, \dots, O_{інфn}$
S_1	Чт, В	Чт	Чт, Зп
…		…	
…		…	
S_k	Чт, В	Чт	Чт, Зп
A	Чт,Зп, В	Чт,Зп	Чт, Зп

M=

	$O_{вик1}, \dots, O_{викq}$	$O_{сист1}, \dots, O_{систm}$	$O_{інф1}, \dots, O_{інфn}$
S_1	Чт, В, ~~ЗпН~~, ~~ЗпІ~~, ~~Пв~~, ~~ПвВ~~, ~~Д~~	Чт, ~~ЗпН~~, ~~ЗпІ~~, ~~Пв~~, ~~ПвВ~~, ~~Д~~	Чт, Зп, ~~Пв~~, ~~ПвВ~~, ~~Д~~
…		…	
S_k	Чт, В, ~~ЗпН~~, ~~ЗпІ~~, ~~Пв~~, ~~ПвВ~~, ~~Д~~	Чт, ~~ЗпН~~, ~~ЗпІ~~, ~~Пв~~, ~~ПвВ~~, ~~Д~~	Чт, Зп, ~~Пв~~, ~~ПвВ~~, ~~Д~~
VM_1	~~Чт~~, В, ~~ЗпН~~, ~~ЗпІ~~, ~~Пв~~, ~~ПвВ~~, ~~Д~~	Чт, ~~ЗпН~~, ~~ЗпІ~~, ~~Пв~~, ~~ПвВ~~, ~~Д~~	Чт, Зп, ~~Пв~~, ~~ПвВ~~, ~~Д~~
…		…	
VM_j	~~Чт~~, В, ~~ЗпН~~, ~~ЗпІ~~, ~~Пв~~, ~~ПвВ~~, ~~Д~~	Чт, ~~ЗпН~~, ~~ЗпІ~~, ~~Пв~~, ~~ПвВ~~, ~~Д~~	Чт, Зп, ~~Пв~~, ~~ПвВ~~, ~~Д~~
A	Чт,Зп, В	Чт,Зп	Чт, Зп

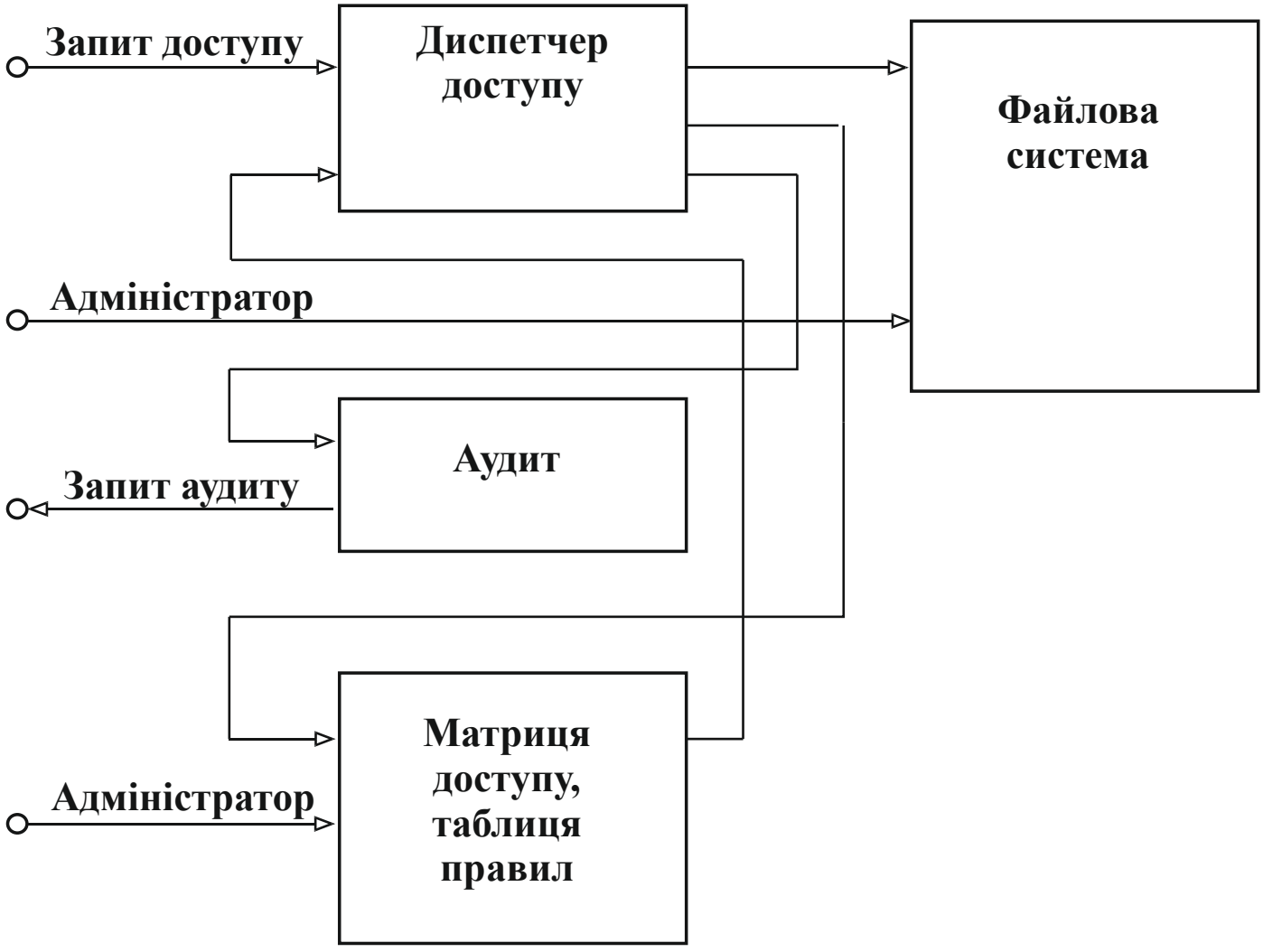
M =

Схема системи захисту

Вимоги о системи захисту:

1. Реалізувати дискреційний механізм контролю доступу.
2. Забезпечити розмежування доступу до суб'єктів.
3. Передбачити примусове управління потоками для виключення сутності «Власник».
4. Реалізувати дозвільну політику доступу.
5. Забезпечити механізм контролю уособлення, який дозволяє забороняти отримання прав інших користувачів.

Функціональна схема системи захисту



Виконання вимог, забезпечує побудову захищеної системи в частині запобігання витоків прав доступу і вдосконалює відомі підходи до реалізації контролю доступу.

Блок-схема алгоритму підвищення безпеки

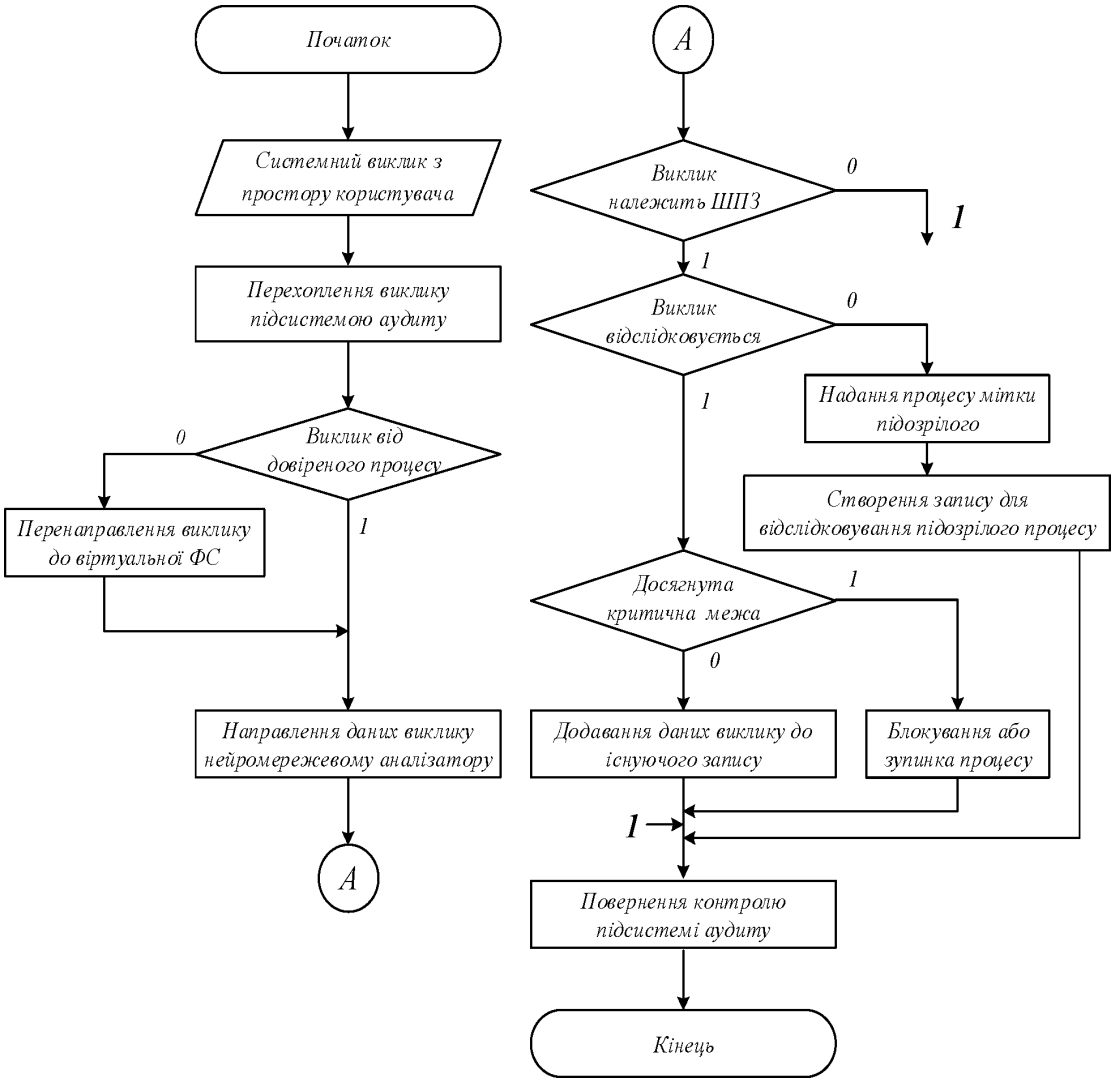
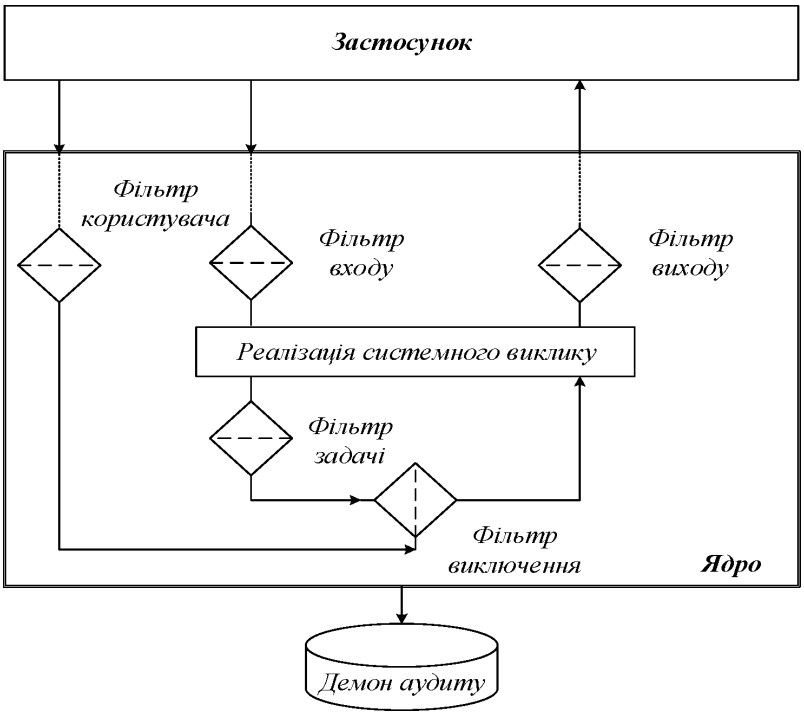


Схема реалізації викликів з простору користувача



ВИСНОВКИ

Дослідження принципу роботи сучасних засобів захисту та особливостей реалізації шкідливого програмного забезпечення надали можливість виокремити ключові недоліки сучасних способів виявлення загроз.

В ході вирішення поставлених завдань було:

1. Використовуючи модель атаки, показано, що найбільш небезпечними загрозами для сучасних інформаційних ресурсів є двійкові файли та шкідливі файли. Зазначається, що ці класи шкідливих програм передбачають обов'язкове збереження загрозливого файлу перед його виконанням (читанням). Це дозволило зробити висновок, що захист від ШПЗ може здійснюватися шляхом контролю (розмежування прав) доступу до файлів за розширеннями та типами файлів. Можливість використання такого підходу обґрунтована вивченням засобів правового захисту.

2. Представлено можливість практичної реалізації розроблених методів захисту від загрозливих програм з урахуванням політики безпеки та обмеження доступу.

3. Використовуючи розглянуту модель засобів захисту та сформульовані вимоги, до інтенсивності виявлення помилок засобами захисту, при виконанні яких досягається високий рівень ефективності застосування пропонованого рішення на практиці. Зокрема показано, що при інтенсивності виправлення помилок у засобі захисту, яка становить 3 - 7 днів (ці значення досяжні на практиці), величина ймовірності готовності засобів захисту до безпечної експлуатації становить 0,97 - 0,99, середній час безвідмовної роботи. збій інформаційної безпеки) засобів захисту в цьому випадку становить від шести місяців до двох років.

4. Вдосконалено метод побудови системи захисту від реалізації та запуску загрозливих програм на основі контролю доступу, моделі та методи захисту інформаційної системи від бінарних та скриптових файлів, на основі реалізації політики розмежування доступу до файлових об'єктів.

5. Проведено випробування та оцінено вплив засобів захисту, що реалізують розроблені методи, на навантаження обчислювального ресурсу, в результаті чого показано, що це набагато менше, ніж вплив на завантаження обчислювального ресурсу антивірусним засобами, зокрема, додаткове навантаження процесора в системі не перевищує 22%, а в звичайному режимі програми не перевищує 16%, тоді як час запуску додатків збільшується не більше ніж на 3 секунди, що майже не впливає на роботу користувача.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «магістр»

Магістр Казіміров В.О.

Тема Удосконалення методу захисту інформаційних ресурсів від ШПЗ

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «магістр»:

кількість листів креслень 10; кількість сторінок записки 89

1. Короткий зміст кваліфікаційної роботи та прийнятих рішень в рамках роботи розроблено моделі системи захисту, яка базується на контролі доступу що дозволяє оцінити можливі способи витоку прав доступу і сформулювати вимоги, дотримання яких забезпечує ефективний захист.

Використовуючи розглянуту модель засобів захисту та сформульовані вимоги, до інтенсивності виявлення помилок засобами захисту, при виконанні яких досягається високий рівень ефективності застосування пропонованого рішення на практиці.

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота ОС «магістр» у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині роботи.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі висвітлюється актуальність теми роботи, дається аналіз досліджуваної проблеми і обгрунтовується застосовуваний підхід до її вирішення, формулюються цілі і завдання дослідження, описується наукова новизна і практична значимість отриманих результатів. У першому розділі проведено аналіз ШПЗ та методів боротьби з ними. Наступні розділи присвячені розробці математичної моделі захисту з використанням матриці розмежування доступу та вдосконалено метод захисту інформаційних ресурсів від ШПЗ Розглянуто питання застосування розробленого методу.

4. Позитивні сторони роботи Кваліфікаційна робота містить ряд інноваційних рішень, зокрема запропонований метод дозволяє дозволяють проводити активну протидію атакам безпосередньо на стороні атакованого ресурсу

5. Негативні сторони роботи Запропонована дискреційна політика безпеки передбачає трудомісткий процес формування матриці доступу та знань системного адміністратора.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи з дотриманням стандартів. В загальному графічне оформлення виконане якісно. Пояснювальна записка відповідає нормам для її оформлення.

7. Відгук про роботу в цілому В загальному кваліфікаційної роботи заслуговує позитивної оцінки. Весь матеріал дипломної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої задачі

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку добре / B (4,50)

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) проф. д. ф.-м. н. Бедратюк М. П. зав. кафедри ІПЗ

« 8 » 12 2020. (підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:
Назва: Удосконалення методу захисту інформаційних ресурсів від шкідливого програмного забезпечення

Автор: Казіміров Володимир Олегович
Спеціальність: 123 – Комп'ютерна інженерія
Освітня програма: Програмування та захист комп`ютерних систем і мереж
Науковий керівник: Орленко Вікторія Сергіївна, к.т.н., доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-35 джерелами на один фрагмент речення;
4) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 4.52% і адресується до 369 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ І.В. Муляр

зав. каф. КБКСМ Клюз Ю.Г.

UNICHECK

User name:
Кафедра кибербезпеки

Check date:
09.12.2020 12:17:18 EET

Report date:
09.12.2020 12:41:34 EET

Check ID:
1005410529

Check type:
Doc vs Internet

User ID:
100005590

File name: **казіміров_на юнічек**

Page count: **87** Word count: **15695** Character count: **119737** File size: **17.01 MB** File ID: **1005702360**

4.52% Matches

Highest match: **0.45%** with Internet source (https://ela.kpi.ua/bitstream/123456789/26203/1/Nafiiev_magistr.docx)

4.52% Internet sources	369	..... Page 89

No Library search was conducted

0% Quotes

Exclusion of quotes is off

Exclusion of references is off

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.

Replaced characters	40

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 3.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 6%**

	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
ID: 81081 Название: Удосконалення методу захисту інформаційних ресурсів від шкідливог програмного забезпечення Добавлено в БД: 2020-11-24 Авторы: Казіміров В.О. Руководители: Орленко В.С. Консультанты: Опоненты:	99960	762	5270 (5%)	52 (7%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы