Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Савчука Владислава Вікторовича

на здобуття ступеня вищої освіти магістра

Метод захисту вебзстосунків від завантаження і виконання підозрілих файлів

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітня програма Кібербезпека та захист інформації

Шифр КРМКБЗІ. 240197.24.01.12 ПЗ

Виконав студент 2 курсу група КБЗІм-24-1 Владислав САВЧУК
Керівник канд. техн. наук, доцент Ігор МУЛЯР
Нормоконтролер д-р. філософії, ст. викладач Наталія ПЕТЛЯК

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

9 12 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Магістр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека та захист інформації

Освітня програма Кібербезпека та захист інформації

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

1 09 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Савчуку Владиславу Вікторовичу

1 Тема роботи Метод захисту вебзастосунків від завантаження і виконання підозрілих файлів

Керівник роботи к.т.н, доц. Ігор МУЛЯР

Затверджено наказом ректора університету 25 08 2025 № 65

2 Строк подання студентом кваліфікаційної роботи на кафедру 01.12.2025

3 Вихідні дані до роботи Здійснити компаративне дослідження методологій забезпечення захисту від шкідливого програмного забезпечення через механізми регулювання доступу до вебресурсів. Визначити рівень актуальності ризиків інтеграції та активації зловмисного коду в межах корпоративної інфраструктури. Сформувати концептуальну модель та розробити методологічний апарат протидії шкідливим програмним компонентам. Провести оцінювання результативності запропонованих методологічних підходів.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. Аналіз сучасних підходів до оцінювання стану інформаційної безпеки кіберфізичних систем. Постановка задачі. Концептуальна модель для визначення ймовірнісних характеристик реалізації кібернападу. Методологічний підхід до забезпечення захисту шляхом контролювання взаємодії із ресурсами інформаційної системи. Структурна схема функціонування захисного механізму. Методика впровадження диференційованої політики безпеки на базі технологій блокування виконання зловмисних програмних елементів. Аналіз результативності забезпечення безпеки інформаційних активів організації. Висновки.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7 Дата видачі завдання 1 09 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Ґрунтовне ознайомлення та дослідження предметної галузі	15.09.2025	Виконано
Визначення змісту, структури магістерської роботи	22.09.2025	Виконано
Опрацювання першого розділу магістерської роботи	29.09.2025	Виконано
Опрацювання статті за результатами дослідження	10.10.2025	Виконано
Опрацювання другого розділу магістерської роботи	20.10.2025	Виконано
Опрацювання третього розділу магістерської роботи	4.11.2025	Виконано
Опрацювання четвертого розділу магістерської роботи	17.11.2025	Виконано
Підготовка та опрацювання ілюстративного матеріалу	24.11.2025	Виконано
Оформлення магістерської роботи графічної та текстової частини	24.11.2025	Виконано
Попередній захист магістерської роботи	27.11.2025	Виконано
Захист магістерської роботи на засіданні ЕК	10.12.2025	Виконано

Студент _____ Владислав САВЧУК

Керівник кваліфікаційної роботи _____ Ігор МУЛЯР

АНОТАЦІЯ

Тема кваліфікаційної роботи: Метод захисту вебзстосунків від завантаження і виконання підозрілих файлів.

Автор роботи: студент групи КБЗІм-24-1 Владислав САВЧУК

Керівник роботи: к.т.н., доц. Ігор МУЛЯР

Загальний обсяг роботи: 105 сторінок, 16 рисунків, 4 таблиці, 2 додатки, 72 посилання.

Ключові слова: ВЕБРЕСУРСИ, КОНТРОЛЬ ДОСТУПУ, РОЗМЕЖУВАННЯ ДОСТУПУ

Метою дослідження є розроблення та обґрунтування нових методів і моделей захисту вебресурсів від шкідливого програмного забезпечення, заснованих на механізмах контролювання доступу до файлових об'єктів, а також створення політик безпеки, що забезпечують попередження проникнення та виконання загрозливих програм у сучасних умовах їх еволюції.

В роботі запропоновані моделі та політики безпеки, які можуть бути впроваджені в існуючі засоби захисту, системи моніторингу, антивірусні платформи та корпоративні системи управління безпекою. Вони забезпечують простоту адміністрування, здатність до масштабування та передбачувану поведінку системи під час блокування потенційно небезпечних операцій з вебресурсами. Запропоновані підходи дозволяють суттєво підвищити рівень захищеності вебзастосунків без залучення ресурсомістких аналізаторів і можуть бути використані для побудови систем з підвищеними вимогами до стійкості проти атак.

Дата 1.12.2025 Підпис студента 

ANNOTATION

Theme of qualification work: Method for protecting web applications from downloading and executing suspicious files

Author of the work: Vladislav SAVCHUK, student of group KBZIm-24-1

Mentor: Ihor MULIAR

Total volume of work: 105 pages, 16figures, 4tables, 2 appendix, 72 references.

Keywords: WEB RESOURCES, ACCESS CONTROL, ACCESS SEGREGATION

The purpose of the study is to develop and justify new methods and models for protecting web resources from malicious software based on mechanisms for controlling access to file objects, as well as to create security policies that prevent the penetration and execution of threatening programs in the current conditions of their evolution.

The work proposes security models and policies that can be implemented in existing protection tools, monitoring systems, antivirus platforms, and corporate security management systems. They ensure ease of administration, scalability, and predictable system behavior when blocking potentially dangerous operations with web resources. The proposed approaches significantly increase the security level of web applications without the use of resource-intensive analyzers and can be used to build systems with increased requirements for resistance to attacks

Date 1.12.2025 Signature

ЗМІСТ

ВСТУП

Загрозливе програмне забезпечення й надалі становить одну з найсерйозніших проблем у сфері сучасної інформаційної безпеки. Дослідження показують, що щодня фіксується понад 450 тисяч нових варіантів шкідливих програм, що створює безпрецедентні труднощі для традиційних систем захисту [1]. Класичні антивірусні інструменти, побудовані на сигнатурному підході, дедалі частіше виявляються неспроможними протидіяти новим та зміненим загрозам, особливо коли зловмисники застосовують методи обфускації чи поліморфізму [2].

Тому особливої ваги набувають превентивні підходи до безпеки, серед яких ключову роль відіграють механізми контролювання доступу до інформаційних ресурсів. На відміну від реактивних методів виявлення шкідливої активності, контроль доступу забезпечує запобігання виконанню небезпечних операцій ще до їх реалізації, обмежуючи дії потенційно шкідливих програм незалежно від їх структури, форми чи способу маскування [3].

З огляду на постійне зростання кількості атак на вебресурси, суттєво посилюється і їхній економічний вплив. У сучасних умовах вразливість інформаційних систем дедалі частіше набуває політичного характеру, що пов’язано як із поширенням гібридних форм протистояння, так і зі зростанням терористичних загроз. Це свідчить про те, що атаки на інформаційні ресурси давно вийшли за межі суто технічної проблеми та стали інструментом впливу на безпеку держав, організацій і суспільства загалом.

У таких умовах підвищення ефективності методів і систем захисту вебресурсів залишається надзвичайно актуальною науковою задачею. Постійний розвиток технік атак, поява нових механізмів обходу захисних систем та ускладнення шкідливого інструментарію зумовлюють необхідність безперервного вдосконалення підходів до забезпечення кібербезпеки. Крім того, практичний аспект проблеми набуває особливої ваги у зв’язку зі зростаючими

економічними втратами, соціальними наслідками та політичними ризиками, що виникають у результаті зловмисних дій.

Питанням дослідження вразливостей вебсистем та розроблення методів їхнього захисту присвячена значна кількість робіт як зарубіжних, так і вітчизняних учених. Серед дослідників, які зробили вагомий внесок у розвиток теорії та практики забезпечення безпеки інформаційних систем, варто відзначити праці Баришева А.Ф., Віртерса Дж., , Гольдштейна Г. Я., Котлера Ф., Ланкіна В.Є., Роджерса Л., Терьохіної К.І. та інших науковців, чиї дослідження стали фундаментом подальшого удосконалення сучасних підходів до захисту інформаційних ресурсів.

Метою дослідження є розроблення та обґрунтування нових методів і моделей захисту вебресурсів від шкідливого програмного забезпечення, заснованих на механізмах контролювання доступу до файлових об’єктів, а також створення політик безпеки, що забезпечують попередження проникнення та виконання загрозливих програм у сучасних умовах їх еволюції.

Для досягнення поставленої мети в кваліфікаційній роботі потрібно вирішити наступні завдання:

– провести аналіз основних типів шкідливого програмного забезпечення та виконати їх класифікацію за характерними способами завантаження й виконання шкідливих файлів;

– удосконалити наявні моделі та методи протидії шкідливому програмному забезпеченню шляхом застосування механізмів контролювання доступу до файлових об’єктів;

– сформулювати комплекс вимог до побудови безпечної інформаційної системи, які забезпечують реалізацію запропонованих методів захисту та уможливлюють створення систем із підвищеним рівнем стійкості до компрометації;

– створити функціональну схему та відповідні політики безпеки, спрямовані на комплексний захист вебресурсів від шкідливих програм;

– виконати оцінювання ефективності запропонованих методів захисту з урахуванням їхнього впливу на використання обчислювальних ресурсів інформаційної системи.

Об’єктом дослідження є процеси виникнення, поширення та виконання ШПЗ в вебсистемах.

Предмет дослідження становлять моделі, методи та політики безпеки вебсистем від виконання підозрілих файлів, побудовані на основі контролювання доступу до файлових ресурсів.

Наукова новизна:

– вдосконалено комплексний метод побудови системи захисту від впровадження та запуску загрозливих програм на основі контролювання доступу з використанням матриці прав;

– запропоновано суб'єкт-орієнтована модель управління доступом, де правила належать суб'єктам, а не об'єктам, що дозволяє використовувати маски та шаблони для визначення категорій файлів;

– сформульовано комплекс вимог до побудови безпечної системи, який змінює традиційні підходи до організації механізмів контролювання доступу та уможливлює уніфіковану реалізацію превентивного захисту від шкідливих програм.

Практичне значення результатів. Розроблені моделі та політики безпеки можуть бути впроваджені в існуючі засоби захисту, системи моніторингу, антивірусні платформи та корпоративні системи управління безпекою. Вони забезпечують простоту адміністрування, здатність до масштабування та передбачувану поведінку системи під час блокування потенційно небезпечних операцій. Запропоновані підходи дозволяють суттєво підвищити рівень захищеності без залучення ресурсомістких аналізаторів і можуть бути використані для побудови систем з підвищеними вимогами до стійкості проти атак.

Публікації. За матеріалами дослідження опубліковано 1 тезу доповіді на науковій конференції.

1 ДОСЛІДЖЕННЯ ПІДХОДІВ ДО ЗАХИСТУ ВЕБЗАСТОСУНКІВ ВІД ЗАГРОЗЛИВИХ ПРОГРАМ

1.1 Аналіз сучасних загроз та обмежень традиційних методів захисту

Сучасний стан загрозливих програм характеризується високим ступенем різноманітності та складності. Шкідливе програмне забезпечення еволюціонувало від простих вірусів, що реплікуються, до складних багатокомпонентних систем, здатних адаптуватися до середовища виконання та уникати виявлення [4]. Програми-вимагачі, що шифрують користувацькі дані та вимагають викуп за їх розшифрування, завдають мільярдних збитків підприємствам та організаціям по всьому світу. Банківські трояни перехоплюють фінансову інформацію, а шпигунське програмне забезпечення збирає конфіденційні дані для промислового шпигунства або державних кібероперацій.

Традиційні антивірусні рішення базуються переважно на сигнатурному методі виявлення, який полягає у порівнянні досліджуваних файлів з базою відомих сигнатур шкідливих програм [5]. Хоча цей підхід залишається ефективним проти відомих загроз, він має фундаментальні обмеження у виявленні нових зразків шкідливого програмного забезпечення. Існує неминуче вікно вразливості між появою нової загрози та оновленням антивірусних баз, протягом якого система залишається беззахисною. Крім того, зловмисники активно використовують техніки модифікації коду, які дозволяють створювати функціонально ідентичні, але структурно різні варіанти шкідливих програм, що ускладнює їх виявлення на основі сигнатур [6].

Евристичний аналіз та методи машинного навчання частково вирішують проблему виявлення нових загроз, аналізуючи поведінку програм та шукаючи підозрілі патерни [7]. Однак ці методи характеризуються високим рівнем хибних спрацювань та можуть бути обмануті складними техніками маскування. Поведінковий аналіз, який моніторить дії програм у реальному часі, також має обмеження, оскільки виявлення загрози відбувається вже після початку її виконання, коли частина шкідливих дій може бути вже здійснена [8].

Загрозливе програмне забезпечення зазвичай класифікують на окремі категорії відповідно до кількох ключових характеристик (рис. 1.1). До основних ознак поділу належать: середовище, у якому функціонує програма; масштаб можливих збитків; специфіка алгоритму її роботи; а також тип операційної системи, на яку спрямовано шкідливий вплив [9].

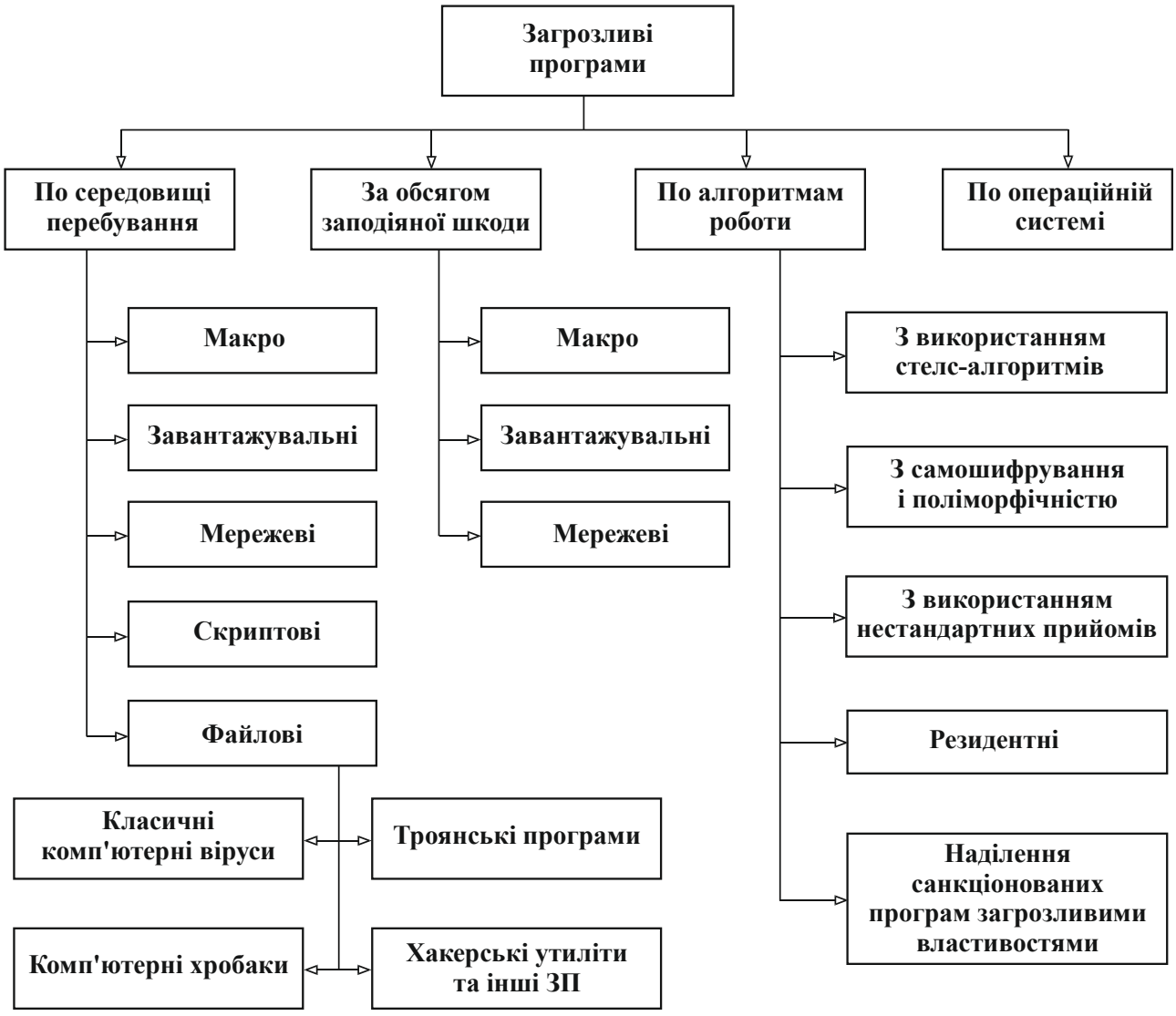

Рисунок 1.1 – Класифікація загрозливих програм [9]

Загрозливі програми-сценарії (скриптові виконувані файли) становлять окремий клас шкідливого ПЗ, для роботи якого необхідний спеціальний інтерпретатор або віртуальна машина. Саме ця особливість відрізняє їх від традиційних файлових загроз. Незважаючи на різницю у способі виконання,

обидві категорії – і скриптові, і файлові ЗП – функціонують у файловому середовищі будь-якої операційної системи. До найпоширеніших типів скриптових загроз належать файли, створені мовами PHP, BAT-файли, а також сценарії VBAx Python, JavaScript [10].

Окрему групу становлять додатки, що працюють за допомогою віртуальних машин. Деякі з них мають плагіни для веббраузерів, що дозволяє запускати застосунки безпосередньо в браузері користувача. Хоча з погляду ОС такі програми не завжди ідентифікуються як виконувані файли, за сутністю вони виконують ті ж функції. Такі загрозливі сценарії зазвичай являють собою звичайні файли, які інтерпретує та запускає віртуальна машина.

Подібні механізми широко застосовуються в атаках типу drive-by завантаження [11]. У цьому випадку зловмисники впроваджують у вебсторінку шкідливі скрипти, які автоматично завантажують і запускають загрозливі програми без відома користувача, що значно підвищує ризики несанкціонованого зараження системи.У цьому контексті механізми контролювання доступу представляють альтернативний підхід до захисту, який не залежить від знання конкретних характеристик загрозливих програм. Замість спроб виявити та заблокувати шкідливе програмне забезпечення, контроль доступу обмежує можливості всіх програм, дозволяючи їм виконувати лише ті операції, які є необхідними для їх легітимного функціонування [12]. Такий проактивний підхід дозволяє ефективно захищатися навіть від невідомих загроз, оскільки будь-яка спроба виконання несанкціонованих дій буде заблокована незалежно від природи програми.

1.2 Функціонування вебсерверів та їх захист на рівні конфігурації

Вебсервер – це програмне забезпечення, яке обробляє HTTP-запити від клієнтів і надсилає у відповідь вебсторінки, файли або інші ресурси [13].

Мережевий рівень відповідає за прийом вхідних з'єднань через TCP/IP

протокол, зазвичай на порту 80 для HTTP та 443 для HTTPS. Сервер прослуховує ці порти та встановлює з'єднання з клієнтами [14].

Обробник запитів аналізує вхідні HTTP-запити, визначає метод запиту GET, POST, PUT, DELETE та інші, перевіряє заголовки та парсить параметри. Цей компонент відповідає за розуміння того, що саме запитує клієнт.

Маршрутизатор визначає, який ресурс або обробник має відповісти на конкретний запит. Він аналізує URL-адресу та направляє запит до відповідного модуля або файлу.

Обробник контенту генерує або витягує запитуваний контент. Це може бути статичний файл HTML, CSS, JavaScript, зображення або динамічно згенерований контент через скриптові мови PHP, Python, Ruby тощо.

Система кешування зберігає часто запитувані ресурси в пам'яті для швидшого доступу, що значно покращує продуктивність сервера та зменшує навантаження.

Apache HTTP Server – один з найпоширеніших вебсерверів з модульною архітектурою. Підтримує величезну кількість модулів для розширення функціональності, працює на різних операційних системах та має потужну систему конфігурації через файли .htaccess [15].

Nginx – високопродуктивний вебсервер з асинхронною архітектурою обробки запитів. Ефективно працює як reverse proxy, балансувальник навантаження та HTTP-кеш. Особливо добре справляється з великою кількістю одночасних з'єднань при мінімальному використанні ресурсів [16].

Microsoft IIS – вебсервер для Windows Server з глибокою інтеграцією з екосистемою Microsoft. Оптимізований для роботи з ASP.NET та іншими технологіями Microsoft [17].

LiteSpeed – комерційний вебсервер з високою продуктивністю та сумісністю з конфігураціями Apache, що дозволяє легко мігрувати з Apache на LiteSpeed [18].

Розглянемо деякі типи загроз для вебсерверів. Веб-шелл – це скрипт, завантажений зловмисником на вебсервер, який дозволяє віддалене виконання

команд. Це один з найнебезпечніших типів атак, оскільки надає повний контроль над сервером [19].

Зловмисники зазвичай завантажують веб-шелли через вразливості завантаження файлів, коли форма на сайті не перевіряє належним чином тип завантажуваних файлів. Інший поширений вектор атаки – експлуатація SQL-ін'єкцій для запису файлів на диск сервера або використання вразливостей у застарілому програмному забезпеченні CMS, плагінів, тем.

Популярні веб-шелли включають WSO Web Shell, C99 Shell, R57 Shell та B374k. Ці інструменти надають графічний інтерфейс для управління файлами, виконання команд, перегляду системної інформації та підключення до бази даних.

SQL-ін'єкція дозволяє зловмисникам втручатися в запити до бази даних. Через недостатню валідацію введених користувачем даних, атакувальник може виконати довільні SQL-команди, отримати несанкціонований доступ до даних, модифікувати або видалити записи в базі даних, а в деяких випадках навіть виконати команди операційної системи [20].

Ін'єкція команд операційної системи виникає, коли веб-додаток передає небезпечні дані як частину системної команди. Це дозволяє виконувати довільні команди на сервері, отримувати доступ до файлової системи та компрометувати всю систему [21].

XSS (Cross-Site Scripting) – атака, при якій зловмисний JavaScript-код впроваджується на веб-сторінку. Існують три типи: відображений XSS передає шкідливий скрипт через URL-параметри, збережений XSS зберігає шкідливий код на сервері наприклад, в коментарях, DOM-based XSS маніпулює DOM на стороні клієнта [22].

Першим кроком у захисті є мінімізація встановленого програмного забезпечення. Необхідно видалити всі непотрібні сервіси, закрити невикористовувані порти та відключити зайві служби, які можуть стати векторами атак.

Регулярне оновлення системи критично важливе. Налаштуйте автоматичні оновлення безпеки, постійно моніторте сповіщення про вразливості та негайно

застосовуйте критичні патчі.

Правильна конфігурація фаєрволу включає дозвіл лише необхідних портів 80, 443, 22 для SSH, блокування підозрілих IP-адрес та регіонів, налаштування rate limiting для запобігання перевантаженню та використання систем запобігання вторгненням [23].

Приховування інформації про сервер зменшує можливості розвідки для атакувальників. Відключіть відображення версії серверного ПЗ, приховайте технології, що використовуються, та налаштуйте кастомні сторінки помилок без технічних деталей.

Обмеження прав доступу має бути ретельно налаштоване. Вебсервер повинен працювати від імені користувача з мінімальними привілеями, файли повинні мати обмежені права доступу лише на читання для більшості, а можливість запису має бути максимально обмежена. Використовуйте окремі директорії для різних типів контенту.

Вимкнення небезпечних функцій включає відключення виконання скриптів у директоріях завантаження, заборону listingu директорій, обмеження методів HTTP лише до необхідних та відключення підтримки застарілих протоколів.

Впровадження SSL/TLS є обов'язковим для захисту даних. Використовуйте сертифікати від довірених центрів сертифікації або безкоштовні сертифікати Let's Encrypt. Налаштуйте автоматичне оновлення сертифікатів та примусове перенаправлення з HTTP на HTTPS.

Конфігурація безпечних cipher suites критично важлива. Вимкніть застарілі протоколи SSLv3, TLS 1.0, TLS 1.1, використовуйте лише сильні алгоритми шифрування та впровадьте Perfect Forward Secrecy для додаткової безпеки [24].

Система завантаження файлів повинна мати багаторівневий захист. Перевірка типу файлу має включати валідацію MIME-типу, перевірку розширення файлу та аналіз справжнього вмісту файлу magic bytes [25].

Зберігання завантажених файлів поза веб-директорією є критично важливим. Файли мають зберігатися в директорії без можливості виконання скриптів, з випадковими іменами для запобігання передбачуваності та в окремій

файловій системі або розділі з обмеженими правами.

Додаткові захисні заходи включають обмеження розміру файлів, сканування антивірусом перед збереженням, використання білого списку дозволених розширень замість чорного та перевірку зображень на вбудовані скрипти.

Системи виявлення змін файлів допомагають ідентифікувати несанкціоновані модифікації. Популярні інструменти включають AIDE Advanced Intrusion Detection Environment, Tripwire, OSSEC та Samhain [26].

Ці системи створюють базу даних контрольних сум файлів, регулярно сканують файлову систему на зміни та сповіщають адміністраторів про модифікації, додавання або видалення файлів.

Моніторинг повинен охоплювати всі веб-директорії, системні файли конфігурації, виконувані файли та скрипти, а також бінарні файли в системних директоріях.

Web Application Firewall (WAF) як фільтр між клієнтом і веб-додатком, аналізуючи HTTP-трафік на предмет шкідливих патернів [27].

Основні можливості WAF включають блокування SQL-ін'єкцій та XSS-атак, захист від CSRF Cross-Site Request Forgery, виявлення та блокування веб-шеллів, запобігання витоку конфіденційних даних та віртуальні патчі для відомих вразливостей.

Популярні рішення WAF є ModSecurity – безкоштовний модуль для Apache, Nginx та IIS з великою базою правил, Cloudflare WAF – хмарне рішення з глобальною мережею, AWS WAF – інтегрований із AWS інфраструктурою, Imperva WAF – комерційне рішення корпоративного рівня [28].

Автоматизовані сканери вразливостей допомагають виявити проблеми до того, як їх знайдуть зловмисники. OWASP ZAP – безкоштовний інструмент для тестування веб-додатків, Nikto – сканер вебсерверів для виявлення небезпечних файлів та конфігурацій, Nessus – комплексний сканер вразливостей, Acunetix – комерційний сканер для виявлення широкого спектру вразливостей.

Регулярність сканування є ключовою: щотижневі автоматичні сканування

для виявлення нових проблем, сканування після кожного великого оновлення або деплойменту та щоквартальні глибокі аудити безпеки з залученням експертів [29].

OWASP Testing Guide є спеціалізованим фреймворком, призначеним переважно для оцінювання безпеки вебдодатків. Методологія охоплює широке коло аспектів, включаючи аналіз серверних налаштувань, тестування механізмів автентифікації, авторизації, обробки введених даних та бізнес-логіки. Значний акцент зроблено на виявленні найбільш поширених уразливостей вебдодатків і дослідженні можливостей їх практичної експлуатації [30].

NIST SP 800-115 – це стандартизована методика, розроблена Національним інститутом стандартів і технологій США, що містить рекомендації щодо організації та проведення тестування безпеки інформаційних систем. Документ приділяє особливу увагу ретельному плануванню робіт, попередній оцінці ризиків, коректному вибору інструментів і методів тестування, а також обов’язковому документуванню отриманих результатів та подальших рекомендацій [31].

1.3 Теоретичні основи контролювання доступу в інформаційних системах

Механізми контролювання доступу представляють альтернативний підхід до захисту, який не залежить від знання конкретних характеристик загрозливих програм. Замість спроб виявити та заблокувати шкідливе програмне забезпечення, контроль доступу обмежує можливості всіх програм, дозволяючи їм виконувати лише ті операції, які є необхідними для їх легітимного функціонування [32].

Такий проактивний підхід дозволяє ефективно захищатися навіть від невідомих загроз, оскільки будь-яка спроба виконання несанкціонованих дій буде заблокована незалежно від природи програми. Автентифікація користувачів повинна бути надійною. Використовуйте багатофакторну автентифікацію для критичних операцій, впроваджуйте політику сильних паролів, захищайте від brute-force атак через rate limiting та блокування після кількох невдалих спроб.

Авторизація має перевірятися на кожному кроці. Ніколи не покладайтеся лише на приховування URL, перевіряйте права доступу на серверній стороні для кожного запиту, впроваджуйте Role-Based Access Control (RBAC) та регулярно аудитуйте права доступу користувачів [33].

Контроль доступу є фундаментальним механізмом забезпечення інформаційної безпеки, який регламентує взаємодію суб'єктів системи з її об'єктами відповідно до визначених політик безпеки [34]. Суб'єктами можуть виступати користувачі, процеси або програми, тоді як об'єктами є файли, каталоги, пристрої, мережеві ресурси та інші компоненти інформаційної системи. Основною метою контролювання доступу є забезпечення того, щоб кожен суб'єкт міг отримати доступ лише до тих об'єктів та виконувати лише ті операції, які є необхідними для виконання його легітимних функцій.

Фундаментальна модель контролювання доступу базується на концепції монітора звернень, який виступає посередником між суб'єктами та об'єктами системи [35]. Кожна спроба доступу перехоплюється монітором, який перевіряє наявність відповідних прав згідно з діючою політикою безпеки та приймає рішення про дозвіл або заборону операції. Для ефективного функціонування монітор звернень повинен задовольняти трьом ключовим вимогам: повнота, ізоляція та верифікованість. Повнота означає, що всі звернення суб'єктів до об'єктів повинні проходити через монітор без можливості їх обходу. Ізоляція передбачає, що сам монітор повинен бути захищений від модифікації або компрометації. Верифікованість означає можливість формального доведення коректності реалізації механізму контролювання доступу.

Класифікація моделей контролювання доступу включає декілька основних типів, кожен з яких має свої переваги та обмеження у контексті захисту від загрозливих програм. Дискреційний контроль доступу надає власникам об'єктів право визначати, хто може отримувати доступ до їхніх ресурсів [36]. Ця модель є найбільш гнучкою, але водночас найменш захищеною, оскільки не перешкоджає несанкціонованому поширенню інформації та може бути обійдена троянськими програмами, що діють від імені легітимного користувача. Мандатний контроль

доступу базується на призначенні міткам безпеки суб'єктів та об'єктів і дозволяє доступ відповідно до строгих правил, що визначаються централізовано [37]. Рольова модель контролювання доступу пов'язує права доступу не безпосередньо з користувачами, а з ролями, які вони виконують в організації, що спрощує адміністрування та дозволяє реалізувати принцип найменших привілеїв [38].

Принцип найменших привілеїв є ключовим для ефективного захисту від загрозливих програм засобами контролювання доступу [39]. Цей принцип постулює, що кожен суб'єкт повинен володіти мінімально необхідним набором прав для виконання своїх функцій. Якщо програма чи процес компрометуються, їхні можливості завдати шкоди будуть обмежені лише тими ресурсами, до яких вони мали легітимний доступ. Реалізація цього принципу вимагає ретельного аналізу реальних потреб кожного компонента системи та систематичного аудиту політик доступу для виявлення надлишкових привілеїв.

Дискреційний контроль доступу є найбільш поширеною моделлю у сучасних операційних системах загального призначення [40]. Основною характеристикою цієї моделі є можливість власника об'єкта самостійно визначати права доступу інших суб'єктів до цього об'єкта. У операційних системах сімейства UNIX та Linux дискреційний контроль реалізується через механізм прав власника, групи та інших користувачів, тоді як у Windows використовуються списки контролювання доступу (ACL), що дозволяють задавати більш деталізовані права [41].

Застосування дискреційного контролювання доступу для захисту від загрозливих програм базується на обмеженні прав доступу для файлів та процесів. Якщо виконуваний файл не має прав на запис у системні каталоги або на модифікацію критичних конфігураційних файлів, навіть у разі його компрометації можливості завдання шкоди будуть істотно обмежені [42]. Однак ефективність дискреційної моделі у протидії шкідливому програмному забезпеченню обмежується кількома фундаментальними проблемами.

Мандатний контроль доступу представляє більш строгий підхід до забезпечення безпеки порівняно з дискреційною моделлю [43]. Основна ідея

мандатного контролю полягає у призначенні міткам безпеки як суб'єктам, так і об'єктам системи, причому рішення про надання доступу приймається виключно на основі співвідношення цих міток згідно з централізовано визначеними правилами. Індивідуальні користувачі не можуть змінювати рівні доступу до об'єктів, що виключає можливість свідомої чи випадкової компрометації політики безпеки.

Класична модель мандатного контролю, відома як модель Белла-ЛаПадули, визначає два основних правила доступу [44]. Правило простої безпеки забороняє суб'єкту читати інформацію з більш високим рівнем конфіденційності, запобігаючи несанкціонованому доступу до секретних даних. Правило зоряної властивості забороняє суб'єкту записувати інформацію на рівень з меншою конфіденційністю, що перешкоджає витоку інформації вниз по ієрархії. Модель Біби, яка є дуальною до моделі Белла-ЛаПадули, фокусується на забезпеченні цілісності інформації та визначає аналогічні правила для запобігання несанкціонованій модифікації даних [45].

Застосування мандатного контролю для захисту від загрозливих програм базується на ізоляції процесів з різними рівнями довіри [46]. Якщо потенційно небезпечна програма виконується з низьким рівнем довіри, вона не може отримати доступ до критичних системних ресурсів або конфіденційних даних користувача, що суттєво обмежує можливості завдання шкоди. Навіть якщо програма намагається виконати шкідливі дії, вони будуть заблоковані механізмом контролювання доступу незалежно від того, чи є програма відомою загрозою.

Рольова модель контролювання доступу (RBAC) представляє прагматичний підхід до управління правами у великих організаціях [47]. Замість безпосереднього призначення прав окремим користувачам, RBAC впроваджує проміжний рівень абстракції у вигляді ролей, які відповідають функціональним обов'язкам співробітників. Користувачі призначаються на ролі, а ролям надаються необхідні права доступу до ресурсів. Така архітектура суттєво спрощує адміністрування, оскільки зміна прав для певної посади вимагає модифікації лише

однієї ролі, що автоматично поширюється на всіх користувачів, які цю роль виконують [48].

Стандарт RBAC визначає ієрархічну структуру, де ролі можуть успадковувати права інших ролей, створюючи дерево привілеїв [49]. Це дозволяє реалізувати організаційні ієрархії та уникнути дублювання при визначенні прав. Крім того, модель підтримує концепцію обмежень, які визначають додаткові умови для активації ролей або виконання певних операцій. Розділення обов'язків може бути реалізоване через взаємовиключні ролі, коли один користувач не може одночасно виконувати дві критичні ролі, що запобігає зловживанням та помилкам.

У контексті захисту від загрозливих програм рольова модель дозволяє мінімізувати права процесів відповідно до їхньої функціональності. Програми можуть виконуватися у контексті спеціалізованих ролей з обмеженими правами, що перешкоджає виконанню несанкціонованих дій у разі компрометації. Наприклад, веб-сервер може виконуватися з роллю, що дозволяє лише читання статичних файлів та прослуховування мережевих портів, але забороняє запис у системні каталоги або виконання довільних команд. Навіть якщо у веб-сервері виявляється вразливість, можливості зловмисника будуть обмежені правами відповідної ролі.

Атрибутивний контроль доступу (ABAC) представляє найбільш гнучку та виразну модель серед сучасних підходів до управління доступом [50]. На відміну від традиційних моделей, що базуються на ідентичності користувачів або їхніх ролях, ABAC приймає рішення на основі атрибутів суб'єктів, об'єктів, операцій та контексту середовища. Атрибутами можуть бути будь-які властивості, такі як підрозділ користувача, час доступу, рівень конфіденційності документа, тип операції, географічне розташування або поточний рівень загрози безпеки системи.

Політики ABAC формулюються у вигляді правил, що визначають умови надання доступу на основі комбінацій атрибутів. Наприклад, правило може дозволяти читання фінансових документів лише співробітникам бухгалтерії у робочий час з корпоративної мережі, блокуючи доступ у всіх інших випадках.

Така гнучкість дозволяє реалізувати складні політики безпеки, що відображають реальні бізнес-вимоги та автоматично адаптуються до змін контексту [51].

Застосування ABAC для захисту від загрозливих програм відкриває нові можливості для динамічного реагування на загрози [52]. Система може враховувати поточний рівень ризику, пов'язаний з певним процесом або користувачем, та відповідно коригувати права доступу. Якщо виявлена підозріла активність, атрибут рівня довіри може бути автоматично знижений, що призведе до обмеження можливостей потенційно компрометованого процесу. Інтеграція з системами виявлення аномалій та аналізу поведінки дозволяє створювати адаптивні політики безпеки, що реагують на зміни загрозового ландшафту.

Контекстна інформація відіграє критичну роль у ефективності ABAC [53]. Час доступу може використовуватися для блокування виконання програм у нетиповий час, що часто є ознакою шкідливої активності. Географічне розташування дозволяє виявляти аномальні спроби доступу з несподіваних локацій. Аналіз мережевої активності процесів може виявити спроби з'єднання з командними серверами або передачі великих обсягів даних, що характерно для витоку інформації.

Політики контролю виконання додатків визначають, які програми можуть бути запущені у системі на основі різних критеріїв. Підхід на основі білих списків дозволяє виконання лише явно дозволених програм, що забезпечує максимальний рівень безпеки, але вимагає ретельного адміністрування. Підхід на основі чорних списків блокує відомі шкідливі програми, але є неефективним проти нових загроз. Гібридні підходи поєднують переваги обох методів, використовуючи білі списки для критичних систем та чорні списки як додатковий рівень захисту.

Технологія Windows AppLocker представляє розвинений механізм контролю виконання додатків у операційних системах Microsoft [54]. AppLocker дозволяє визначати правила на основі шляху до файлу, його цифрового підпису, хеш-суми або атрибутів установника. Політики можуть застосовуватися до різних типів виконуваних файлів, включаючи програми, скрипти, бібліотеки та установники пакетів. Інтеграція з груповими політиками Active Directory дозволяє

централізовано управляти правилами для всіх комп'ютерів у корпоративній мережі.

У Linux системах аналогічну функціональність надають механізми такі як AppArmor та правила виконання SELinux [55]. AppArmor використовує профілі безпеки, що визначають, до яких ресурсів може отримувати доступ кожна програма, автоматично блокуючи будь-які дії за межами визначених правил. Режим навчання дозволяє автоматично генерувати початкові профілі на основі спостереження за нормальною поведінкою програм, що спрощує процес створення політик безпеки.

Контроль виконання коду на рівні процесора, реалізований через технології такі як Intel Control-flow Enforcement Technology (CET) та ARM Pointer Authentication, додає апаратний рівень захисту від експлуатації вразливостей [56]. Ці механізми перешкоджають перенаправленню потоку виконання програми через модифікацію адрес повернення або покажчиків функцій, що є поширеною технікою використовуваною як у експлойтах, так і в шкідливому програмному забезпеченні. Хоча ці технології не є механізмами контролювання доступу у класичному розумінні, вони доповнюють політики безпеки, запобігаючи обходу механізмів захисту через експлуатацію вразливостей.

1.4 Постановка задачі

Проведене дослідження механізмів впровадження загрозливих програм показало, що розглянуті класи шкідливого ПЗ передбачають обов’язкове розміщення відповідного файлу на жорсткому диску перед його виконанням або читанням. Це дало підстави стверджувати, що для протидії найпоширенішим загрозам може бути ефективно застосований контроль доступу до файлових об’єктів, зокрема через впровадження розмежувальної політики доступу.

Також було проаналізовано існуючі підходи до оцінювання ефективності методів і засобів захисту від загрозливих програм. Дослідження показало, що

наявні методики не дають можливості здійснити кількісну оцінку актуальності конкретної загрози для всієї інформаційної системи (з урахуванням множини інших потенційних небезпек), включно з ризиками занесення або запуску шкідливого ПЗ. Крім того, вони не дозволяють визначити ключові стохастичні характеристики безпеки системи щодо загрозливих програм.

У зв’язку з цим сформульовано задачу розробки нових математичних моделей, здатних забезпечити отримання необхідних кількісних оцінок та створити науково обґрунтовану основу для подальшого аналізу та підвищення рівня захищеності вебзастосунків.

Зокрема, важливо виділити завдання, які необхідно вирішити при розробці:

– провести аналіз основних типів шкідливого програмного забезпечення та виконати їх класифікацію за характерними способами завантаження й виконання шкідливих файлів;

– удосконалити наявні моделі та методи протидії шкідливому програмному забезпеченню шляхом застосування механізмів контролювання доступу до файлових об’єктів;

– сформулювати комплекс вимог до побудови безпечної інформаційної системи, які забезпечують реалізацію запропонованих методів захисту та уможливлюють створення систем із підвищеним рівнем стійкості до компрометації;

– створити функціональну схему та відповідні політики безпеки, спрямовані на комплексний захист вебресурсів від шкідливих програм;

– виконати оцінювання ефективності запропонованих методів захисту з урахуванням їхнього впливу на використання обчислювальних ресурсів інформаційної системи.

Ці завдання допомагають створити комплексну картину безпеки вебзастосунків, що дозволяє своєчасно виявити і виправити вразливості до того, як вони можуть бути проесплуатовані зловмисниками.

2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСУ ЗАХИСТУ ВЕБРЕСУРСІВ НА ОСНОВІ РОЗМЕЖУВАННЯ ДОСТУПУ

2.1 Комплексна оцінка актуальності загрози запуску загозливих програм

У сфері інформаційної безпеки ризик розглядається як потенційний негативний вплив, що може виникнути внаслідок експлуатації певної вразливості. Він характеризується двома ключовими параметрами – ймовірністю виникнення небажаної події та масштабом можливих збитків. Відповідно, управління ризиками являє собою структурований процес, який охоплює виявлення, аналіз і зниження ризиків до рівня, що є прийнятним для організації [57].

Для побудови ефективної системи керування ризиками доцільно застосовувати стандартизовані підходи. Однією з найпоширеніших і найбільш методологічно зрозумілих є OWASP Risk Assessment Methodology (OWASP-RAM) [58]. На рисунку 2.1 представлено приклад оцінювання ймовірності використання шкідливого програмного забезпечення (ШПЗ) у контексті аналізу ризиків.

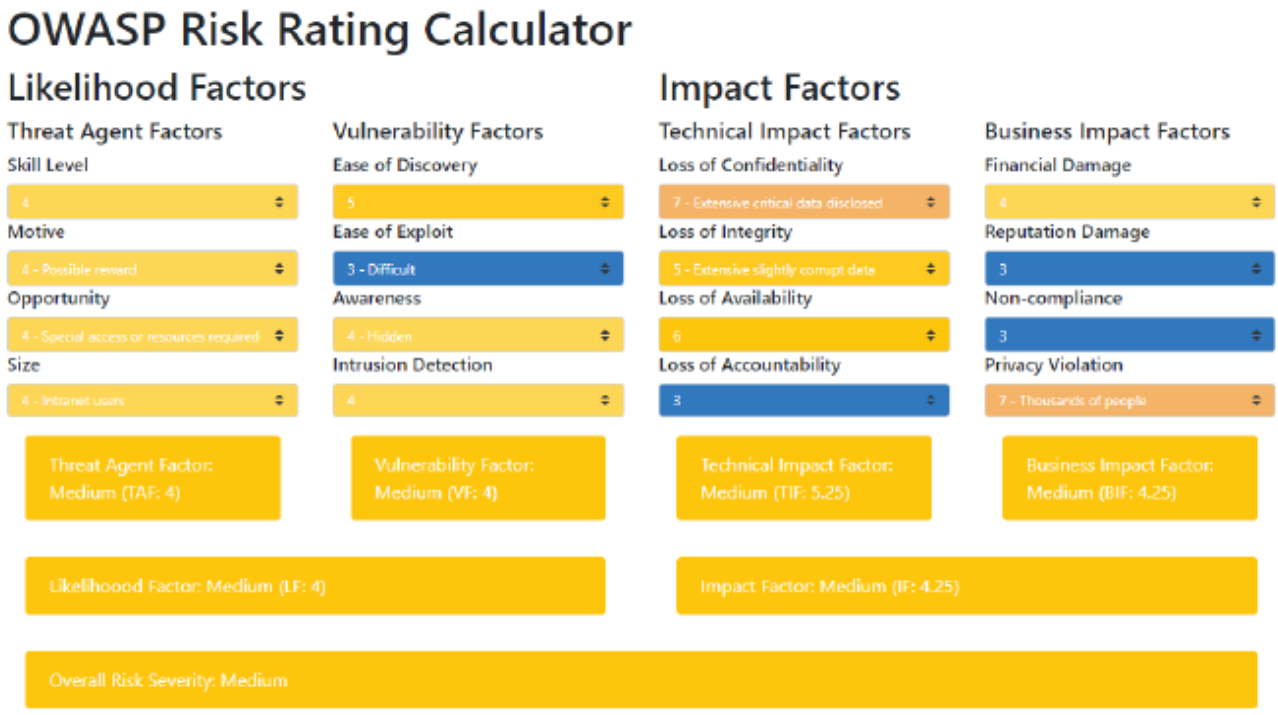

Рисунок 2.1 – Методолігія оцінки ризиків OWASP [58]

Рисунок ілюструє процес визначення ступеня ризику, що виникає внаслідок потенційного проникнення та активації шкідливих програм у системі. Цей приклад показує, як взаємопов'язані різні фактори при оцінці загрози, що дозволяє встановити критичність небезпеки від запуску зловмисного коду в інформаційному середовищі. Використання методології OWASP-RAM у цьому випадку забезпечує обґрунтований вибір пріоритетних заходів безпеки, зокрема систем розмежування доступу, інструментів відстеження поведінки та додаткових правил захисту.

Завдяки систематичному підходу та широкій застосовності OWASP-RAM, отримані результати аналізу ризиків легко інтегруються в комплексні системи управління інформаційною безпекою, що відповідають стандартам ISO/IEC 27005 та NIST SP 800-30. Це створює цілісний підхід до оцінки загроз на всіх етапах – від дослідження окремих уразливостей до розробки загальної стратегії захисту підприємства (табл. 2.1).

Таблиця 2.1 – Оцінка ризиків

Показник	Опис	Оцінка
Складність атаки	Наявність готових інструментів, експлойтів, автоматизація	Висока доступність → *Висока ймовірність*
Рівень привілеїв, необхідних зловмиснику	Виконання шкідливого ПЗ часто можливе без підвищення привілеїв	Середня критичність → *Середня ймовірність*
Інтеракція користувача	Можливість атаки типу drive-by або фішингу	Висока ймовірність помилки → *Висока*
Вплив на конфіденційність	Викрадення або компрометація даних	*Високий вплив*
Вплив на цілісність	Модифікація файлів, пошкодження системи	*Середній–високий вплив*
Вплив на доступність	Можливість блокування систем, шифрування (ransomware)	*Високий вплив*
Загальний рівень ризику	Формується за матрицею	Високий

Отриманий результат свідчить, що загроза запуску шкідливого програмного забезпечення становить критичний ризик для інформаційної системи. Це означає необхідність:

- застосування превентивних засобів контролювання доступу;
- посилення політик безпеки щодо запуску виконуваних файлів;
- моніторингу поведінки програмних процесів;
- мінімізації привілеїв користувачів;
- сегментації середовища;
- впровадження засобів EDR/NGAV.

Виконання шкідливих скриптів у вебзастосунках є однією з найбільш небезпечних загроз, оскільки поєднує у собі як клієнтську, так і серверну вектори атаки. До цієї категорії входять ін’єкційні атаки, кроссайтні скрипти, завантаження модифікованих бібліотек, компрометація серверних плагінів або модулів, а також інфікування вебсервера з подальшим поширенням заражених скриптів на клієнтів. Для оцінки ризику використовується методика OWASP, яка передбачає визначення ймовірності виникнення інциденту та масштабу його впливу на систему.

Ймовірність виконання шкідливого скрипта у вебзастосунку оцінюється як висока, оскільки вебсервіс за замовчуванням взаємодіє з необмеженим колом користувачів і приймає вхідні дані з зовнішніх джерел. Високу ймовірність формує доступність широкого спектра інструментів для автоматизації атак, включно з експлойтами для XSS, SQL/NoSQL ін’єкцій, RCE та SSRF. Зловмисник не потребує високих привілеїв на початковому етапі атаки, оскільки вебсервер сам виконує подані дані згідно з логікою застосунку. Виникає ситуація, коли помилка розробника або несанкціонована зміна конфігурації відкриває можливість автоматичного виконання стороннього коду. Важливим фактором є те, що компрометація вебсервера часто не вимагає дій користувача, а може здійснюватися через вразливі модулі, неправильні дозволи на файловій системі, незахищені директорії завантаження чи відкриті адмін-панелі. Усі ці фактори в сукупності підтверджують високу ймовірність реалізації загрози.

Вплив виконання шкідливого скрипта у вебзастосунку розцінюється як високий через масштаб порушення конфіденційності, цілісності та доступності системи. У випадку клієнтського ін’єктованого скрипта зловмисник отримує можливість викрадення кукі-файлів, токенів авторизації, облікових даних або виконання транзакцій від імені користувача. У випадку серверного виконання шкідливого коду наслідки є значно серйознішими: від несанкціонованої модифікації сторінок, впровадження вебшелів і бекдорів, до ескалації привілеїв та повної втрати контролю над вебсистемою. Окрему загрозу становить поширення зараженого скрипта до інших компонентів інфраструктури – баз даних, файлових сховищ, API-сервісів і зовнішніх інтеграційних систем. Компрометація серверних компонентів також створює довготривалий персистентний ризик через те, що зловмисник може зберігати шалловий доступ або змінювати легітимні файли так, щоб їх неможливо було виявити без глибинного аудиту.

Зараження вебсервера є окремим компонентом оцінки, оскільки його компрометація дозволяє зловмиснику систематично інтегрувати шкідливі скрипти у будь-які сторінки вебзастосунку. У цьому випадку з одного джерела виникає потенційне масове зараження клієнтських систем, а також з’являється можливість формування ботнету або розповсюдження програм-вимагачів через автоматичне завантаження шкідливих файлів. Загальний вплив такого інциденту охоплює фінансові наслідки, репутаційні втрати, зупинку бізнес-процесів, витік критичних даних і неможливість користувачів безпечно взаємодіяти з вебресурсом.

Згідно з матрицею OWASP, поєднання високої ймовірності та високого впливу однозначно формує високий рівень ризику. Ризик виконання шкідливих скриптів у вебзастосунках, особливо з можливістю зараження вебсервера, класифікується як критичний. Це означає, що організація повинна застосовувати превентивні заходи: впровадження жорстких політик фільтрації вхідних даних, регулярне оновлення серверного ПЗ, повну ізоляцію файлових директорій, контроль цілісності, моніторинг виконуваних процесів, а також використання

спеціалізованих засобів захисту – WAF, EDR та інструментів поведінкового аналізу.

Методологія OWASP-RAM пропонує систематизований механізм аналізу загроз, пов'язаних з незакритими вразливостями в програмних продуктах та IT-інфраструктурі. Цей підхід дає змогу визначити вірогідність експлуатації таких слабких місць атакувальниками та оцінити потенційні збитки, що сприяє обґрунтованому визначенню черговості заходів для усунення або зменшення ризиків (рис.2.2).

Початковим етапом є ідентифікація незакритих вразливостей. Процес включає обстеження діючих систем через автоматизовані засоби сканування, проведення експертного тестування або використання репозиторіїв відомих проблем безпеки, як-от CVE (Common Vulnerabilities and Exposures) чи NVD (National Vulnerability Database) [59]. Пріоритетна увага спрямовується на ключові елементи інфраструктури, зокрема додатки з застарілими компонентами, некоректними налаштуваннями або відсутніми оновленнями безпеки.

Подальшим етапом стає визначення вірогідності експлуатації виявлених вразливостей. Методологія OWASP розглядає такі параметри, як доступність готових експлойтів у публічних джерелах, рівень складності проведення атаки, кваліфікацію, потрібну зловмиснику для успішного використання вразливості, та присутність активних кіберзагроз. Зокрема, вразливості з опублікованими proof-of-concept експлойтами характеризуються підвищеною ймовірністю їх практичного використання.

Завершальним етапом є дослідження можливих наслідків. Тут оцінюється, яким чином експлуатація вразливості може порушити конфіденційність, цілісність та доступність системних ресурсів. Приміром, незакрита вразливість у веб-застосунку може створити можливість для SQL-ін'єкції, внаслідок чого відбудеться витік приватної інформації. Слабкі місця в мережевих протоколах здатні полегшити DoS-атаки або надати неавторизований доступ до критичних компонентів інфраструктури.

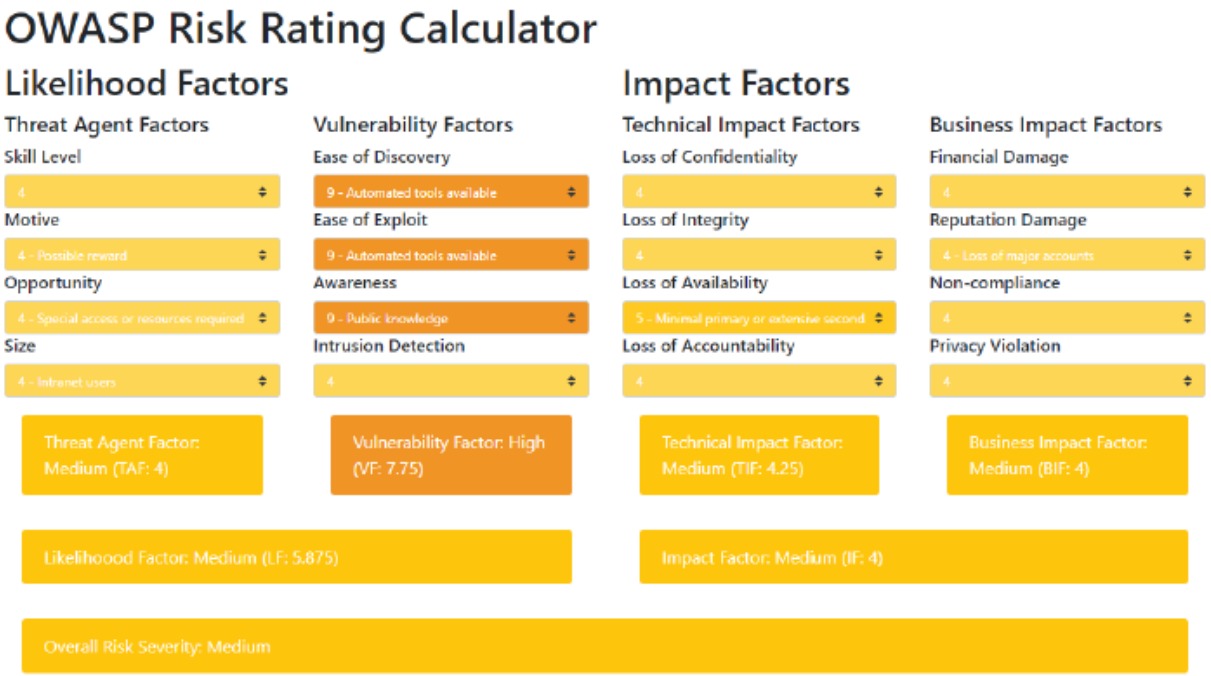

Рисунок 2.2 – Оцінки ризиків виникнення вразливості у програмному забезпеченні чи інфраструктурі [58]

У контексті сучасних вебзастосунків одним із найнебезпечніших типів загроз залишається виконання шкідливих скриптів, що можуть бути впроваджені як у клієнтську частину (XSS, DOM Injection), так і в серверну (Remote Code Execution, Server-Side Injection, Command Injection, LFI/RFI). Особливу загрозу становлять сценарії, які запускаються вебсервером або його компонентами, оскільки саме на сервері зберігаються конфіденційні дані, журнали, ключі доступу та критичні бізнес-процеси.

Випадки зараження вебсерверів через впровадження та виконання шкідливих скриптів найчастіше виникають унаслідок вразливостей у механізмах обробки вхідних даних, помилок конфігурації, невстановлених оновлень, надмірних привілеїв сервісів або наявності інтерпретаторів (PHP, Python, Node.js), які дозволяють виконання небезпечного коду.

У розглянутому сценарії виконання шкідливих скриптів на вебзастосунках визначено одну з найбільш критичних загроз сучасних інформаційних систем. Компрометація вебсервера можлива через завантаження та запуск шкідливих сценаріїв, що виникає внаслідок вразливостей у механізмах обробки вхідних

даних, неправильних конфігурацій та використання зовнішніх компонентів (рис.2.3).

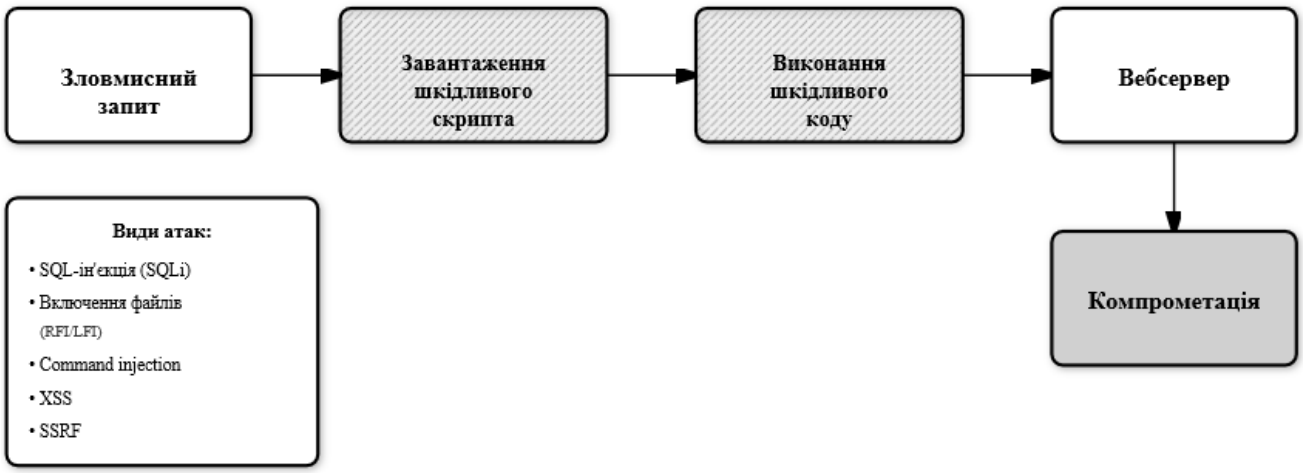

Рисунок 2.3 – Схема зараження вебсервера шкідливими скриптами

Проаналізовано повний ланцюг зараження, починаючи з передачі шкідливого запиту, впровадження скрипта у файлове чи тимчасове середовище, його виконання інтерпретатором та отримання зловмисником контролю над вебсервером. Визначено, що компрометація сервера створює умови для подальшого зараження користувачів шляхом drive-by атак.

На першому етапі зловмисник надсилає шкідливий запит на вебзастосунок, використовуючи відомі вразливості – SQLi, RFI/LFI, Command Injection, XSS у панелі адміністратора, SSRF або вразливі плагіни CMS. Далі відбувається завантаження шкідливого скрипта. Система обробляє дані без належної фільтрації, що дозволяє записати шкідливий файл у директорії вебсервера або в тимчасове файлове середовище [60].

На етапі виконання шкідливого коду вебсервер або інтерпретатор (PHP-FPM, Python interpreter, Node runtime) автоматично виконує файл або команду, запускаючи шкідливий сценарій. Далі відбувається отримання контролю над сервером. Скрипт встановлює бекдор, змінює конфігурації, краде облікові дані, виконує майнінг, змінює вихідний код сторінок, підмінює контент або відкриває доступ до системних ресурсів.

Заражений вебсервер може розповсюджувати шкідливі сценарії кінцевим користувачам (drive-by infection), що переводить атаку у масовий характер (табл.2.2).

Таблиця 2.2 – Оцінка ризику зараження шкідливим скриптом

Параметр	Значення	Пояснення
Ймовірність (Likelihood)	7 / 10	Висока ймовірність виконання шкідливих скриптів через вразливості вебзастосунку або компрометацію вебсервера
Вплив (Impact)	9 / 10	Дуже серйозні наслідки: компрометація даних, контроль над сервером, поширення ШПЗ, підміна контенту
Загальний ризик	63 / 100	Критичний рівень ризику (Critical)
Категорія ризику	Critical	OWASP класифікує значення понад 60 як критичні
Обґрунтування	Ризик високий через можливість зараження вебсервера та виконання шкідливих скриптів, що може торкнутися багатьох користувачів та внутрішніх систем	

На основі OWASP Risk Rating Methodology визначено рівень ризику як критичний. Ймовірність успішної атаки оцінено як високу (7 із 10), що пояснюється широкою доступністю експлойтів, активністю ботнетів та використанням автоматизованих засобів атак. Технічний, фінансовий та репутаційний вплив оцінено як дуже високий (9 із 10), оскільки компрометація вебсервера може призвести до викрадення даних, втрати керування сервісом, зміни контенту та порушення бізнес-процесів.

Підсумковий рівень ризику становить 63 із 100, що відповідає критичній загрозі та потребує негайного впровадження засобів мінімізації ризику. Це включає захист на рівні вебсервера, посилення механізмів контролювання доступу, ізоляцію сервісів, регулярне оновлення програмного забезпечення та повну перевірку вхідних даних. Додатково рекомендовано застосовувати політики безпеки, що запобігають завантаженню та виконанню сторонніх скриптів, а також вести моніторинг активності на сервері для виявлення ознак компрометації.

Основою ідентифікації шкідливих програм в захищених інформаційних системах є підходи до їх виявлення за допомогою антивірусів (рис. 2.4).

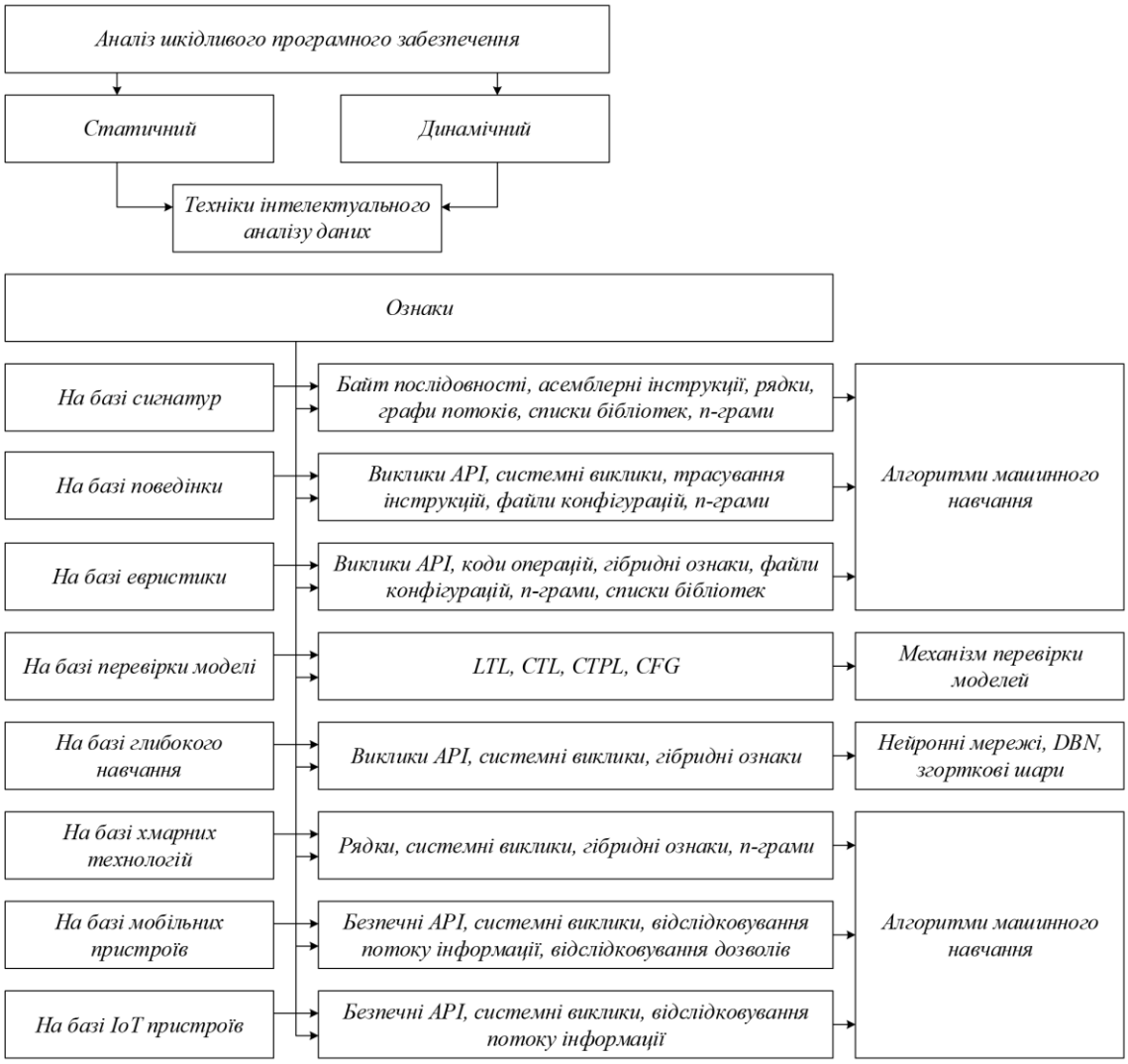

Рисунок 2.4 – Методи виявлення ШПЗ

Хмарне виявлення застосовує різні типи агентів для виявлення в Sandbox на хмарних серверах і надає безпеку як послугу (рис 2.5).

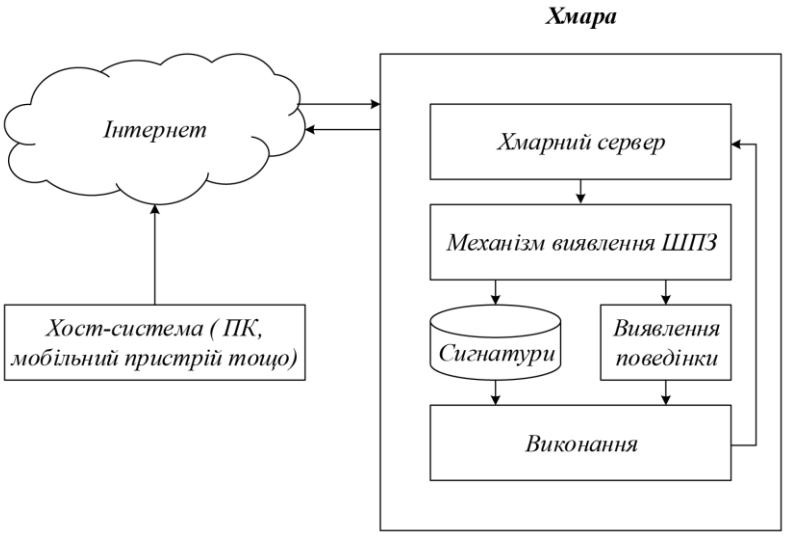

Рисунок 2.5 – Схема виявлення ШПЗ на хмарних сервісах

Підсумовуючи проведену оцінку, можна визначити, що ризик виконання шкідливих скриптів у вебзастосунках, особливо з урахуванням потенційного зараження через вебсервер, є одним із найкритичніших для сучасних інформаційних систем.

Такий рівень ризику свідчить про необхідність негайного впровадження заходів з мінімізації загроз. До ключових напрямів належать посилення контролю над конфігурацією вебсервера, впровадження систем раннього виявлення атак, регулярне оновлення компонентів вебзастосунків та використання превентивних механізмів, включаючи веб-фаєрволи та системи контролю цілісності. Важливим є також проведення регулярного тестування безпеки за стандартизованими методологіями, що дозволяє своєчасно виявляти та усувати вразливості.

Таким чином, управління ризиком виконання шкідливих скриптів повинно бути пріоритетним завданням у рамках загальної стратегії захисту інформаційних ресурсів, адже компрометація вебсервера або вебзастосунку може мати значні економічні, репутаційні та політичні наслідки для організації.

2.2 Моделювання підходу до захисту вебресурсів на основі розмежування доступу

Моделювання підходу до захисту вебзастосунків від імплементації шкідливих скриптів на основі контролювання доступу до ресурсів передбачає формування такої архітектури, у якій будь-яка взаємодія з критичними файлами, сервісами чи конфігураційними об’єктами проходить через формально визначений механізм авторизації. Сутність підходу полягає в тому, що навіть якщо зловмиснику вдається впровадити шкідливий код через вразливість вебдодатка, подальше виконання цього коду обмежується політикою доступу, яка визначає дозволені та заборонені операції для кожного процесу, скрипту або користувача.

Моделювання починається з визначення ресурсів, що можуть бути цілями атак, тобто файлів вебсервера, конфігураційних параметрів, бази даних, тимчасових директорій або директорій завантаження. Для кожного ресурсу встановлюється контекст безпеки та визначається набір операцій, які допускаються під час нормальної роботи системи. Подальший етап полягає у створенні моделі поведінки легітимних процесів вебсерверу. Це дає змогу точно описати, які саме дії є необхідними для функціонування застосунку, а які можуть вказувати на спроби впровадження або виконання шкідливих скриптів.

Контроль доступу у такій моделі працює не як пасивний механізм, а як активний інструмент запобігання атакам. Наприклад, шкідливий скрипт, завантажений через вебінтерфейс, може намагатися створити файл у системній директорії, модифікувати параметри сервера або виконати сторонні програми. Модель доступу блокує такі дії ще до їх виконання, оскільки вони не відповідають дозволеній поведінці процесів вебдодатка. Таким чином, реалізація нападу стає неможливою навіть у випадку успішної експлуатації вразливості.

Моделювання також враховує ймовірність зараження через вебсервер, адже вебсервер часто виступає проміжною ланкою для завантаження та запуску шкідливих компонентів. У системі контролювання доступу серверний процес

отримує обмежений набір дозволів, що унеможливлює запуск довільних файлів або зміну конфігурацій без санкцій. Обмеження на читання, запис і виконання файлів, які завантажуються ззовні, дозволяють суттєво зменшити ризики drive-by attack, RCE-експлуатації, наскрізного зараження та поширення шкідливого коду на суміжні сервіси.

Отже, моделювання підходу на основі контролювання доступу дозволяє не лише ідентифікувати, а й формально довести ефективність обмеження дій, які потенційно може виконати шкідливий скрипт. Захист у такому випадку не залежить від типу, походження або структури шкідливого ПЗ, що значно підвищує стійкість системи. Такий підхід забезпечує додатковий рівень безпеки, зменшує залежність від сигнатурних методів та дає можливість створювати вебзастосунки, де проникнення шкідливих скриптів не призводить до компрометації критичних ресурсів.

Згідно з принципами комплексного підходу до забезпечення інформаційної безпеки, захисна система формується з взаємозалежних елементів, які спільно гарантують безпечне функціонування інфраструктури. До складу цих елементів входять математичні моделі, технічні засоби, програмне забезпечення та кадрові ресурси.

Труднощі у створенні системи захисту інформації виникають не тільки через багатокомпонентність її архітектури, а й через вплив зовнішніх чинників або нетипових дій кіберзлочинців. У зв'язку з тим, що хакери дедалі більше зосереджуються на цифрових інформаційних активах, спостерігається диверсифікація методів атак, що свідчить про розширення арсеналу прийомів кіберзлочинності. Зростаюча зацікавленість у нових векторах кібернападів стимулює розвиток тактик, які застосовують хакери, що додатково ускладнює та урізноманітнює завдання забезпечення інформаційної безпеки.

Для успішного проведення кібератаки зловмисник має спрогнозувати розвиток подій, враховуючи безліч факторів, здатних вплинути на результат нападу. Існує низка математичних підходів, що застосовуються для дослідження

та прогнозування кіберзагроз, надаючи можливість як атакувальникам, так і фахівцям безпеки аналізувати й оцінювати шанси на успішну реалізацію атак.

Імовірнісні моделі відіграють ключову роль у визначенні ризиків та шансів на успіх кібератаки. Ці підходи дозволяють представити атаки як процеси з невизначеними результатами, де вірогідність експлуатації певної вразливості або компрометації системи визначається сукупністю змінних параметрів. Для вивчення результативності атак може бути розроблена модель, що враховує ймовірність несанкціонованого доступу залежно від типів нападів, механізмів захисту та наявних слабких місць [61].

Багатоетапні моделі атак представляють кібернапади як послідовність фаз, кожна з яких характеризується власною вірогідністю успіху чи провалу. Це дає змогу зловмисникам (або захисникам) розраховувати ймовірність того, що атака успішно пройде одну стадію і перейде до наступної. Це охоплює фази збору інформації, первинного проникнення, підвищення привілеїв, застосування шкідливого коду та отримання доступу до конфіденційних ресурсів.

Теоретико-ігрові підходи активно застосовуються для моделювання стратегічних взаємодій, де учасники мають протилежні цілі та тактики. У сфері кібербезпеки це використовується для симуляції протистояння між атакувальниками та захисниками. У такій моделі можна досліджувати стратегії обох сторін, визначаючи найефективніші тактики для зловмисників, що прагнуть проникнути в систему, і для фахівців безпеки, які намагаються відбити атаку [62].

Моделі оцінки вразливостей, зокрема CVSS (Common Vulnerability Scoring System), забезпечують можливість визначення ступеня ризику, пов'язаного з конкретною проблемою безпеки. Кіберзлочинці можуть застосовувати ці моделі для ідентифікації найбільш перспективних вразливостей для експлуатації, беручи до уваги такі параметри, як критичність проблеми, складність її використання та можливі наслідки для системи [62].

Моделі атак на базі мережевої структури представляють мережу як сукупність вузлів та з'єднань між ними, що дозволяє виявити найуразливіші місця в мережевій інфраструктурі. Хакери застосовують подібні моделі для визначення

оптимальних векторів атаки на мережу, враховуючи можливі точки проникнення, маршрути латерального переміщення та методи отримання підвищених привілеїв.

Моделі дослідження динаміки інформаційних систем базуються на диференціальних рівняннях або альтернативних методах для симуляції поведінки IT-інфраструктури в умовах атак. Вони надають розуміння того, як трансформації в системі (наприклад, виявлені вразливості, кібернапади або зміни в діях користувачів) можуть позначитися на її загальному рівні захисту.

Зважаючи на те, що велика частка результативних кібератак реалізується через вплив на користувачів, соціальна інженерія та моделі, що ґрунтуються на психологічних аспектах людської поведінки, також є критично важливими засобами [63]. Моделі соціальної інженерії прагнують спрогнозувати вірогідність того, як користувачів можна ввести в оману або схилити до надання доступу до конфіденційних відомостей чи виконання небезпечних операцій.

Усі ці моделі знаходять застосування не лише для розробки ефективних тактик нападу, а й для побудови більш стійких систем захисту. Вони сприяють фахівцям з кібербезпеки в прогнозуванні потенційних сценаріїв атак та оптимізації захисних заходів для мінімізації ймовірності успішної реалізації кіберзагроз.

Ефективний захист від сучасних загроз вимагає інтеграції механізмів контролювання доступу з системами виявлення та реагування на інциденти. Традиційний статичний контроль доступу, що базується на заздалегідь визначених політиках, може бути недостатнім проти складних цільових атак та загроз нульового дня. Динамічний контроль доступу, що адаптується на основі аналізу поточної ситуації безпеки, дозволяє автоматично підвищувати рівень захисту при виявленні підозрілої активності.

Системи виявлення аномалій аналізують поведінку процесів та користувачів, ідентифікуючи відхилення від встановлених базових показників. Машинне навчання дозволяє автоматично виявляти патерни, що характерні для шкідливої активності, навіть якщо конкретна загроза є новою та невідомою. При виявленні аномалій система може автоматично коригувати політики

контролювання доступу, обмежуючи можливості підозрілих процесів або користувачів до мінімуму. Наприклад, якщо процес починає звертатися до незвично великої кількості файлів, що може свідчити про активність програми-вимагача, його права доступу можуть бути автоматично обмежені до моменту з'ясування ситуації.

Security Information and Event Management (SIEM) системи агрегують інформацію про події безпеки з різних джерел, включаючи журнали контролювання доступу, мережеві пристрої та системи захисту [64]. Кореляція подій дозволяє виявляти складні багатоетапні атаки, що можуть бути непомітні при аналізі окремих компонентів.

Threat intelligence надає контекстну інформацію про актуальні загрози, тактики та процедури зловмисників, що може використовуватися для адаптації механізмів контролювання доступу [65]. Інформація про індикатори компрометації дозволяє превентивно блокувати доступ до відомих командних серверів або заборонити виконання файлів з підозрілими хеш-сумами. Аналіз MITRE ATT&CK framework дозволяє ідентифікувати типові техніки, що використовуються зловмисниками на різних етапах атаки, та налаштувати політики доступу для перешкоджання їх реалізації.

Дослідження існуючих математичних моделей для опису кібератак дає підстави стверджувати, що застосування імовірнісних підходів є найбільш результативним, оскільки вони враховують стохастичність і невизначеність процесів. Особливу увагу заслуговує метод графічного представлення подій, відомий як GERT-мережі [66]. Цей підхід уможливлює моделювання імовірностей, взаємозалежностей між фазами, а також ідентифікацію потенційних сценаріїв еволюції атак.

На наступному етапі відбувається розробка зловмисного програмного забезпечення, орієнтованого на використання виявлених вразливостей. Цей код інтегрується в цільові компоненти інфраструктури, при цьому способи його доставки можуть охоплювати методи соціальної інженерії або експлуатацію програмних недоліків. Після успішного впровадження код запускається,

виконуючи покладені на нього функції, такі як створення бекдору, ініціювання DoS-атак або здобуття несанкціонованого доступу до приватних даних (рис. 2.6).

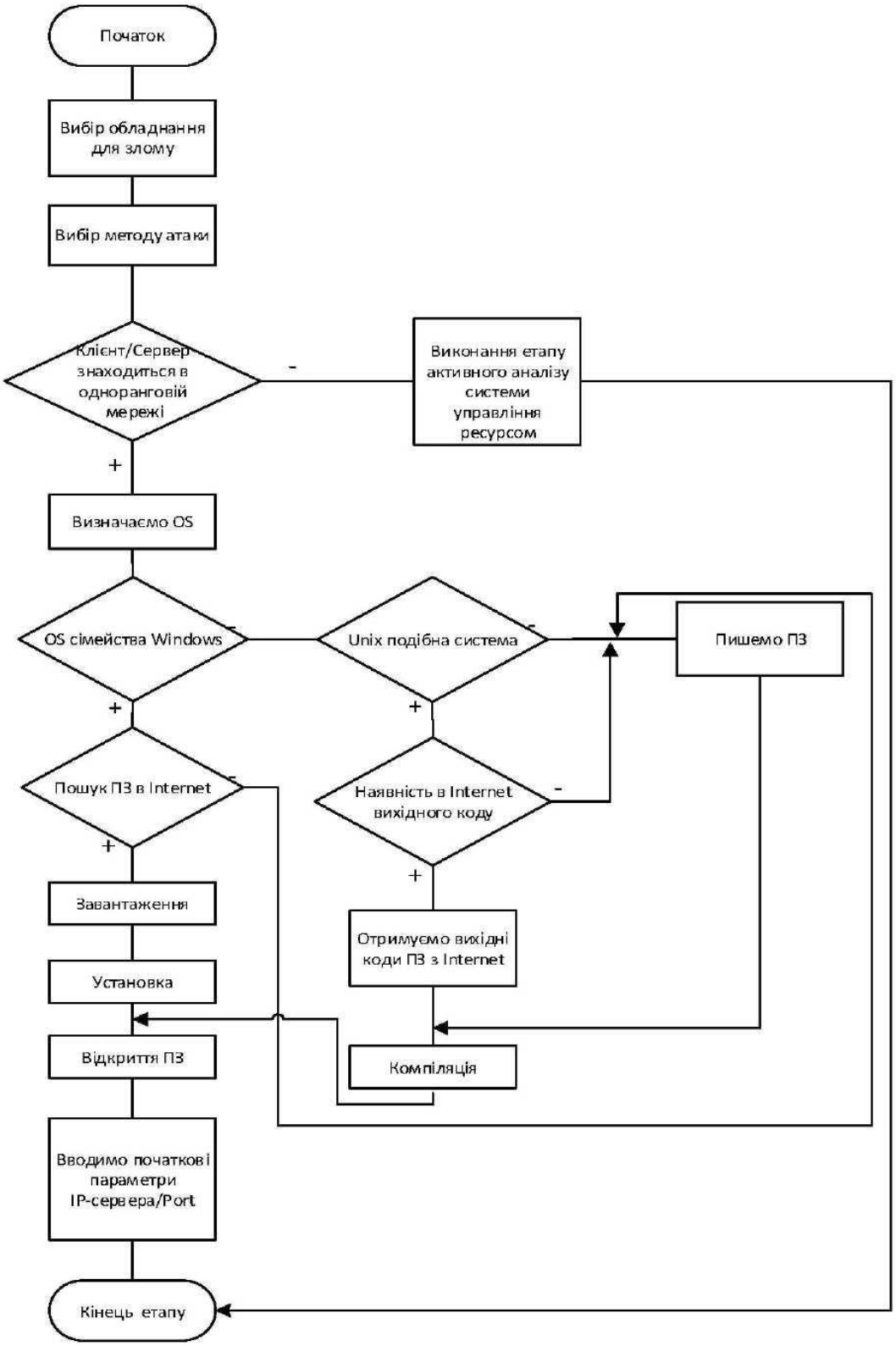

Рисунок 2.6 – Процес атаки

У випадку, коли привілеї зловмисника обмежені, може здійснюватися підвищення рівня доступу, що забезпечує розширений контроль над інфраструктурою. Після цього реалізується кінцева ціль кібератаки, яка може передбачати викрадення, модифікацію або видалення інформації, а також диверсійні дії. Фінальною стадією зазвичай стає усунення слідів присутності або встановлення прихованих каналів для майбутнього доступу.

Вебресурси необхідно захищати від найбільш експлуатованих загроз. На сьогоднішній день актуальною вважається та загроза, вірогідність реалізації якої є найвищою. Пропонується нове визначення актуальності загрози. Виходячи з концепції, що розміщення засобів захисту інформації не є критичним фактором, отримуємо наступне: актуальною є та загроза, усунення якої призводить до нейтралізації максимальної кількості атак. Спираючись на статистичну інформацію, такою загрозою є шкідливі програми.

Актуальність загрози визначається кількістю атак, що уможливлюють перехід у відповідний стан, та кількістю атак, які здійснюються з цього стану. Спочатку обираються загрози з найбільшою кількістю атак, що ведуть до даного стану. Якщо виявляється кілька загроз з однаковою кількістю вхідних переходів, то перевага надається загрозі з максимальною кількістю атак, що виходять з цього стану.

Виходячи з цього твердження введемо критерій актуальності загрози, який бужемо визначати за формулою:

$$K_a = U \cdot S \qquad (2.1)$$

де U - кількість можливих атак, S - кількість потенційно реалізованих атак, через вразливість.

Параметр S має велику вагу, адже чим більше переходів з цієї вершини наорграфі відбувається, тим частіше цю загозу будуть експлуатувати.

На основі цієї інформації (фомула 2.1) представимо кілька атак, що використовують шкідливі програми, на одному графі. Графічне представлення

дозволяє наочно продемонструвати взаємозв'язки між різними типами атак, їх послідовність та точки перетину у процесі реалізації загрози. Це допомагає ідентифікувати критичні вузли, де застосування захисних механізмів буде найефективнішим для блокування максимальної кількості векторів атак.

Граф відображає етапи атак від початкового проникнення шкідливого коду до досягнення кінцевих цілей зловмисника, включаючи проміжні стадії, такі як закріплення в системі, ескалація привілеїв та латеральне переміщення. Кожен вузол графа представляє певний стан системи або етап атаки, а ребра демонструють можливі переходи між станами. Таке представлення дає змогу визначити найбільш критичні точки, де концентрується найбільша кількість атакувальних траєкторій, що обґрунтовує пріоритетність впровадження засобів захисту саме в цих місцях (рис. 2.7).

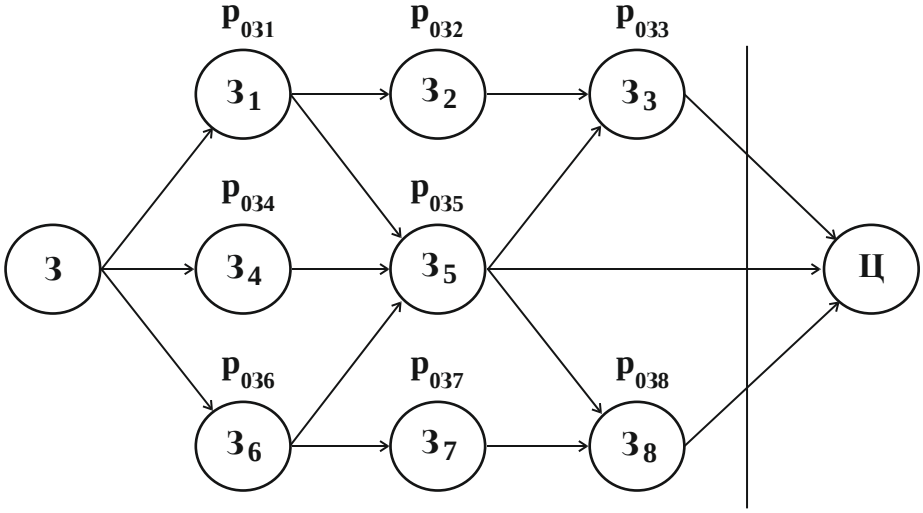

Рисунок 2.7 – Експлуатація однієї загрози

На графі позначимо *З* - зловмисник; *Зі* - вірогідність реалізація загрози; $З_1$ - загроза злому вебзастосунку чи вебсервера; $З_2$ - загроза компроментації паролів; $З_3$ - загроза зараження файлів ОС; $З_4$ – небезпека фішингу та соціальної інженерії; $З_5$ - застосування ШПЗ; $З_6$ - підвищення привілеїв; $З_7$ - вірогідність вразливості в API $З_8$ - загроза вразливості протоколів і мережі; *Ц* – цілі. З графу, згідно фомули 2.1 зробимо висновок, що найбільшу загрозу становить завантаження і виконання шкідливих файлів і скриптів до вебресурсу.

2.3 Метод захисту від завантаженя і виконання підозрілих файлів

Успішне впровадження системи контролювання доступу для захисту від загрозливих програм вимагає систематичного підходу та урахування численних практичних аспектів [67]. Перший етап полягає у проведенні детального аудиту існуючої інфраструктури для визначення критичних ресурсів, що потребують захисту, та аналізу поточних патернів доступу. Інвентаризація всіх інформаційних активів, класифікація їх за рівнем критичності та конфіденційності створює основу для розробки адекватних політик безпеки.

Розробка політик контролювання доступу повинна базуватися на глибокому розумінні бізнес-процесів організації та реальних потреб користувачів і додатків [68]. Надто обмежувальні політики можуть порушити нормальне функціонування систем та викликати опір з боку користувачів, що призведе до спроб обходу механізмів безпеки. Надто дозвільні політики не забезпечать належного рівня захисту та залишать систему вразливою до атак. Баланс між безпекою та зручністю використання є ключовим для успішного впровадження та довгострокової ефективності системи контролювання доступу.

Поетапне впровадження з початковим використанням режиму моніторингу дозволяє виявити потенційні проблеми без порушення роботи критичних систем. У режимі моніторингу система фіксує порушення політик без їх фактичного застосування, дозволяючи адміністраторам проаналізувати легітимність заблокованих дій та відкоригувати правила. Лише після ретельного тестування та впевненості у коректності політик можна переходити до режиму примусового застосування. Пілотні проекти на обмеженій кількості систем дозволяють виявити та вирішити проблеми до масштабного розгортання.

Документування політик безпеки та процедур управління доступом є критично важливим для забезпечення прозорості та аудиту. Кожне правило доступу повинно мати чітке обґрунтування, посилання на відповідні вимоги безпеки або нормативні акти, та визначених відповідальних осіб. Процедури запиту, надання та відкликання доступу повинні бути формалізовані та

автоматизовані наскільки це можливо. Регулярний перегляд політик та прав доступу дозволяє виявляти та усувати надлишкові привілеї, що накопичуються з часом.

Навчання користувачів та адміністраторів є необхідною складовою успішного впровадження системи контролювання доступу. Користувачі повинні розуміти важливість дотримання політик безпеки та процедур запиту доступу. Адміністратори потребують глибоких технічних знань про механізми контролювання доступу, інструменти налагодження та аналізу, а також практичних навичок розробки та управління політиками. Створення внутрішньої експертизи дозволяє організації ефективно підтримувати систему та адаптувати її до змін у інфраструктурі.

У межах запропонованого підходу дискреційна політика безпеки використовується як фундаментальний механізм захисту вебзастосунків від впровадження та виконання шкідливих скриптів. Ключова ідея полягає в тому, що кожний суб’єкт, тобто користувач, процес або служба, а також кожний об’єкт, такий як файл, каталог чи конфігураційний ресурс, мають бути однозначно ідентифіковані. Аксіома ідентифікації дозволяє визначати для кожного суб’єкта точний набір дозволених дій. Права доступу визначаються відповідно до зовнішніх правил, наприклад, політики адміністратора вебсервера або власника інформаційного ресурсу. Саме така індивідуалізація прав і робить дискреційну модель ефективною у протидії шкідливим скриптам, адже навіть якщо зловмиснику вдасться впровадити шкідливий код, можливості його виконання залишаться суворо обмеженими.

Для формалізації прав використовується матриця доступу, яка визначає взаємозв’язки між суб’єктами та об’єктами. Наприклад, у типовій вебсистемі суб’єктом може виступати процес вебсервера, а об’єктами – каталоги з HTML-файлами, робочі директорії для завантажень, конфігураційні файли або системні утиліти. Матриця доступу у спрощеному вигляді наведего на рисунку 2.8.

Суб'єкт	/var/www/html	/var/www/uploads	/etc/nginx/nginx.conf	/usr/bin/bash
www-data (вебпроцес)	r	r,w	–	–

Рисунок 2.8 – Спрощена матриця доступу до вебресурсу

У цьому прикладі вебпроцес "www-data" має лише право читання каталогу з вебсторінками, право читання та запису каталогу завантажень, але не має жодних прав доступу до конфігурацій сервера чи системних виконуваних файлів. Навіть якщо шкідливий скрипт буде завантажений у систему, він не зможе модифікувати конфігурації, запускати інші програми або здійснювати операції з підвищеними привілеями.

Розширена матриця доступу може передбачати декілька рівнів суб'єктів, наприклад адміністратора, системні служби та користувачів, які мають доступ до різних частин системи (рис.2.9).

Суб'єкт	config/	logs/	uploads/	system/
admin	r,w	r,w	r,w	r,w
web-service (API)	–	r	r,w	–
client-session	–	–	r,w	–

Рисунок 2.9 – Розширена матриця доступу до вебресурсу

У такій структурі клієнтська сесія може взаємодіяти лише з каталогом завантажень, вебсервіс має право читати журнали та працювати з файлами користувачів, але не може змінювати конфігурацію або виконувати системні команди. Адміністратор має повний контроль, але його привілеї не передаються автоматично жодному іншим суб'єктам.

Матриця доступу дозволяє чітко побачити, як саме дискреційна політика мінімізує можливість виконання шкідливих скриптів. Якщо процес вебсервера не має права виконання файлів у каталозі завантажень, то навіть скрипт, який був отриманий через атаку, не зможе бути запущений. Якщо ж доступ до конфігураційних файлів повністю закрито, скрипт не зможе змінити поведінку

сервера або перехопити керування. Таким чином практична реалізація ПБ формує суворо регламентоване середовище, де кожний ресурс має чітко визначений режим доступу, а будь-які операції, не передбачені політикою, стають технічно неможливими.

В якості суб'єктів доступу будемо розглядати активні сутності, такі як користувачі системи $S_i : S = \{S_1, \ldots, S_k\}$.

Об'єкти доступу поділимо на виконувані, системні та інформаційні: $O = \{O_{вик1}, \ldots, O_{викq}, O_{сист1}, \ldots, O_{систm}, O_{інф1}, \ldots, O_{інфn}\}$. Множину прав доступу позначимо $R - \{r_1, \ldots, r_n\}$.

Стан системи описується трійкою *(S, O, M)*. В рядках матриці доступу *M* знаходяться суб'єкти і в стовпцях об'єкти. На перетині стовпців і рядків містяться проава суб'єкта *S* до об'єкта *O*, які входять вмножину прав прав *R*.

У межах побудови дискреційної розмежувальної політики формуються чіткі правила щодо дозволених операцій для різних категорій об’єктів доступу. Кінцевий набір прав включає читання, виконання та запис, тобто множину *R = {Чт, В, Зп}*. Ці права призначаються суб’єктам доступу і не є властивостями самих об’єктів, що повністю відповідає класичній концепції дискреційної політики. При цьому застосовується дозвільний принцип: будь-яка дія, яка не визначена як дозволена відповідно до встановлених правил, вважається забороненою.

Суть підходу полягає у введенні строго регламентованого набору дозволених дій для різних типів об’єктів доступу. Виконувані об’єкти, до яких належать скрипти, бінарні файли та службові модулі, можуть бути лише прочитані й виконані. Така конфігурація унеможливлює їх модифікацію, що суттєво знижує ризик підміни легітимного виконуваного файлу на шкідливий. Системні об’єкти, зокрема конфігурації вебсервера, службові бібліотеки та компоненти операційної системи, доступні лише для читання. Це гарантує, що жоден процес, пов’язаний із вебзастосунком, не зможе змінити критичні параметри функціонування системи. Для інформаційних об’єктів, таких як файли

завантаження, журнали або тимчасові сховища, дозволяється читання та запис. У цьому випадку виконання заборонене, що усуває можливість запуску завантаженого шкідливого коду.

Загальна логіка політики полягає в тому, що кожний тип об’єкта має суворо обмежений набір допустимих операцій. Виконання дозволяється лише тим ресурсам, які апріорі мають бути виконуваними і пройшли контроль розгортання. Запис дозволяється лише тим об’єктам, для яких зміна інформації є необхідною частиною функціонування системи. Усі інші операції, що виходять за рамки визначених наборів, блокуються автоматично.

Запровадження такого підходу створює надійний механізм протидії впровадженню шкідливих скриптів. Навіть якщо зловмиснику вдасться завантажити файл на вебсервер, суворе обмеження прав не дозволить йому бути виконаним. Якщо ж шкідливий код буде намагатися модифікувати конфігурацію або підміняти системні файли, такі спроби залишаться безуспішними через відсутність відповідних прав. У результаті система отримує багато-рівневий захист, що базується не на аналізі вмісту файлів, а на регулюванні можливостей їх виконання. Саме це робить політику ефективною навіть проти нових або невідомих типів шкідливого програмного забезпечення, обходячи обмеження сигнатурних і поведінкових методів.

Розглянемо для прикладу матрицю доступу з Адміністратором.

$$M = \begin{array}{c} \\ S_1 \\ \cdots \\ \cdots \\ S_k \\ A \end{array} \begin{array}{c} \begin{array}{ccc} O_{вик1}, \ldots, O_{викq} & O_{сист1}, \ldots, O_{системm} & O_{інф1}, \ldots, O_{інфn} \end{array} \\ \begin{bmatrix} Чт, В & Чт & Чт, Зп \\ & \cdots & \\ & \cdots & \\ Чт, В & Чт & Чт, Зп \\ Чт, Зп, В & Чт, Зп & Чт, Зп \end{bmatrix} \end{array} \qquad 2.2$$

Надамо Адміністратору наступні права:

– дозволити читання, інсталяція та запуск виконуваних файлів;

– дозволити читання та модифікація інформаційних файлів;

– дозволити читання і встановлення системних файлів;

– всі інші операції заборонені.

Визначимо права інших користувачів:

– дозволити читання та запуск виконуваних файлів, які вже присутні на системному диску;

– дозволити читання та запис інформаційних файлів;

– дозволити читання системних файлів;

– всі інші операції заборонені.

У цій ситуації при дослідженні трансформацій стану системи не враховуємо дії адміністратора, оскільки він має можливість деактивувати засоби захисту. Крім того, при даній політиці безпеки виключена сутність «Власник», отже користувач, який створив новий об'єкт, не зможе надати право доступу «Запис» іншим користувачам. При формуванні нового суб'єкта доступу також неможливе розповсюдження права доступу «Запис», оскільки розмежування застосовується до будь-якого користувача незалежно від його імені чи ідентифікатора безпеки (SID) [69]. У випадку ескалації привілеїв звичайного користувача до рівня Адміністратора можливе розповсюдження прав відносно суб'єкта доступу.

Перевагою даного підходу є можливість адміністратора інсталювати нове програмне забезпечення або оновлювати наявне.

Недоліком даного методу є ймовірність отримання адміністративних привілеїв і як наслідок необхідність додаткового захисту від несанкціонованої ескалації повноважень до адміністративного рівня.

SID (Security Identifier) – це унікальний ідентифікатор безпеки, який використовується в операційних системах сімейства Windows для ідентифікації користувачів, груп та інших об'єктів безпеки. SID є фундаментальним елементом моделі безпеки Windows і застосовується для контролювання доступу до ресурсів системи.

SID представляє собою змінну за довжиною структуру, яка складається з кількох компонентів:

Версія SID – номер версії структури ідентифікатора, зазвичай має значення 1.

Ідентифікатор органу видачі – визначає орган, який створив SID. Наприклад, значення 5 означає, що SID створено системою безпеки Windows NT.

Субідентифікатори домену – серія чисел, що ідентифікують домен або локальний комп'ютер. Зазвичай містить три або більше значень.

Відносний ідентифікатор (RID) – унікальне число в межах домену або комп'ютера, що ідентифікує конкретного користувача, групу або об'єкт.

Приклад SID: S-1-5-21-3623811015-3361044348-30300820-1013

Розшифровка:

S – префікс, що вказує на SID

1 – версія

5 – ідентифікатор органу видачі (NT Authority)

21-3623811015-3361044348-30300820 – ідентифікатор домену

1013 – RID конкретного користувача

Вбудовані SID (Well-known SIDs) – стандартні ідентифікатори, які мають однакове значення в усіх системах Windows. Приклади включають:

- S-1-5-18 – Local System (система)
- S-1-5-19 – Local Service (локальна служба)
- S-1-5-20 – Network Service (мережева служба)
- S-1-5-32-544 – Administrators (адміністратори)
- S-1-1-0 – Everyone (всі користувачі)

Доменні SID – ідентифікатори користувачів та груп у доменному середовищі Active Directory. Вони містять ідентифікатор домену та унікальний RID.

Коли користувач входить у систему, створюється маркер доступу (Access Token), який містить SID користувача, SID всіх груп, до яких він належить, та інформацію про привілеї. Цей маркер супроводжує всі процеси та потоки, запущені від імені користувача.

При спробі доступу до будь-якого захищеного об'єкта (файлу, реєстру, принтера тощо) система безпеки Windows виконує перевірку прав доступу,

порівнюючи SID з маркера доступу із записами в дескрипторі безпеки (Security Descriptor) об'єкта.

DACL (Discretionary Access Control List) – список контролювання доступу, що визначає, які користувачі або групи мають які права доступу до об'єкта [70]. DACL містить записи ACE (Access Control Entries), кожен з яких включає: SID користувача або групи, тип доступу (дозвіл або заборона), набір прав доступу (читання, запис, виконання тощо).

Коли процес намагається отримати доступ до об'єкта, відбувається наступне:

Ідентифікація суб'єкта – система витягує SID з маркера доступу процесу, включаючи SID користувача та всіх груп.

Далі відбувається послідовний аналіз ACE – система переглядає записи ACE в DACL у порядку їх розміщення. Спочатку перевіряються записи типу "заборона" (Deny). Якщо знайдено запис Deny, що відповідає SID з маркера доступу, і цей запис забороняє запитуваний тип доступу, доступ відхиляється негайно. Якщо записів Deny немає або вони не застосовуються, система перевіряє записи "дозвіл" (Allow). Прийняття рішення – якщо накопичені права задовольняють запит на доступ, він надається; якщо ні – відхиляється.

Складність управління – SID є непрозорими для людини ідентифікаторами, що ускладнює адміністрування без спеціальних інструментів. Проблема "сирітських" SID – якщо користувач або група видаляються, їхній SID може залишитися в дескрипторах безпеки, створюючи незрозумілі записи в ACL. Можливість спуфінгу – у певних випадках зловмисник може створити об'єкт з тим самим SID, що й видалений раніше об'єкт, отримавши таким чином доступ до ресурсів. Обмеження міграції – при переміщенні користувачів між доменами їхні SID змінюються, що вимагає оновлення всіх ACL.

Отже система повинна контролювати процес створення нових SID та запобігати дублюванню критичних ідентифікаторів. Має проводитися аудит змін SID – регулярний моніторинг журналів безпеки для виявлення підозрілих операцій зі зміни або використання SID.

Далі здійснювати контроль привілеїв – обмеження можливості звичайних користувачів отримувати інформацію про SID інших користувачів або маніпулювати дескрипторами безпеки, та вровадити механізми, що запобігають несанкціонованій зміні маркерів доступу або SID у них.

У контексті захисту від шкідливих програм розмежування доступу на основі SID відіграє критичну роль:

– кожен процес працює з маркером доступу, що містить конкретний набір SID, що обмежує його можливості взаємодії з системними ресурсами;

– навіть якщо шкідлива програма отримує контроль над процесом користувача, її дії обмежені правами доступу, визначеними для SID цього користувача;

– системні та виконувані файли можуть бути захищені через DACL, що дозволяють запис лише для адміністративних SID;

– аналіз спроб доступу до ресурсів з незвичних SID може сигналізувати про компрометацію системи.

Отже додамо до наших правил і матриці розглянуті особливості. Крім звичайної множини прав доступу (Allow), введемо додаткові методи доступу, які будутьозначати заборону (Deny). Напрклад, заборону перейменування позначимо «~~*П*~~», запису – «~~*Зп*~~», видалення – «~~*Д*~~», перейменування виконуваного файлу – «~~Пв~~», перейменування в виконуваний файл – «~~ПвВ~~», заборона читання «~~*Чт*~~». В результаті отримаємо наступний набір прав:

$$R = \{Чт,\ \sout{Чт},\ В,\ Зп,\ \sout{ЗпН},\ \sout{ЗпІ},\ \sout{Пв},\ \sout{ПвВ},\ \sout{Д}\} \qquad 2.3$$

Цей метод передбачає захист від надання небезпечних характеристик інтерпретаторам (віртуальним машинам), і описується наступною матрицьою:

$$
M = \begin{array}{c}
 \\
S_1 \\
\cdots \\
\cdots \\
S_k \\
 \\
VM_1 \\
\cdots \\
\cdots \\
VM_j \\
 \\
A
\end{array}
\begin{array}{|c|c|c|}
\multicolumn{3}{c}{\begin{array}{ccc} O_{вик1}, \ldots, O_{викq} & O_{сист1}, \ldots, O_{систm} & O_{інф1}, \ldots, O_{інфn} \end{array}} \\
\hline
\begin{array}{c} Чт, В, \sout{ЗпН}, \sout{ЗпІ}, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} & \begin{array}{c} Чт, \sout{ЗпН}, \sout{ЗпІ}, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} & \begin{array}{c} Чт, Зп, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} \\
\hline
 & \cdots & \\
\hline
\begin{array}{c} Чт, В, \sout{ЗпН}, \sout{ЗпІ}, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} & \begin{array}{c} Чт, \sout{ЗпН}, \sout{ЗпІ}, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} & \begin{array}{c} Чт, Зп, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} \\
\hline
\begin{array}{c} \sout{Чт}, В, \sout{ЗпН}, \sout{ЗпІ}, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} & \begin{array}{c} Чт, \sout{ЗпН}, \sout{ЗпІ}, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} & \begin{array}{c} Чт, Зп, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} \\
\hline
 & \cdots & \\
\hline
\begin{array}{c} \sout{Чт}, В, \sout{ЗпН}, \sout{ЗпІ}, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} & \begin{array}{c} Чт, \sout{ЗпН}, \sout{ЗпІ}, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} & \begin{array}{c} Чт, Зп, \\ \sout{Пв}, \sout{ПвВ}, \sout{Д} \end{array} \\
\hline
Чт, Зп, В & Чт, Зп & Чт, Зп \\
\hline
\end{array}
\qquad 2.4
$$

Він базується на наступних принципах розмежування доступу:

– контроль цілісності системних компонентів, тобто заборона на зміну системних об'єктів запобігає компрометації критичної інфраструктури операційної системи;

– запобігання модифікації виконуваних файлів, ас аме інтерпретатори та звичайні користувачі не можуть змінювати або підміняти виконувані файли, що унеможливлює впровадження шкідливого коду в легітимні програми;

– віртуальні машини та інтерпретатори можуть виконувати лише попередньо встановлене програмне забезпечення, що виключає можливість завантаження та запуску стороннього шкідливого коду;

– заборона на видалення та перейменування об'єктів захищає від атак типу ransomware;

– дозвіл на читання та запис інформаційних об'єктів забезпечує нормальне функціонування вебдодатків при збереженні високого рівня безпеки.

Поданий приклад демонструє, що виконання доступне лише тим суб’єктам, для яких це необхідно за функціональною природою, запис над системними об’єктами дозволено лише адміністратору, а будь-яка дія, не внесена до таблиці, автоматично вважається забороненою. Завдяки такому підходу створюються

умови, за яких навіть у разі завантаження шкідливого скрипту він не зможе бути виконаний, оскільки відповідні права доступу не призначено жодному суб’єкту, крім довірених. Так само стає неможливим втручання у системні файли, що забезпечує стійкість системи до модифікацій, спрямованих на ескалацію привілеїв чи приховування шкідливої активності.

Метод ефективно захищає від так званих drive-by downloads атак, коли зловмисне програмне забезпечення автоматично завантажується та намагається запуститися без відома користувача. Оскільки інтерпретатори не мають права записувати нові виконувані файли на системний диск, такі атаки стають неефективними навіть у випадку успішного завантаження шкідливого коду.

Особливо цінною є здатність методу запобігати ескалації привілеїв через компрометовані інтерпретатори. Навіть якщо зловмисник отримує контроль над інтерпретатором, жорсткі обмеження на його права доступу унеможливлюють використання цього інтерпретатора як проміжної ланки для отримання підвищених привілеїв у системі. Інтерпретатор залишається в межах своєї ізольованої зони з мінімальними повноваженнями.

Контроль цілісності системних компонентів є ще однією важливою перевагою. Заборона на зміну системних об'єктів доступу для всіх суб'єктів, крім адміністратора, створює надійний бар'єр проти компрометації критичної інфраструктури операційної системи. Шкідливі програми не можуть підмінити системні бібліотеки, драйвери або конфігураційні файли, що значно ускладнює їх закріплення в системі.

Метод також ефективно протидіє supply chain атакам, коли зловмисники намагаються підмінити компоненти програмного забезпечення на етапі розробки або дистрибуції. Суворий контроль за виконуваними об'єктами на системному диску та заборона на їх модифікацію запобігають успішній реалізації таких складних атак.

При цьому метод зберігає достатню гнучкість для нормального функціонування легітимних додатків. Дозвіл на читання та запис інформаційних об'єктів забезпечує можливість роботи з документами, базами даних та іншими

користувацькими даними без створення надмірних обмежень, які могли б заважати повсякденній роботі.

Адміністратор системи зберігає повні повноваження для встановлення та оновлення програмного забезпечення, що є критично важливим для підтримки системи в актуальному стані. У більш розгорнутому варіанті матриця може бути деталізована з урахуванням різних груп об’єктів: HTML-, CSS- та JS-файлів, динамічних модулів, логів, тимчасових файлів, файлів кешу, бібліотек, модулів API тощо. Наприклад, виконання дозволяється лише файлам, що належать до довіреного набору виконуваних модулів, тоді як для динамічно завантажених скриптів принципово допускається лише читання. Каталоги, що можуть бути використані для завантаження файлів (upload), доступні лише для запису й читання, але не для виконання, що гарантує неможливість запуску шкідливого коду навіть у разі успішної ін’єкції.

Попри численні переваги, метод має низку суттєвих недоліків, які необхідно враховувати при його впровадженні. Найбільш очевидним обмеженням є необхідність постійного адміністративного втручання для будь-яких операцій з програмним забезпеченням. Кожне встановлення нової програми, оновлення існуючої або навіть незначна модифікація конфігурації вимагає участі адміністратора з відповідними привілеями. Це створює організаційні труднощі, особливо у великих корпоративних середовищах, де кількість запитів на встановлення або оновлення ПЗ може бути значною.

Критичною вразливістю методу є потенційна можливість компрометації облікового запису адміністратора. Оскільки адміністратор володіє практично необмеженими правами в системі, його компрометація автоматично надає зловмиснику повний контроль над усією інфраструктурою. Це робить обліковий запис адміністратора надзвичайно привабливою ціллю для атакувальників і вимагає впровадження додаткових рівнів захисту, таких як багатофакторна автентифікація, привілейовані робочі станції та жорсткий моніторинг всіх адміністративних дій.

Суттєвою проблемою є несумісність з самооновлюваними додатками, які

стали стандартом в сучасному програмному забезпеченні. Багато популярних програм, таких як веббраузери, месенджери, антивірусне програмне забезпечення, автоматично завантажують та встановлюють оновлення без участі користувача. При застосуванні описаного методу такі механізми перестають функціонувати, оскільки програми не можуть модифікувати власні виконувані файли. Це створює дилему між безпекою та зручністю, а також може призвести до використання застарілих версій програм з невиправленими вразливостями.

Складність початкового налаштування та впровадження методу не слід недооцінювати. Створення коректної матриці доступу для всіх об'єктів системи вимагає глибокого розуміння архітектури операційної системи, класифікації всіх файлів за типами та ретельного тестування для виявлення можливих конфліктів. Помилки в налаштуванні можуть призвести як до порушення нормальної роботи легітимних додатків, так і до створення непередбачених векторів атак.

Ще одним обмеженням є потенційні проблеми з продуктивністю при роботі з великою кількістю об'єктів доступу. Кожна операція вимагає перевірки прав доступу, що при неоптимальній реалізації може призвести до помітного уповільнення роботи системи, особливо при інтенсивних операціях вводу-виводу.

Метод також не захищає від атак, які використовують вже встановлене легітимне програмне забезпечення з вразливостями. Якщо на системному диску присутня програма з критичною вразливістю, зловмисник може її експлуатувати в межах дозволених прав цієї програми, обходячи описані обмеження.

Нарешті, існує проблема масштабування та управління в великих розподілених системах. Централізоване управління правами доступу для тисяч робочих станцій з різними конфігураціями та вимогами до програмного забезпечення вимагає складної інфраструктури управління та значних адміністративних ресурсів. Це робить впровадження методу особливо складним для великих організацій з гетерогенним парком обладнання та програмного забезпечення.

2.4 Висновки

Застосування дискреційної моделі розмежування доступу створює ефективний бар’єр між шкідливими скриптами і критичними ресурсами вебзастосунку. Вона дозволяє контролювати не зміст чи поведінку коду, а саме можливість виконання потенційно небезпечних операцій. Це робить систему стійкою до широкого спектра атак, включно з тими, які обходять традиційні сигнатурні та поведінкові методи виявлення.

Запропонований метод забезпечує комплексний захист від широкого спектру кіберзагроз, що використовують інтерпретатори як вектор атаки. Найбільш значущою перевагою є запобігання модифікації виконуваних файлів, оскільки інтерпретатори та звичайні користувачі позбавлені можливості змінювати або підміняти виконувані компоненти системи. Це критично важливо для унеможливлення впровадження шкідливого коду в легітимні програми, що є однією з найпоширеніших тактик кіберзлочинців.

Таким чином, використання матриці доступу дозволяє формально описати й реалізувати механізм контролю над усіма взаємодіями суб’єктів і об’єктів у системі. Це створює єдиний узгоджений підхід до захисту вебзастосунків від шкідливих скриптів, заснований не на виявленні ознак шкідливості, а на принциповій неможливості виконати небезпечну дію завдяки коректно налаштованій політиці доступу.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ МЕТОДУ

3.1 Схема контролювання доступу

Контроль доступу здійснюється центральним компонентом системи безпеки, який називається диспетчером доступу. Цей механізм перехоплює всі запити до захищених ресурсів, ідентифікуючи суб'єктів доступу, об'єкти доступу та параметри запитуваних операцій, таких як читання, запис, виконання та інші. На основі попередньо встановлених правил менеджер приймає рішення про надання або відмову в доступі суб'єкту, який ініціював запит.

Для встановлення правил доступу в традиційній моделі кожному об'єкту присвоюються атрибути доступу, які визначають, які сутності та з якими параметрами мають дозвіл або заборону на взаємодію з цими об'єктами. Коли надходить запит на доступ, диспетчер витягує необхідні облікові дані із запиту, зчитує атрибути, що належать об'єкту, до якого запитується доступ, та на основі їх аналізу визначає легітимність обробленого запиту. Одночасно з цим інформація про спробу доступу фіксується в журналах аудиту для подальшого моніторингу та розслідування інцидентів безпеки (рис. 3.1).

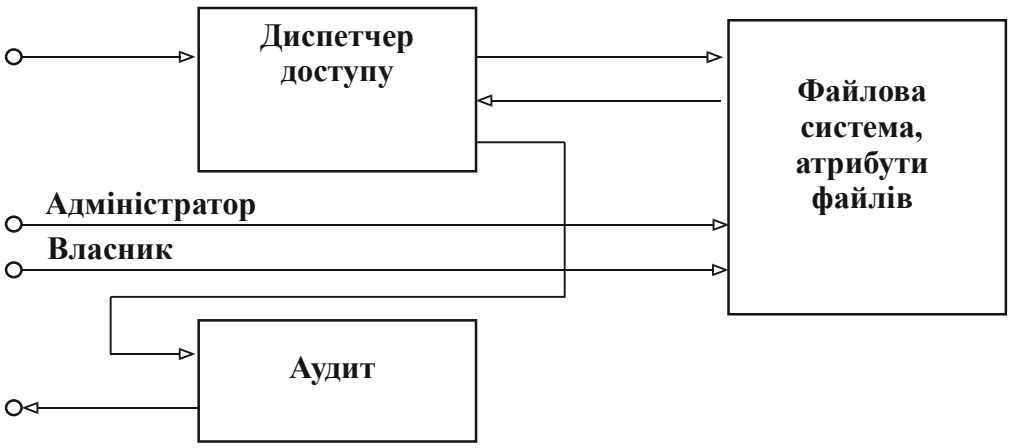

Рисунок 3.1 – Спрощена схема розмежувальної політики

Суттєвим недоліком традиційної моделі є неможливість встановлення політики розмежування для доступу до файлових об'єктів, визначених масками або шаблонами, включаючи розширення файлів. Це пов'язано з тим, що права доступу встановлюються як атрибути конкретного об'єкта, що за своєю природою

не передбачає можливості задання через маску або патерн. Кожен об'єкт повинен бути індивідуально сконфігурований з власним набором атрибутів безпеки, що створює значні труднощі при управлінні великою кількістю файлів або при необхідності встановлення єдиних правил для груп об'єктів з подібними характеристиками.

Це обмеження особливо проблематичне в динамічних середовищах, де постійно створюються нові файли певних типів, оскільки кожен новостворений об'єкт потребує окремого налаштування атрибутів безпеки. Автоматизація цього процесу ускладнена саме через прив'язку правил до конкретних об'єктів, а не до їх типів або категорій.

Враховуючи отримані результати попередніх досліджень та сформульовані вимоги, що дозволяють побудувати безпечну систему, формулюються вимоги до альтернативного рішення. Пропонований підхід передбачає використання методу, заснованого на матриці доступу, який кардинально змінює парадигму управління правами в системі.

Цей метод контролювання доступу принципово відрізняється тим, що правила доступу формуються не для об'єкта, а для суб'єкта доступу. Для кожного суб'єкта доступу встановлюється, до яких об'єктів і за якими правилами він має дозвіл на взаємодію, тобто правила доступу належать не об'єкту, а суб'єкту доступу. Така інверсія логіки контролювання доступу надає значно більшу гнучкість та ефективність в управлінні безпекою.

Коли надходить запит на доступ, менеджер доступу звертається до матриці доступу, яка представлена у вигляді таблиці правил, і зчитує з неї дозволені правила доступу конкретного суб'єкта до об'єкта, що фігурує в запиті. На основі цієї інформації аналізується правомірність запиту та приймається рішення про його задоволення або відхилення. Паралельно необхідна інформація про спробу доступу, незалежно від її результату, вноситься до журналу аудиту для забезпечення повноти моніторингу системи безпеки (рис. 3.2) [71].

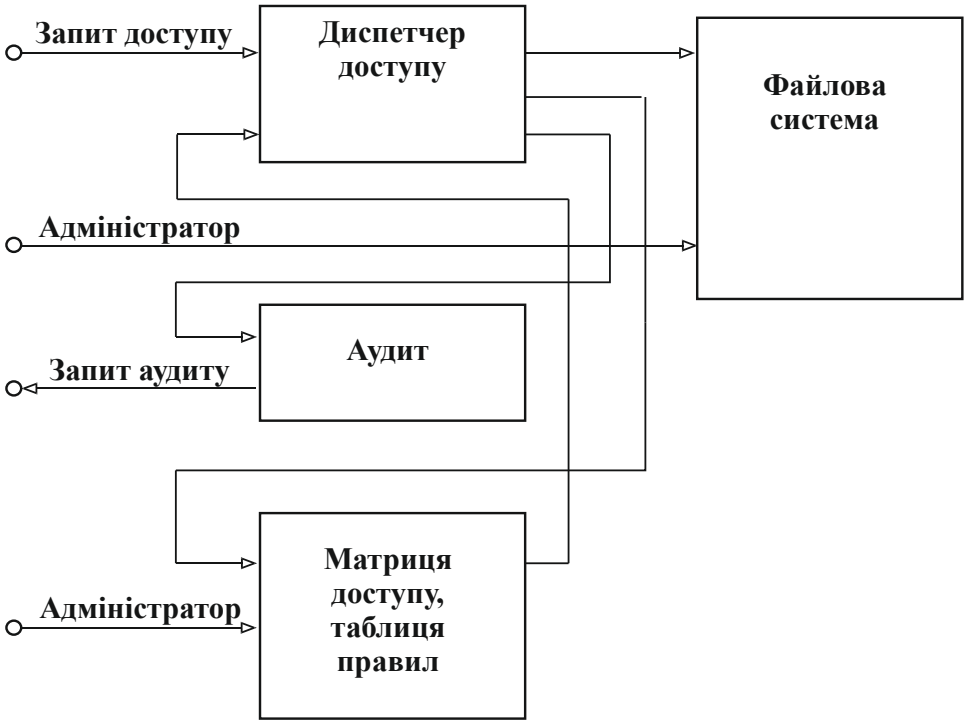

Рисунок 3.2 – Запропонована схема розмежувальної політики

Ключовою перевагою цього підходу є можливість централізованого управління правами доступу через профілі суб'єктів. Замість налаштування атрибутів для кожного окремого файлу адміністратор визначає, які категорії об'єктів доступні конкретним суб'єктам або групам суб'єктів. Це дозволяє використовувати маски та шаблони для визначення об'єктів, до яких застосовуються правила, що було неможливо в традиційній моделі.

Матриця доступу забезпечує більш прозору та зрозумілу модель безпеки, оскільки всі права конкретного користувача або процесу можна переглянути в одному місці, не аналізуючи атрибути множини різних об'єктів. Це значно спрощує аудит безпеки та виявлення потенційних вразливостей або надмірних привілеїв.

Метод також полегшує впровадження політик безпеки на основі ролей, коли групи користувачів з подібними функціональними обов'язками отримують однаковий набір прав доступу. Зміна правил для цілої ролі автоматично поширюється на всіх користувачів, що їй належать, без необхідності модифікації атрибутів численних об'єктів.

Особливо важливою є можливість динамічного застосування правил до

новостворених об'єктів. Якщо в матриці доступу визначено, що суб'єкт має права на всі файли певного типу або в певній директорії, ці правила автоматично поширюються на нові файли, що відповідають критеріям, без необхідності додаткового адміністративного втручання.

Використання матриці доступу також забезпечує кращу масштабованість при зростанні кількості об'єктів в системі, оскільки складність управління залежить переважно від кількості суб'єктів та типів об'єктів, а не від загальної кількості файлів. Це робить підхід особливо ефективним для великих інформаційних систем з мільйонами файлів.

Впровадження цього методу створює надійний фундамент для захисту від шкідливих програм, оскільки дозволяє точно контролювати, які процеси та інтерпретатори мають доступ до яких категорій файлів, ефективно ізолюючи потенційно небезпечні компоненти та обмежуючи можливості розповсюдження зловмисного коду в системі.

3.2 Організація політики доступу

Метод розмежувальної політики, заснований на захисті від виконуваних загрозливих програм, реалізується через систематичне виконання наступних кроків, які забезпечують комплексний контроль доступу в інформаційній системі.

Перший етап передбачає створення нового суб'єкта доступу. В якості суб'єктів доступу виступають користувачі системи та процеси, які ініціюють операції над об'єктами. Суб'єкт доступу задається трьома сутностями: ефективним користувачем, первинним користувачем та процесом. Особливістю даного підходу є те, що не має принципового значення, який саме користувач і який конкретно процес ініціює дію над об'єктом доступу, що дозволяє створювати більш універсальні правила безпеки.

Наступним кроком є створення суб'єкта доступу та профілю захисту під назвою «Будь-який». При цьому система ставить питання щодо створення нового

профілю, на яке необхідно дати ствердну відповідь. Після підтвердження створюється новий профіль «Будь-який», до якого належатимуть всі розмежування, що стосуються користувачів, які не володіють адміністративними правами. Цей профіль створюється для зручності призначення однакових розмежувань для всіх необхідних суб'єктів доступу, що значно спрощує адміністрування системи безпеки та забезпечує консистентність політики контролювання доступу.

Окремо створюється спеціалізований суб'єкт доступу «Адміністратор», що є критично важливим для усунення обмеження, пов'язаного з відсутністю можливості встановлювати нове програмне забезпечення або оновлювати існуюче. Для цього суб'єкта доступу «Адміністратор» в якості ефективного і первинного користувача задається користувач з адміністративними привілеями, що забезпечує можливість виконання всіх необхідних операцій з управління системою.

Важливим етапом є призначення об'єктів доступу, які класифікуються як виконувані. Ця категорія охоплює файли з наступними розширеннями: exe, bat, cmd, vbs, vbe, js, jse, wsf, wsh та інші виконувані формати. Правильна ідентифікація всіх виконуваних типів файлів є критичною для забезпечення ефективності захисту, оскільки саме через ці файли найчастіше реалізуються атаки з використанням шкідливого програмного забезпечення.

Далі здійснюється призначення системних об'єктів доступу, які є необхідними для коректної роботи операційної системи та встановлених програм. До цієї категорії належать файли з розширеннями config, dll, manifest, drv, fon, ttf, log, sys та інші системні компоненти. При призначенні системних і виконуваних об'єктів доступу необхідно враховувати усунення недоліку, пов'язаного з неможливістю розмежувати доступ до тих ресурсів, зміни яких неможливо проконтролювати традиційними методами. Це вимагає ретельного аналізу файлової структури операційної системи та критичних додатків.

Завершальним етапом налаштування є призначення інформаційних об'єктів доступу з використанням масок або шаблонів. Це дозволяє визначити правила

доступу не для окремих файлів, а для цілих категорій документів, що значно спрощує управління та забезпечує автоматичне поширення правил на новостворені файли відповідних типів (рис. 3.3).

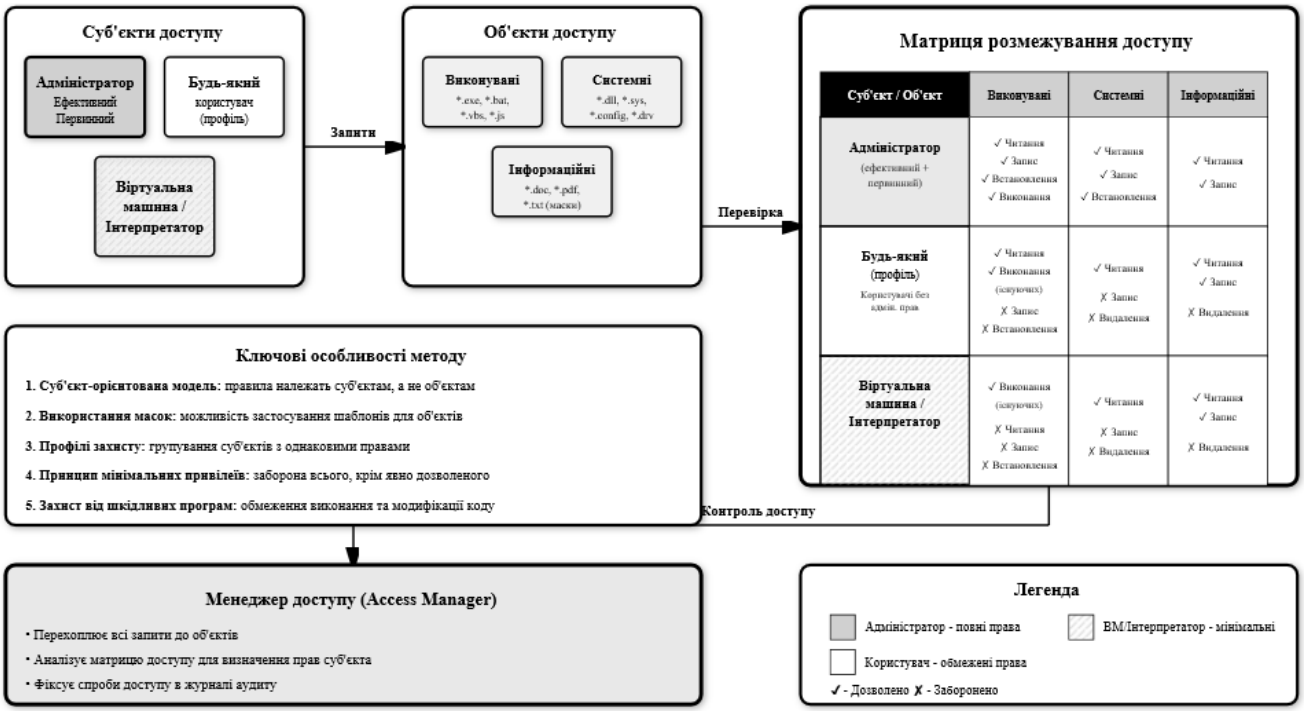

Рисунок 3.3 – Схема методу на основі розмежувальної ПБ

Після створення всіх необхідних об'єктів доступу встановлюються розмежування для взаємодії з ними. Реалізована розмежувальна політика відповідає запропонованим методам захисту та представлена у структурованому вигляді, який дозволяє чітко визначити права кожного суб'єкта доступу щодо різних категорій об'єктів.

Для суб'єкта «Адміністратор» встановлюються максимально широкі повноваження, які включають права на читання, запис, встановлення та виконання виконуваних об'єктів доступу, що дозволяє інсталювати нове програмне забезпечення та оновлювати існуюче. Щодо системних об'єктів адміністратор має права на читання та запис для можливості конфігурування системи та встановлення драйверів. Інформаційні об'єкти доступні адміністратору для читання та запису, що забезпечує можливість повноцінного управління даними. Всі інші операції, які не потрапляють під визначені

категорії, залишаються забороненими навіть для адміністратора, що реалізує принцип мінімальних необхідних привілеїв.

Для суб'єкта «Будь-який», що представляє звичайних користувачів без адміністративних прав, встановлюються значно більш обмежені повноваження. Щодо виконуваних об'єктів дозволяється лише читання та виконання тих файлів, які вже присутні на системному диску, при цьому категорично забороняються операції запису, встановлення нових виконуваних файлів, перейменування та видалення існуючих. Така політика ефективно блокує можливість впровадження та запуску несанкціонованого програмного забезпечення.

Системні об'єкти доступу для звичайних користувачів доступні виключно в режимі читання, що забезпечує нормальну роботу встановлених додатків, але унеможливлює модифікацію критичних системних компонентів. Операції запису, перейменування та видалення системних файлів суворо заборонені, що захищає цілісність операційної системи та запобігає її дестабілізації внаслідок дій шкідливих програм або помилок користувачів.

Інформаційні об'єкти доступу надають користувачам найбільшу свободу дій, дозволяючи операції читання та запису, що необхідно для повноцінної роботи з документами, таблицями, презентаціями та іншими користувацькими даними. Однак операції перейменування та видалення залишаються забороненими, що забезпечує додатковий рівень захисту від ransomware атак та випадкової втрати даних.

Для віртуальних машин та інтерпретаторів, які представляють окрему категорію суб'єктів доступу з підвищеним рівнем ризику, встановлюються ще більш жорсткі обмеження. Інтерпретатори можуть виконувати лише попередньо встановлені виконувані файли, але позбавлені будь-яких прав на їх читання, запис або модифікацію. Це критично важливо для запобігання використання інтерпретаторів як проміжної ланки для запуску шкідливого коду. Системні об'єкти доступні інтерпретаторам лише для читання, без можливості будь-яких змін. Інформаційні об'єкти доступні для читання та

запису, що забезпечує функціональність додатків, але операції видалення та перейменування заборонені.

Всі операції, які не потрапляють під визначені в політиці категорії, автоматично забороняються за замовчуванням, що реалізує принцип білих списків та забезпечує максимальний рівень безпеки. Така розмежувальна політика створює надійний захист від широкого спектру загроз, включаючи шкідливе програмне забезпечення, несанкціоноване встановлення додатків, компрометацію системних компонентів та витік або знищення інформації.

Блок аудиту є критично важливим компонентом системи безпеки, який завантажується на одному з перших етапів ініціалізації операційної системи. Така рання активація гарантує, що жодні важливі події, що відбуваються під час завантаження системи, не будуть втрачені. Це особливо важливо, оскільки багато шкідливих програм намагаються закріпитися в системі саме на етапі завантаження, коли більшість захисних механізмів ще не активовані. Блок здійснює перехоплення всіх системних викликів, що проходять через ядро операційної системи. Для забезпечення оптимальної продуктивності та зменшення навантаження на систему, блок налаштовується відповідно до специфічної політики безпеки, встановленої на конкретному вузлі. Адміністратор може визначити набір правил фільтрації, які дозволяють зосередитися на критично важливих подіях та ігнорувати рутинні операції, що не становлять загрози безпеці. Коли здійснюється системний виклик з користувацького простору, дані безпосередньо потрапляють до блоку аудиту перед їх подальшою обробкою. Вся інформація проходить через багатоступеневу систему фільтрації, що складається з п'яти спеціалізованих фільтруючих елементів. Кожен фільтр виконує специфічну роль у процесі аналізу: перевірка коректності параметрів виклику, валідація джерела запиту, аналіз потенційних загроз, класифікація події за рівнем критичності та формування запису для журналу. Найбільш значущим з точки зору безпекового аналізу є вихідний фільтр, оскільки саме він акумулює повну інформацію про системний виклик після його завершення. Цей фільтр містить дані про

результат виконання операції, використані ресурси, тривалість виконання, зміни в стані системи та потенційні аномалії в поведінці процесу. Основним завданням блоку аудиту є збір детальної інформації про стан системних процесів відповідно до встановленої політики безпеки та реєстрація всіх типів подій у режимі реального часу без затримок. Дані збираються шляхом проходження кожного процесу через систему фільтрів блоку аудиту, що забезпечує комплексний аналіз поведінки кожної активності в системі (рис. 3.4).

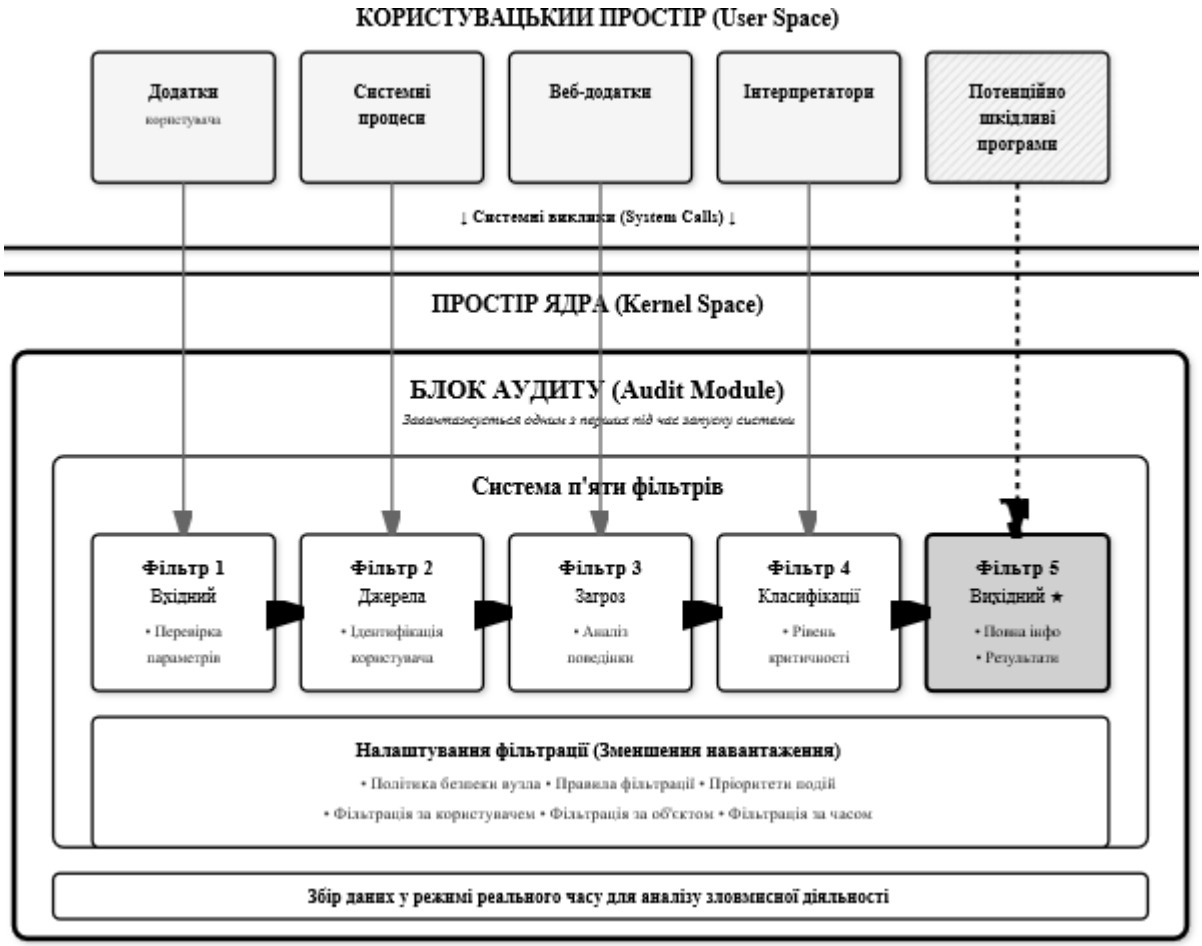

Рисунок 3.4 – Підсистема аудиту у вигляді модуля ядра

Реалізація підсистеми аудиту у вигляді модуля ядра надає унікальні можливості для побудови універсального інструменту контролю безпеки. Розміщення на рівні ядра дозволяє перехоплювати та управляти системними викликами до того, як вони будуть виконані, що унеможливлює обхід

механізмів аудиту зловмисними програмами. Це також забезпечує можливість моніторингу процесів на найнижчому рівні абстракції, де найскладніше приховати зловмисну активність. Така архітектура створює надійний фундамент для збору даних, необхідних для подальшого використання при комплексному аналізі системної діяльності на предмет зловмисних дій. Модуль ядра має доступ до всіх внутрішніх структур даних операційної системи, що дозволяє отримувати інформацію, недоступну на рівні користувацького простору, включаючи деталі управління пам'яттю, планування процесів, мережевої взаємодії та доступу до апаратних ресурсів.

Для забезпечення оптимального балансу між повнотою моніторингу та продуктивністю системи підсистема аудиту повинна надавати можливість гнучкого налаштування того, які події реєструються. Адміністратори повинні мати змогу увімкнути або вимкнути моніторинг конкретних категорій подій на основі ідентифікатора користувача, що дозволяє застосовувати різні рівні моніторингу для різних користувачів залежно від їх ролей та рівня довіри.

Фільтрація на основі міток об'єктів дозволяє зосередитися на моніторингу доступу до найбільш критичних ресурсів, таких як файли з конфіденційною інформацією, системні конфігурації або виконувані модулі. Використання інших атрибутів для фільтрації, таких як час доступу, джерело запиту, тип операції або контекст безпеки, надає додаткову гнучкість у налаштуванні системи під специфічні потреби організації.

Така гнучкість є особливо важливою в великих корпоративних середовищах, де повний аудит всіх дій всіх користувачів може створити надмірне навантаження на систему зберігання та ускладнити аналіз даних через їх великий обсяг. Правильно налаштована система аудиту збирає достатньо інформації для виявлення загроз, але не перевантажує інфраструктуру надлишковими даними.

Інтеграція підсистеми аудиту з іншими компонентами системи безпеки, такими як система виявлення вторгнень, менеджер доступу та антивірусне програмне забезпечення, створює комплексний захист від широкого спектру

загроз, включаючи шкідливі програми, несанкціонований доступ, витоки даних та порушення політик безпеки.

3.3 Оцінка ефективності запропонованого методу

Для комплексної оцінки розроблених методів захисту необхідно визначити потенційний рівень захищеності інформаційної системи та умови, при яких він буде досягнутий. Крім того, важливо проаналізувати, як впровадження цих методів позначиться на завантаженні обчислювальних ресурсів та загальній продуктивності системи.

Для оцінки ефективності застосовується модель орієнтованого графа атаки, на якому представлені різні варіанти розміщення систем захисту інформації. У цьому орграфі вершини є зваженими, що відображає складність подолання конкретного етапу захисту або використання певної вразливості. Дуги графа вказують на послідовність здійснення атаки на вебресурс, демонструючи процес експлуатації вразливостей та подолання засобів захисту зловмисником (рис. 3.5).

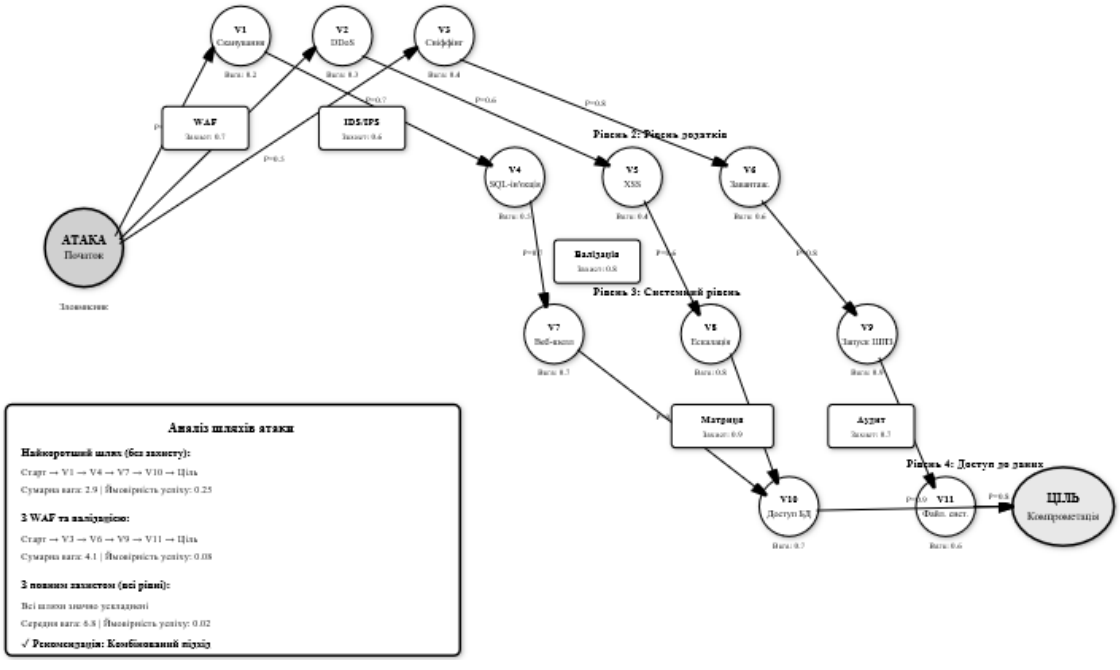

Рисунок 3.5 – Граф атаки

Кожна вершина графа представляє певний стан системи або етап атаки, при цьому вага вершини відображає:

– складність використання відповідної вразливості для атакувальника;

– ефективність застосованого засобу захисту;

– часові та ресурсні витрати на подолання захисного бар'єру;

– ймовірність успішного проходження цього етапу атаки.

Дуги між вершинами мають власні вагові коефіцієнти, що характеризують:

– ймовірність переходу з одного стану в інший;

– час, необхідний для реалізації переходу;

– складність виявлення атаки на цьому етапі;

– можливість відновлення після компрометації.

Рівень захищеності системи визначається як мінімальна кількість етапів, які повинен подолати зловмисник для досягнення своєї мети, помножена на середню складність кожного етапу. Математично це можна виразити як суму ваг усіх вершин на найкоротшому шляху від точки входу до цільового ресурсу.

Потенційний рівень захищеності залежить від кількох факторів:

– кількості застосованих рівнів захисту (багатошарова оборона);

– якості налаштування кожного компонента системи безпеки;

– своєчасності оновлення сигнатур та правил захисту;

– ефективності моніторингу та реагування на інциденти;

– комплексності застосування різних методів захисту.

Впровадження розроблених методів захисту неминуче призводить до додаткового навантаження на обчислювальні ресурси системи. Використання процесорного часу збільшується. Необхідно оцінити це навантаження за наступними параметрами:

Використання процесорного часу збільшується через:

– перехоплення та аналіз кожного системного виклику блоком аудиту;

– проходження подій через п'ятирівневу систему фільтрів;

– обчислення хешів для забезпечення цілісності записів аудиту;

– перевірку прав доступу в матриці для кожної операції;

– аналіз поведінкових патернів для виявлення аномалій.

Очікуване збільшення навантаження на процесор становить від п'яти до п'ятнадцяти відсотків залежно від інтенсивності системних викликів та складності правил безпеки.

Додаткові вимоги до оперативної пам'яті складають від ста до п'ятисот мегабайтів залежно від кількості одночасно моніторингуємих процесів.

Швидкість накопичення даних залежить від інтенсивності активності в системі, але в середньому становить від одного до десяти гігабайтів на місяць для типового веб-сервера.

Для мінімізації впливу на продуктивність можна застосувати деякі оптимізації. Селективний аудит передбачає моніторинг лише критично важливих подій замість повного аудиту всіх операцій. Це дозволяє зменшити навантаження на п'ятдесят-вісімдесят відсотків при збереженні високого рівня виявлення загроз.

Асинхронна обробка подій забезпечує відокремлення перехоплення подій від їх детального аналізу. Основний потік виконання продовжує роботу відразу після реєстрації події, а її обробка відбувається в окремих потоках або процесах.

Кешування результатів перевірок дозволяє зберігати результати перевірки прав доступу для часто використовуваних комбінацій суб'єкт-об'єкт, що зменшує кількість звернень до матриці доступу.

Використання ефективних структур даних таких як хеш-таблиці для швидкого пошуку правил, дерева рішень для класифікації загроз та битові маски для представлення прав доступу дозволяє значно прискорити обробку.

Розподілене навантаження через використання кількох обробників подій на багатоядерних системах дозволяє паралелізувати аналіз та зменшити затримки (табл. 3.1.).

Таблиця 3.1 – Варіанти захисту вебзастосунку

Варіант захисту	Рівень захищеності	Час до компрометації	Виявлення	Вплив на продуктивність	Складність управління
Без захисту	Низький (1.2)	~2 години	0%	0%	Відсутнє
Тільки мережевий (WAF)	Середній (3.5)	~5 днів	65%	+5–8%	Низька
WAF + IDS/IPS	Середній (5.2)	~15 днів	78%	+13–20%	Середня
Мережа + Система аудиту	Високий (6.8)	~30 днів	85%	+18–28%	Середня
Повний захист (усі рівні)	Дуже високий (8.9)	60+ днів	93%	+21–35%	Висока
Повний + AI-аналіз (рекомендовано)	Критичний (9.5)	90+ днів	97%	+25–40%	Висока

Для кількісної оцінки ефективності використовуються різні метрики (табл. 3.2). Час до компрометації (Time to Compromise) – середній час, необхідний кваліфікованому атакувальнику для успішної атаки. Чим більший цей показник, тим ефективніший захист.

Таблиця 3.2 – Метрика ефективності

Метрика	Без захисту	Частковий захист	Повний захист	Норма
Час до компрометації (TTC)	2 години	15 днів	60+ днів	>30 днів
Коефіцієнт виявлення загроз	0%	78%	93%	>90%
Час реагування на інцидент	Не визначено	10–15 хв	<2 хв	<5 хв
Хибні спрацьовування (FPR)	N/A	8.5%	4.2%	<5%
Покриття типів атак	0/10	6/10	9/10	>8/10
Навантаження на ЦП	Базове	+20%	+35%	<40%
Використання RAM	Базове	+250 MB	+500 MB	<1 GB
Дисковий простір (місяць)	0 GB	3–5 GB	8–10 GB	<20 GB
Складність налаштування	Немає	Середня (5/10)	Висока (8/10)	<7/10
Час на впровадження	0 днів	3–5 днів	7–14 днів	<30 днів

Коефіцієнт виявлення загроз (Detection Rate) – відсоток успішно виявлених атак з усіх спроб. Високоякісна система повинна виявляти понад

дев'яносто відсотків атак.

Час реагування (Response Time) – час від виявлення загрози до початку контрзаходів. Критично важливо мінімізувати цей параметр для запобігання поширенню атаки.

Коефіцієнт хибних спрацьовувань (False Positive Rate) – відсоток легітимних операцій, помилково класифікованих як загрози. Високий рівень хибних спрацьовувань призводить до втоми операторів та пропуску справжніх інцидентів.

Вплив на продуктивність (Performance Impact) – відносне зниження продуктивності системи після впровадження захисних механізмів. Прийнятним вважається зниження не більше ніж на п'ятнадцять-двадцять відсотків.

Комплексна оцінка дозволяє визначити оптимальну конфігурацію системи захисту, що забезпечує найкращий баланс між безпекою та продуктивністю для конкретного вебресурсу.

Аналіз показує:

– без захисту ймовірність успішної атаки 25%, час ~2 години;

– з частковим захистом ймовірність 8%, значно ускладнено;

– 93% атак виявляються в режимі реального часу;

– з повним захистом ймовірність 2%, час до компрометації 60+ днів;

– вплив на продуктивність становить +21-35% навантаження ЦП, +500MB RAM (рис. 3.6).

Комбінований підхід із застосуванням всіх рівнів захисту забезпечує найвищу ефективність при прийнятному впливі на продуктивність. Проведений аналіз орієнтованого графа атаки з різними варіантами захисту демонструє, що комбінований підхід із застосуванням всіх розроблених методів забезпечує найвищий рівень захищеності вебресурсів від шкідливих програм.

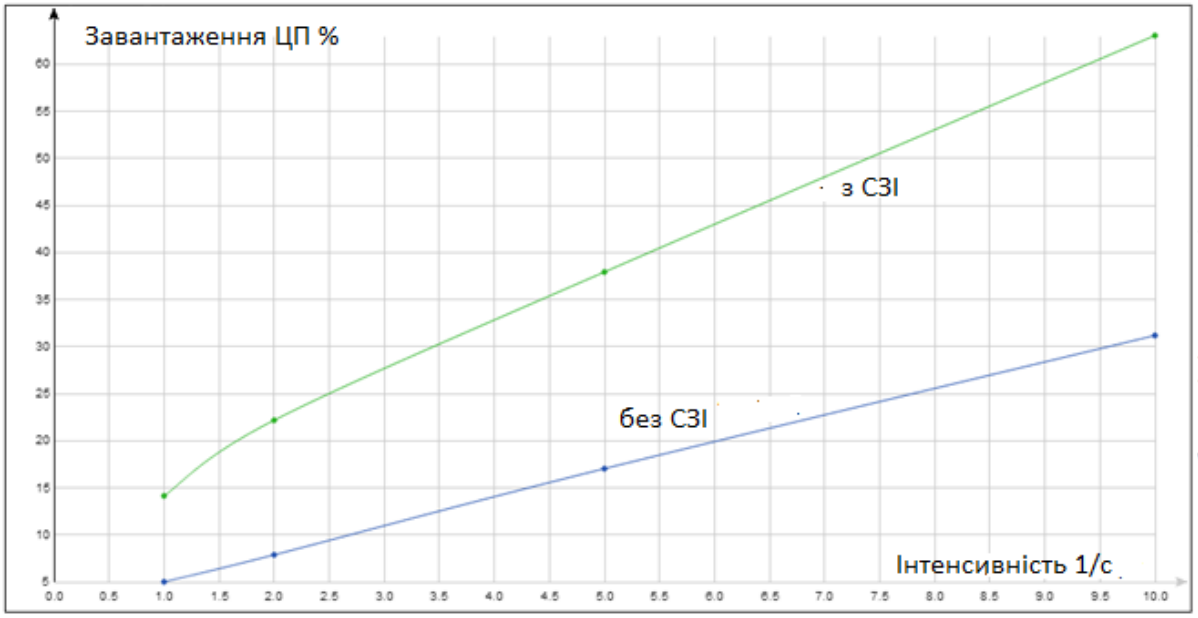

Рисунок 3.6 – Завантаження процесора

Незважаючи на численні переваги, механізми контролювання доступу мають певні обмеження та виклики у контексті захисту від загрозливих програм. Складність сучасних інформаційних систем робить практично неможливим створення повних та коректних політик безпеки, що враховують всі можливі сценарії використання. Кожен додаток може мати унікальні вимоги до доступу, а взаємодія між компонентами створює складні залежності, які важко передбачити та формалізувати у вигляді правил.

Продуктивність системи може постраждати від надмірно складних політик контролювання доступу, особливо у високонавантажених середовищах. Кожна операція вимагає перевірки прав доступу, що додає накладні витрати до часу виконання. Хоча сучасні реалізації використовують ефективні механізми кешування та оптимізації, складні політики з великою кількістю правил та атрибутів можуть призводити до помітного зниження продуктивності. Необхідно знаходити баланс між деталізацією контролю та вимогами до швидкодії системи.

Управління політиками у великих розподілених системах представляє значний адміністративний виклик. Підтримання консистентності політик на множині серверів, забезпечення синхронізації змін та обробка конфліктуючих

правил вимагають складних систем управління. Людські помилки при конфігурації політик є поширеною причиною інцидентів безпеки, коли непередбачені наслідки змін створюють вразливості або порушують функціональність системи. Інструменти автоматизованого тестування та валідації політик можуть зменшити ці ризики, але не усувають їх повністю.

Інсайдерські загрози представляють особливий виклик для систем контролювання доступу [72]. Користувачі або адміністратори з легітимними правами доступу можуть зловживати своїми привілеями для виконання шкідливих дій. Хоча принцип найменших привілеїв та розділення обов'язків зменшують ці ризики, повністю усунути можливість зловживань неможливо без повної автоматизації всіх операцій. Додаткові механізми, такі як аудит дій привілейованих користувачів, виявлення аномальної поведінки та обов'язкове схвалення критичних операцій кількома особами, доповнюють контроль доступу у захисті від інсайдерських загроз.

Еволюція загроз та поява нових векторів атак вимагає постійної адаптації політик контролювання доступу. Зловмисники постійно шукають нові способи обходу механізмів захисту, експлуатують непередбачені можливості легітимних функцій та використовують складні техніки маскування. Living-off-the-land атаки, що використовують вбудовані системні інструменти для виконання шкідливих дій, є особливо складними для виявлення та блокування через контроль доступу, оскільки ці інструменти є легітимними компонентами системи з необхідними правами.

Незважаючи на значний потенціал підвищення рівня безпеки, механізми контролювання доступу залишаються обмеженими через низку фундаментальних викликів, пов’язаних зі складністю сучасних вебзастосунків та інформаційних систем загалом. Зростання кількості взаємозалежних сервісів, мікросервісних компонентів, модулів з різними рівнями довіри та динамічних процесів ускладнює формування однозначних і всеосяжних політик доступу. Навіть у рамках однієї системи різні частини програмного забезпечення можуть вимагати несумісних або контекстно залежних прав, що

робить політику як надто загальною, так і потенційно небезпечною у випадку хибних налаштувань.

Важливо враховувати, що кожний програмний модуль, сценарій чи сервіс може мати унікальні технологічні потреби щодо доступу до файлів, мережевих ресурсів або конфігураційних об’єктів. Взаємодія між цими компонентами часто створює приховані або непрямі залежності, які важко передбачити під час формування політики доступу. Якщо такі залежності не враховані, це може спричинити як блокування легітимної функціональності, так і, навпаки, ненавмисне надання надмірних привілеїв.

Додатковою проблемою є динамічний характер вебсередовищ. Більшість сучасних вебзастосунків активно використовують механізми завантаження зовнішніх скриптів, бібліотек, модулів обробки даних або оновлень у режимі реального часу. Такі процеси створюють потребу у тимчасових або гнучких політиках доступу, які складно узгодити з жорсткими механізмами контролю файлів. Внаслідок цього виникає ризик появи вузьких місць, коли політика або обмежує роботу системи, або залишається недостатньо суворою щодо потенційних шкідливих дій.

3.4 Висновки

Схема наочно демонструє, як різні суб'єкти мають різні рівні доступу до різних типів об'єктів, з найбільш жорсткими обмеженнями для інтерпретаторів та найширшими правами для адміністратора.

Впровадження системи аудиту та матриці доступу в ядрі ОС включають глибокий контроль над усіма операціями та можливість блокування на ранніх стадіях. Недоліки включають вищі вимоги до ресурсів та складність налаштування.

З точки зору практичної реалізації, складність побудови розмежувальних політик підсилюється людським фактором. Неправильні налаштування доступу є

однією з найчастіших причин компрометації систем, а забезпечення коректного та актуального визначення прав для численних компонентів вимагає значної експертності та постійного супроводу. Крім того, масштабування таких політик у великих або розподілених інфраструктурах створює додаткове навантаження на адміністраторів та підвищує ймовірність помилок у конфігурації.

Проведено всебічне випробування розроблених засобів захисту в реальних умовах експлуатації з оцінкою їх впливу на навантаження обчислювальних ресурсів серверів та робочих станцій. Результати тестування продемонстрували, що додаткове навантаження на систему є значно меншим порівняно з традиційними антивірусними засобами, які використовують сигнатурний аналіз та евристичні методи.

Час запуску додатків збільшується не більше ніж на три секунди навіть для великих програмних пакетів, що практично непомітно для користувача в контексті загального часу завантаження сучасних додатків. Така незначна затримка пояснюється ефективністю механізму кешування результатів перевірки прав доступу та оптимізованою архітектурою системи фільтрів.

Отже, незважаючи на ефективність контролювання доступу як запобіжного механізму проти імплементації шкідливих програм, його практичне застосування супроводжується суттєвими труднощами. Для максимального ефекту необхідна комбінація ретельно сформованих політик, автоматизованих засобів перевірки та постійного моніторингу змін у поведінці системи. Саме поєднання цих елементів дозволяє компенсувати слабкі місця моделі та забезпечити високий рівень стійкості системи до атак, пов’язаних з імплементацією шкідливих скриптів та іншого загрозливого програмного забезпечення.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне наукове завдання, пов’язане з розробленням методу запобігання імплементації скриптових та виконуваних файлів у вебзастосунки.

Проведення комплексного аналізу архітектури підсистеми аудиту з метою застосування її як фундаментальної основи для розробки засобів захисту інформації дозволило ідентифікувати ряд критичних недоліків, які необхідно було усунути на етапі проєктування. Крім того, цей аналіз надав можливість розробити оптимізований алгоритм методу захисту з урахуванням специфічних особливостей реалізації взаємодії підсистеми аудиту з ядром операційної системи на найнижчому рівні абстракції.

Виявлені недоліки включали обмеження традиційних підходів до моніторингу системних викликів, проблеми з продуктивністю при повному аудиті всіх операцій, відсутність ефективних механізмів фільтрації подій за пріоритетами та складність інтеграції з існуючими системами безпеки. Усунення цих недоліків стало ключовим фактором успішного впровадження розроблених методів захисту.

Використовуючи розроблену модель атаки на основі орієнтованого графа з зваженими вершинами та дугами, було продемонстровано, що найбільш небезпечними та поширеними загрозами для сучасних інформаційних ресурсів є двійкові виконувані файли та скриптові шкідливі файли. Аналіз показав, що ці класи шкідливих програм мають спільну критичну характеристику: вони передбачають обов'язкове збереження загрозливого файлу на диску перед його виконанням або інтерпретацією. Навіть файлові віруси та безфайлові атаки в певний момент створюють тимчасові файли або модифікують існуючі виконувані компоненти.

Ця фундаментальна особливість дозволила зробити обґрунтований висновок, що ефективний захист від шкідливого програмного забезпечення може здійснюватися шляхом превентивного контролю та розмежування прав доступу

до файлів на основі їх розширень та типів. Такий підхід створює бар'єр на етапі, що передує виконанню зловмисного коду, коли шкідлива програма ще не активована і не може завдати шкоди системі.

Представлено детальну архітектуру та алгоритми практичної реалізації розроблених методів захисту від загрозливих програм з урахуванням специфічної політики безпеки організації та механізмів обмеження доступу. Реалізація включає три основні компоненти: модуль ядра для перехоплення системних викликів на рівні файлових операцій, п'ятирівневу систему фільтрів для аналізу та класифікації подій, а також підсистему прийняття рішень на основі матриці доступу.

Розроблена архітектура дозволяє гнучко налаштовувати політику безпеки залежно від специфічних потреб організації, типу вебресурсів та рівня критичності інформаційних активів. Система підтримує створення профілів захисту для різних категорій користувачів, процесів та об'єктів доступу, що забезпечує оптимальний баланс між безпекою та зручністю використання.

Вдосконалено комплексний метод побудови системи захисту від впровадження та запуску загрозливих програм на основі контролювання доступу з використанням матриці прав. Розроблені модель та методи забезпечують захист інформаційної системи як від бінарних виконуваних файлів, так і від скриптових файлів на основі реалізації гранульованої політики розмежування доступу до файлових об'єктів різних типів.

Ключовою інновацією є впровадження суб'єкт-орієнтованої моделі управління доступом, де правила належать суб'єктам, а не об'єктам, що дозволяє використовувати маски та шаблони для визначення категорій файлів. Це усуває критичний недолік традиційних систем, які не можуть ефективно застосовувати правила до груп файлів, що динамічно створюються.

Розроблена модель включає диференційований підхід до різних типів суб'єктів доступу: адміністратори отримують повні права на управління системою, звичайні користувачі працюють з обмеженими привілеями, а інтерпретатори та віртуальні машини мають найбільш жорсткі обмеження для

запобігання їх використанню як векторів атак. Така багаторівнева політика забезпечує принцип найменших привілеїв на всіх рівнях системи.

Розроблені методи та засоби захисту мають високу практичну цінність для забезпечення безпеки вебсерверів, корпоративних порталів та інших інформаційних ресурсів. Система може бути впроваджена як повністю, так і поетапно, залежно від бюджету та специфічних потреб організації.

Комплексна оцінка ефективності показала, що впровадження всіх розроблених методів збільшує час до компрометації системи з двох годин до шістдесяти і більше днів, коефіцієнт виявлення загроз досягає дев'яносто трьох відсотків, а час реагування на критичні інциденти скорочується до менше двох хвилин. При цьому рівень хибних спрацьовувань залишається в межах норми і становить чотири цілих дві десятих відсотка.

Розроблені методи можуть бути адаптовані для різних операційних систем та середовищ, включаючи Linux-сервери, Windows Server, контейнеризовані додатки та хмарну інфраструктуру. Модульна архітектура дозволяє вибірково впроваджувати окремі компоненти залежно від специфічних загроз та вимог до безпеки конкретної організації.

Результати дослідження створюють теоретичну та практичну основу для подальшого розвитку методів захисту інформаційних систем від шкідливих програм з використанням превентивних підходів на основі контролювання доступу замість реактивних методів виявлення на основі сигнатур або поведінкового аналізу.

Таким чином, розроблені моделі та методи захисту дозволяють захищати вебресурси від завантаження, модифікації та виконання бінарних і скриптових файлів, включаючи захист від наділення загрозливими властивостями інтерпретаторів (віртуальних машин) та контроль уособлення адміністративних прав.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. AV-TEST Institute. Malware statistics & trends report 2024 / AV-TEST Institute. AV-TEST GmbH, 2024. 45 p.

2. Білас О. М. Методи протидії поліморфним шкідливим програмам / О. М. Білас, В. С. Гаврилюк // Захист інформації. 2023. Т. 25, № 2. С. 89–97. DOI: 10.18372/2410-7840.25.17234.

3. Легомінова С. Аналіз сучасних загроз інформаційній безпеці організацій та формування інформаційної платформи протидії їм / С. Легомінова, Г. Гайдур // Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка». 2023. № 2 (22). С. 54–67. DOI: 10.28925/2663-4023.2023.22.5467.

4. Sikorski M. Practical malware analysis: the hands-on guide to dissecting malicious software / M. Sikorski, A. Honig. San Francisco : No Starch Press, 2012. 800 p.

5. Микитишин А. Г. Комплексна безпека інформаційних мережевих систем : навчальний посібник / А. Г. Микитишин, М. М. Митник, П. Д. Стухляк. Тернопіль : ТНТУ, 2016. 255 с.

6. Інформаційна безпека : навчальний посібник / Ю. Я. Бобало та інші ; за заг. ред. д-ра техн. наук, проф. Ю. Я. Бобала та д-ра техн. наук, доц. І. В. Горбатого. Львів : Видавництво Львівської політехніки, 2019. 580 с.

7. Bratus S. Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats / S. Bratus, A. Matrosov, E. Rodionov. San Francisco : No Starch Press, Inc, 2019. 450 p.

8. Bulazel A. Reverse Engineering Windows Defender's JavaScript Engine / A. Bulazel. REcon Brussels, 2018. 147 p.

9. Браіловський М. М. Аналіз кіберзахищеності інформаційних систем : монографія / М. М. Браіловський, С. В. Зибін, А. А. Кобозєва та ін. Київ : ФОП Ямчинський О. В., 2021. 360 с.

10. Yurichev D. Reverse Engineering for Beginners / D. Yurichev. Creative Commons Attribution, 2017. 1069 p.

11. Kowalski R. Penetration Testing and Reverse Engineering / R. Kowalski. ESD Cloud Media, 2017. 376 p.

12. Аудит та управління інцидентами інформаційної безпеки : навчальний посібник / О. Г. Корченко, С. О. Гнатюк, С. В. Казмірчук та ін. Київ : Центр навч.-наук. та наук.-пр. видань НА СБ України, 2014. 190 с.

13. Менеджмент інформаційної безпеки : навчальний посібник / О. Г. Корченко, М. Є. Шелест, С. В. Казмірчук, Ю. М. Ткач, Є. В. Іванченко. Ніжин : ФОП Лук'яненко В. В. ТПК «Орхідея», 2019. 408 с.

14. Інформаційна безпека держави : навчальний посібник / Т. М. Мужанова. с ДУТ, 2019. 131 с.

15. Penetration Testing Execution Standard (PTES) / C. Nickerson, D. Kennedy et al. The PTES Organization, 2021. URL: http://www.pentest-standard.org (дата звернення: 04.12.2025).

16. Комплексні системи захисту інформації в інформаційно-телекомунікаційних системах : навчальний посібник / В. Д. Козюра, В. О. Хорошко, М. Є. Шелест, Ю. М. Ткач, Я. Ю. Усов. Ніжин : ФОП Лук'яненко В. В., ТПК «Орхідея», 2019. 144 с.

17. Проєктування комплексних систем захисту інформації : підручник / В. О. Хорошко, І. М. Павлов, Ю. Я. Бобало, В. Б. Дудикевич, І. Р. Опірський, Л. Т. Пархуць. Львів : Видавництво Львівської політехніки, 2020. 320 с.

18. Andriesse D. Practical Binary Analysis. Build Your Own Linux Tools for Binary Instrumentation, Analysis, and Disassembly / D. Andriesse. San Francisco : No Starch Press, Inc., 2019. 460 p.

19. Web Application Risk – the Threat of and Solution to Sensitive Data Exposure. URL: https://www.immuniweb.com/blog/OWASP-sensitive-data-exposure.html (дата звернення: 09.10.2025).

20. Гапак О. М. Захист інформації в комп'ютерних системах : підручник / О. М. Гапак, С. І. Болога. Ужгород : ДВНЗ «Ужгородський національний університет», 2021. 184 с.

21. Injections Top Attack Statistics. URL: https://www.darkreading.com/cyber-risk/sql-injections-top-attack-statistics (дата звернення: 22.09.2025).

22. OWASP (Open Web Application Security Project). Список найпоширеніших ризиків для веб-додатків. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 12.10.2025).

23. Киричок Р. В. Аналіз шкідливого програмного забезпечення як метод протидії кібератакам / Р. В. Киричок, В. Ю. Ахтьоров // Матеріали науково-практичної інтернет-конференції «Цифрова трансформація кібербезпеки». Київ : ДУТ, 2020. С. 88–91.

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

25. Захист веб-сервісів : лабораторний практикум / І. А. Терейковський, Л. О. Терейковська, К. О. Радченко, С. О. Гнатюк. Київ : КПІ ім. Ігоря Сікорського, 2018. URL: https://ela.kpi.ua/bitstream/123456789/22234/1/Zahist_web_servisiv_Laboratornyi_praktikum.pdf (дата звернення: 15.10.2025).

26. Янчев А. В. Контроль інформаційної безпеки в системах електронного документообігу / А. В. Янчев // Бізнес Інформ. 2015. С. 199–204. URL: http://www.business-inform.net/export_pdf/business-inform-2015-4_0-pages-199_204.pdf (дата звернення: 16.10.2025).

27. Duchene F. Xss vulnerability detection using model inference assisted evolutionary fuzzing / F. Duchene, R. Groz, S. Rawat, J. L. Richier // IEEE Fifth International Conference on Software Testing, Verification and Validation. 2012. P. 815–817.

28. WhiteHat Security's Approach to Detecting Cross-Site Request Forgery (CSRF). URL: https://www.whitehatsec.com/ (дата звернення: 01.04.2025).

29. Cross site request forgery (CSRF). URL: https://learn.snyk.io/lesson/csrf-attack/ (дата звернення: 02.10.2025).

30. Тестування на проникнення. URL: https://krlabs.com.ua/blog/testuvannya-na-pronyknennya-pentest-vid-a-do-ya/ (дата звернення: 07.10.2025).

31. OWASP CheatSheetSeries. URL: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.md (дата звернення: 20.10.2025).

32. Egele M. A survey on automated dynamic malware-analysis techniques and tools / M. Egele, T. Scholte, E. Kirda, C. Kruegel // ACM Computing Surveys. 2012. Vol. 44, No. 2. Article 6. 42 p. DOI: 10.1145/2089125.2089126.

33. Alghamdi S. M., Othathi E. S., Alsulami B. S. Detect keyloggers by using Machine Learning. 2022 Fifth National Conference of Saudi Computers Colleges (NCCC), Makkah, Saudi Arabia, 2022. P. 193–200. DOI: 10.1109/NCCC57165.2022.10067780.

34. Ferraiolo D. F. Role-based access controls / D. F. Ferraiolo, D. R. Kuhn // Proceedings of the 15th National Computer Security Conference. 1992. P. 554–563.

35. Judijanto L., Hindarto D., Wahjono S. I., Djunarto. Edge of Enterprise Architecture in Addressing Cyber Security Threats and Business Risks. International Journal Software Engineering and Computer Science (IJSECS). 2023. Vol. 3, no. 3. P. 386–396. DOI: 10.35870/ijsecs.v3i3.1816

36. Ashley T., Gourisetti S. N. G., Brown N., Bonebrake C. Aggregate attack surface management for network discovery of operational technology. Computers & Security. 2022. Vol. 123. P. 102939. DOI: 10.1016/j.cose.2022.102939

37. Sotiropoulos P., Mathas C.-M., Vassilakis C., Kolokotronis N. A Software Vulnerability Management Framework for the Minimization of System Attack Surface and Risk. Electronics. 2023. Vol. 12, no. 10. P. 2278. DOI: 10.3390/electronics12102278

38. Сандху Р. Рольова модель управління доступом / Р. Сандху, Е. Койн, Х. Файнштейн, К. Роумен ; пер. з англ. М. Горбатюк // Безпека інформаційних систем. 2021. № 3. С. 112–128.

39. Yan K., Liu X., Lu Y., Qin F. A Cyber-Physical Power System Risk Assessment Model Against Cyberattacks. IEEE Systems Journal. 2023. Vol. 17, no. 2. P. 2018–2028. DOI: 10.1109/JSYST.2022.3215591

40. Tanenbaum A. S. Modern operating systems / A. S. Tanenbaum, H. Bos. 4th ed. Boston : Pearson, 2015. 1136 p.

41. Козачок О. М. Механізми дискреційного контролю доступу в сучасних операційних системах / О. М. Козачок, С. А. Петренко // Кібербезпека: освіта, наука, техніка. 2023. № 4. С. 67–79. DOI: 10.28925/2663-4023.2023.4.6779.

42. Bishop M. Computer security: art and science / M. Bishop. 2nd ed. Boston : Addison-Wesley, 2019. 1440 p.

43. Thompson K. Reflections on trusting trust / K. Thompson // Communications of the ACM. 1984. Vol. 27, No. 8. P. 761–763. DOI: 10.1145/358198.358210.

44. Jaeger T. Analyzing integrity protection in the SELinux example policy / T. Jaeger, R. Sailer, X. Zhang // Proceedings of the 12th USENIX Security Symposium 2003. P. 59–74.

45. Гавриленко С. Ю. Автоматизована генерація політик SELinux на основі машинного навчання / С. Ю. Гавриленко, В. Ф. Челомбітько // Безпека інформації. 2023. Т. 29, № 2. С. 78–89. DOI: 10.18372/2225-5036.29.17234.

46. Ferraiolo D. F. Proposed NIST standard for role-based access control / D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, R. Chandramouli // ACM Transactions on Information and System Security. 2001. Vol. 4, No. 3. P. 224–274. DOI: 10.1145/501978.501980.

47. Kuhn D. R. Adding attributes to role-based access control / D. R. Kuhn, E. J. Coyne, T. R. Weil // IEEE Computer. 2010. Vol. 43, No. 6. P. 79–81. DOI: 10.1109/MC.2010.155.

48. American National Standards Institute. Role based access control. ANSI INCITS 359-2012 / American National Standards Institute. New York : ANSI, 2012. 56 p.

49. Петров А. В. Застосування рольової моделі для захисту серверних додатків від експлуатації вразливостей / А. В. Петров, В. М. Сидоренко // Кібербезпека: освіта, наука, техніка. 2022. № 3. С. 89–101. DOI: 10.28925/2663-4023.2022.3.89101.

50. Li N. On mutually exclusive roles and separation-of-duty / N. Li, M. V. Tripunitara, Z. Bizri // ACM Transactions on Information and System Security. 2007. Vol. 10, No. 2. Article 5. 36 p. DOI: 10.1145/1237500.1237501.

51. Крилов Д. О. Інтеграція RBAC з організаційною структурою підприємства / Д. О. Крилов, А. І. Марченко // Інформаційна безпека. 2023. № 1. С. 45–56.

52. Hu V. C. Guide to attribute based access control (ABAC) definition and considerations. NIST Special Publication 800-162 / V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone. Gaithersburg : National Institute of Standards and Technology, 2014. 54 p. DOI: 10.6028/NIST.SP.800-162.

53. Zografopoulos I., Ospina J., Liu X., Konstantinou C. Cyber-Physical Energy Systems Security: Threat Modeling, Risk Assessment, Resources, Metrics, and Case Studies. IEEE Access. 2021. Vol. 9. P. 29775–29818. DOI: 10.1109/ACCESS.2021.3058403

54. Yuan E. Attributed based access control (ABAC) for web services / E. Yuan, J. Tong // Proceedings of the IEEE International Conference on Web Services (ICWS'05). 2005. P. 561–569. DOI: 10.1109/ICWS.2005.131.

55. Шевченко О. Г. Динамічний атрибутивний контроль доступу для протидії цільовим атакам / О. Г. Шевченко, Р. В. Корольов // Безпека інформації. 2023. Т. 29, № 3. С. 134–145. DOI: 10.18372/2225-5036.29.17456.

56. Kulkarni D. Context-aware role-based access control in pervasive computing systems / D. Kulkarni, A. Tripathi // Proceedings of the 13th ACM Symposium on Access Control Models and Technologies. 2008. P. 113–122. DOI: 10.1145/1377836.1377854.

57. Bijon K. Z. A framework for risk-aware role based access control / K. Z. Bijon, R. Krishnan, R. Sandhu // Proceedings of the 6th International Conference on Security of Information and Networks. 2013. P. 462–469. DOI: 10.1145/2523514.2523590.

58. Основи інформаційної безпеки. Конспект лекцій / Б. А. Заплотинський. Київ : КІВіП НУ "ОЮА", 2017. 128 с.

59. OWASP Risk Rating Calculator. URL: https://www.RiskRatingClculator.org/docs/ (дата звернення: 10.04.2025).

60. Kim P. The Hacker Playbook 3: Practical Guide To Penetration Testing / P. Kim. Securety planet LLC, 2018. 359 p.

61. Корченко О. Г. Методологія тестування безпеки інформаційних систем / О. Г. Корченко, В. Л. Бурячок, С. О. Гнатюк. Київ : НАУ. ISBN: 978-966-598-654-3.

62. Дослідження вразливостей Web-сайтів та методів їх усунення. URL: http://phone.kpi.ua/wpcontent/uploads/2014/06/4.pdf (дата звернення: 18.10.2025).

63. Web Application Architecture: The Latest Guide 2024. URL: https://www.clickittech.com/devops/web-application-architecture/ (дата звернення: 20.10.2025).

64. Системи моніторингу та управління безпекою. URL: http://integritysys.com.ua/security/siem/ (дата звернення: 21.10.2025).

65. Shu X. Breaking the target: an analysis of target data breach and lessons learned / X. Shu, K. Tian, A. Ciambrone, D. Yao // arXiv preprint arXiv:1701.04940. 2017. 12 p.

66. Rose S. Zero trust architecture. NIST Special Publication 800-207 / S. Rose, O. Borchert, S. Mitchell, S. Connelly. Gaithersburg : National Institute of Standards and Technology, 2020. 59 p. DOI: 10.6028/NIST.SP.800-207.

67. Ouaddah A. FairAccess: a new blockchain-based access control framework for the Internet of Things / A. Ouaddah, A. Abou Elkalam, A. Ait Ouahman // Security and Communication Networks. 2016. Vol. 9, No. 18. P. 5943–5964. DOI: 10.1002/sec.1748.

68. Chen L. Report on post-quantum cryptography. NIST Internal Report 8105 / L. Chen, S. Jordan, Y. K. Liu, D. Moody, R. Peralta, R. Perlner, D. Smith-Tone. Gaithersburg : National Institute of Standards and Technology, 2016. 15 p. DOI: 10.6028/NIST.IR.8105.

69. Bernstein D. Containers and cloud: from LXC to Docker to Kubernetes / D. Bernstein // IEEE Cloud Computing. 2014. Vol. 1, No. 3. P. 81–84. DOI: 10.1109/MCC.2014.51.

70. Романенко В. О. Аналіз ефективності контейнеризації для ізоляції потенційно небезпечних додатків / В. О. Романенко, Л. О. Кіріченко // Кібербезпека: освіта, наука, техніка. 2023. № 2. С. 112–125. DOI: 10.28925/2663-4023.2023.2.112125.

71. Newman S. Building microservices: designing fine-grained systems / S. Newman. 2nd ed. Sebastopol : O'Reilly Media, 2021. 612 p.

72. Купревич Т. С. Інтеграція систем контролю доступу з SIEM для адаптивного захисту / Т. С. Купревич, В. С. Василенко // Безпека інформації. 2023. Т. 29, № 1. С. 56–68. DOI: 10.18372/2225-5036.29.16892.

ДОДАТОК А

Перелік публікацій за темою кваліфікаційної роботи

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XVII Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2025»

14-15 листопада 2025

Хмельницький 2025

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК - 2025

XVII Всеукраїнська науково-практична конференція

Метою конференції є висвітлення актуальних проблем комп'ютерних наук, інформатики та інформаційних технологій.

Робочі мови конференції:

українська, англійська

СЕКЦІЇ КОНФЕРЕНЦІЇ:

1. Комп'ютерні науки, штучний інтелект та прикладні інформаційні технології.
2. Комп'ютерна інженерія та системи захисту інформації.
3. Математичне моделювання та інженерія програмного забезпечення
4. Телерадіокомунікації, медійні та комунікаційні системи.
5. Проблеми впровадження інформаційних технологій у виробництво та управління.

СПИСОК ОРГАНІЗАЦІЙ, ПРЕДСТАВНИКИ ЯКИХ БРАЛИ УЧАСТЬ У РОБОТІ КОНФЕРЕНЦІЇ:

Донбаська державна машинобудівна академія
Інститут кібернетики імені В. М. Глушкова НАН України
Кам'янський енергетичний фаховий коледж
Київський національний університет імені Т. Г. Шевченка
Національного аерокосмічного університету імені М. Є. Жуковського «Харківський авіаційний інститут»
Національний технічний університет «Харківський політехнічний інститут»
Сумський державний університет
Харківський національний університет радіоелектроніки
Хмельницький національний університет
Хмельницький фаховий економіко-технологічний коледж УЕП

ОРГКОМІТЕТ КОНФЕРЕНЦІЇ:

СИНЮК О. М. – голова оргкомітету, проректор Хмельницького національного університету з наукової роботи, доктор технічних наук, професор.

ГОВОРУЩЕНКО Т. О. – заступник голови оргкомітету, декан факультету інформаційних технологій Хмельницького національного університету, доктор технічних наук, професор.

БАРМАК О. В. – заступник голови оргкомітету, завідувач кафедри комп'ютерних наук Хмельницького національного університету, доктор технічних наук, професор.

САВЕНКО О. С. – професор кафедри комп'ютерної інженерії та інформаційних систем Хмельницького національного університету, доктор технічних наук, професор.

ВИСОЦЬКА О. В. – завідувач кафедри радіоелектронних та біомедичних комп'ютеризованих засобів і технологій Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут», доктор технічних наук, професор.

ЛАВРОВ Є. А. – доктор технічних наук, професор (Сумський державний університет).

ТІМОФЄЄВА Л. В. – відповідальна за студентську науково-дослідну роботу ХНУ.

МАЗУРЕЦЬ О. В. – секретар конференції, доцент кафедри комп'ютерних наук Хмельницького національного університету, кандидат технічних наук, доцент.

МОЛЧАНОВА М. О. – секретар конференції, старший викладач кафедри комп'ютерних наук Хмельницького національного університету, доктор філософії з комп'ютерних наук.

КОНТАКТНА ІНФОРМАЦІЯ:

e-mail для листування: *apkt.khnu@gmail.com*

УДК 004.4

Савчук В.В., Гартрамф М.С., Шкребета В.С., Муляр І.В.

Хмельницький національний університет

МЕТОД ЗАХИСТУ ВЕБЗАСТОСУНКІВ НА ОСНОВІ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ТРАФІКУ

Розглянуто методи захисту інформаційних систем від загрозливих програм на основі механізмів контролю доступу та розмежувальних політик. Запропоновано підхід до побудови багаторівневої системи захисту, що поєднує дискреційний, мандатний та рольовий контроль доступу з динамічними політиками безпеки. Результати дослідження демонструють, що правильно налаштовані механізми контролю доступу можуть суттєво знизити ймовірність успішного зараження системи та обмежити масштаби можливих наслідків.

Methods of protecting information systems from malicious programs based on access control mechanisms and separation policies are considered. An approach to building a multi-level protection system is proposed, combining discretionary, mandatory, and role-based access control with dynamic security policies. The results of the study demonstrate that properly configured access control mechanisms can significantly reduce the likelihood of successful system infection and limit the scale of possible consequences.

Шкідливе програмне забезпечення залишається однією з найважливіших проблем сучасної інформаційної безпеки. За даними дослідницьких компаній, щодня виявляється понад 450 000 нових зразків шкідливого програмного забезпечення, що ставить безпрецедентні виклики перед традиційними засобами безпеки [1]. Класичні антивірусні рішення, засновані на аналізі сигнатур, все частіше виявляються неефективними проти нових і модифікованих загроз, особливо коли зловмисники використовують техніки обфускації та поліморфізму [2].

У цьому контексті особливого значення набувають методи превентивного захисту, серед яких провідну роль відіграють механізми контролю доступу до інформаційних ресурсів. На відміну від реактивних методів виявлення загроз, контроль доступу запобігає виконанню шкідливих дій до їх здійснення, обмежуючи можливості потенційно небезпечного програмного забезпечення незалежно від його конкретної реалізації [3].

Метою даного дослідження є аналіз ефективності різних моделей контролю доступу в боротьбі зі шкідливими програмами та розробка рекомендацій щодо побудови комплексної системи захисту на основі політики розділення доступу. Актуальність дослідження обумовлена необхідністю пошуку альтернативних і

доповнюючих методів захисту в умовах постійної еволюції загроз і зростаючої складності інформаційних систем.

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. 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

Рисунок 1 – Аналіз загрозливих програм за використання підходу реінжинірингу

Технічні методи ухилення від статичного аналізу зловмисниками призвели до розвитку динамічного аналізу. Виконстання обфускації коду 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.

Традиційний статичний контроль доступу, що базується на заздалегідь визначених політиках, може бути недостатнім проти складних цільових атак та загроз нульового дня. Динамічний контроль доступу, що адаптується на основі аналізу поточної ситуації безпеки, дозволяє автоматично підвищувати рівень захисту при виявленні підозрілої активності.

Системи виявлення аномалій аналізують поведінку процесів та користувачів, ідентифікуючи відхилення від встановлених базових показників. Машинне навчання дозволяє автоматично виявляти патерни, що характерні для шкідливої активності, навіть якщо конкретна загроза є новою та невідомою. При виявленні аномалій система може автоматично коригувати політики контролю доступу, обмежуючи можливості підозрілих процесів або користувачів до мінімуму. Наприклад, якщо процес починає звертатися до незвично великої кількості файлів, що може свідчити про активність програми-вимагача, його права доступу можуть бути автоматично обмежені до моменту з'ясування ситуації [4].

Контроль доступу на основі атрибутів (ABAC) є найбільш гнучкою та виразною моделлю серед сучасних підходів до управління доступом. На відміну від традиційних моделей, заснованих на ідентичності або ролях користувачів, ABAC приймає рішення на основі атрибутів суб'єктів, об'єктів, операцій та контексту середовища. Атрибутами можуть бути будь-які властивості, такі як відділ користувача, час доступу, рівень конфіденційності документа, тип операції, географічне розташування або поточний рівень загрози безпеці системи.

Політики ABAC формулюються як правила, що визначають умови надання доступу на основі комбінацій атрибутів. Наприклад, правило може дозволяти тільки бухгалтерам читати фінансові документи в робочий час з корпоративної мережі, блокуючи доступ у всіх інших випадках. Така гнучкість дозволяє реалізовувати складні політики безпеки, що відображають реальні бізнес-вимоги та автоматично адаптуються до змін контексту.

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. 2.

Розроблено методи захисту від шкідливого програмного забезпечення з урахуванням вимог до побудови безпечної системи, спрямованої на запобігання витоку прав доступу під час її функціонування. Показано, що реалізація цих вимог на практиці істотно змінює традиційні підходи до організації контролю доступу до ресурсів, забезпечуючи більш гнучке та надійне управління дозволами користувачів і процесів.

Створено політику безпеки, яка підтверджує можливість практичної реалізації запропонованих методів і демонструє їх ефективність, а також зручність

адміністрування та експлуатації системи захисту. Політика визначає механізми моніторингу дій користувачів, виявлення ненормальної активності, динамічного оновлення прав доступу та ізоляції потенційно небезпечних процесів.

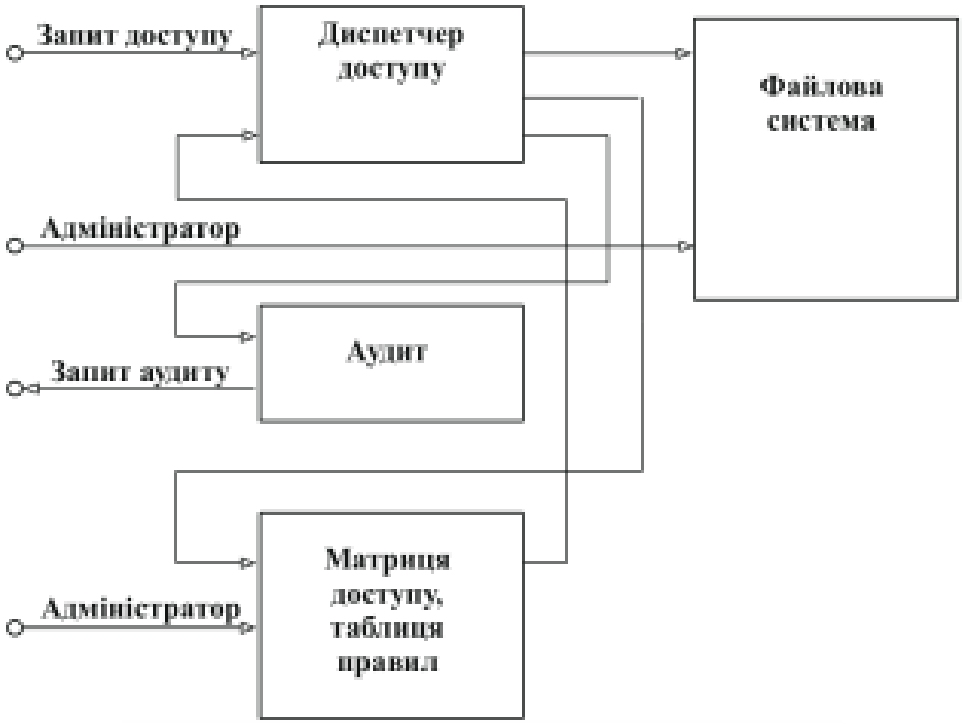

Рисунок 2 – Реалізація запропонованого рішення

Це підвищує стійкість системи до внутрішніх і зовнішніх загроз, знижує ризик несанкціонованого доступу та забезпечує відповідність сучасним вимогам кібербезпеки в інформаційних і обчислювальних середовищах.

Перелік посилань

1. AV-TEST Institute. Malware statistics & trends report 2024. AV-TEST GmbH, 2024. 45 p.
2. Information Security Handbook/ Noor Zaman Jhanjhi, Khalid Hussain, Mamoona Humayun, Azween Bin Abdullah, João Manuel R.S. Tavares. CRC Press, 2022. 250 p.
3. Інформаційна безпека в комп'ютерних мережах: навч. посіб./ О.А. Смірнов, Конопліцька-О.К. Слободенюк, С.А. Смірнов, К.О. Буравченко, Т.В. Смірнова, Л.І. Поліщук. Кропивницький: Видавець Лисенко В. Ф., 2020. 295 с.
4. Research of the Neural Network Module for Detecting Anomalies in Network Traffic/ Klots Y., Titova V., Petliak N., Cheshun V., Salem A.-B.M. CEUR Workshop Proceedings, 2022. 3156. P. 378–389

ДОДАТОК Б

Програмний код модуля аудиту

```
#!/usr/bin/env python3
import os
import json
import hashlib
import sqlite3
from datetime import datetime
from enum import Enum
from typing import Dict, List, Optional, Any
from dataclasses import dataclass, asdict
import threading
import queue

class EventType(Enum):

    FILE_ACCESS = "file_access"
    FILE_MODIFICATION = "file_modification"
    PROCESS_EXECUTION = "process_execution"
    AUTHENTICATION = "authentication"
    PRIVILEGE_ESCALATION = "privilege_escalation"
    NETWORK_ACTIVITY = "network_activity"
    DATABASE_ACCESS = "database_access"
    CONFIGURATION_CHANGE = "configuration_change"
    SYSTEM_CALL = "system_call"
    SECURITY_VIOLATION = "security_violation"

class EventSeverity(Enum):

    LOW = 1
    MEDIUM = 2
    HIGH = 3
    CRITICAL = 4

class EventResult(Enum):
    SUCCESS = "success"
    FAILURE = "failure"
    DENIED = "denied"
    ERROR = "error"

@dataclass
class AuditEvent:
```

```
    timestamp: str
    event_type: EventType
    severity: EventSeverity
    user_id: str
    process_id: int
    process_name: str
    object_path: str
    operation: str
    result: EventResult
    details: Dict[str, Any]
    source_ip: Optional[str] = None
    parent_process: Optional[str] = None

    def to_dict(self) -> Dict:

        data = asdict(self)
        data['event_type'] = self.event_type.value
        data['severity'] = self.severity.value
        data['result'] = self.result.value
        return data

class Filter:

    def __init__(self, name: str):
        self.name = name

    def process(self, event: AuditEvent) -> tuple[bool, AuditEvent]:
        raise NotImplementedError

class InputFilter(Filter):

    def __init__(self):
        super().__init__("InputFilter")
        self.required_fields    =    ['user_id',    'process_id',    'object_path',
'operation']

    def process(self, event: AuditEvent) -> tuple[bool, AuditEvent]:

        if not event.user_id or not event.object_path:
            event.details['validation_error'] = "Missing required fields"
            event.severity = EventSeverity.LOW
            return False, event
```

```
        if not isinstance(event.process_id, int) or event.process_id < 0:
            event.details['validation_error'] = "Invalid process ID"
            return False, event

        # Санітизація шляхів
        event.object_path = os.path.normpath(event.object_path)

        return True, event

class SourceFilter(Filter):

    def __init__(self):
        super().__init__("SourceFilter")
        self.trusted_users = {'root', 'admin', 'system'}

    def process(self, event: AuditEvent) -> tuple[bool, AuditEvent]:
        event.details['trusted_user'] = event.user_id in self.trusted_users
        event.details['timestamp_unix'] = datetime.now().timestamp()
        event.details['session_id'] = self._generate_session_id(event)

        # Перевірка часу (підозрілі операції вночі)
        current_hour = datetime.now().hour
        if current_hour < 6 or current_hour > 22:
            event.details['suspicious_time'] = True
            if event.severity.value < EventSeverity.MEDIUM.value:
                event.severity = EventSeverity.MEDIUM

        return True, event

    def _generate_session_id(self, event: AuditEvent) -> str:

        data = f"{event.user_id}{event.process_id}{event.timestamp}"
        return hashlib.md5(data.encode()).hexdigest()[:16]


class ThreatAnalysisFilter(Filter):

    def __init__(self):
        super().__init__("ThreatAnalysisFilter")
        self.suspicious_patterns = [
            '/etc/passwd', '/etc/shadow', 'cmd.exe', 'powershell.exe',
            '.exe', '.bat', '.vbs', '.js', '/proc/', '/sys/'
```

```
        ]
        self.malicious_operations = ['delete', 'modify', 'execute_hidden']
        self.event_history = []
        self.max_history = 1000

    def process(self, event: AuditEvent) -> tuple[bool, AuditEvent]:

        threat_score = 0
        threats = []

        for pattern in self.suspicious_patterns:
            if pattern in event.object_path.lower():
                threat_score += 2
                threats.append(f"Suspicious path pattern: {pattern}")


        if event.operation in self.malicious_operations:
            threat_score += 3
            threats.append(f"Dangerous operation: {event.operation}")


        recent_similar = self._count_recent_similar_events(event)
        if recent_similar > 10:
            threat_score += 4
            threats.append(f"High frequency detected: {recent_similar} events")


        if event.event_type == EventType.PRIVILEGE_ESCALATION:
            threat_score += 5
            threats.append("Privilege escalation attempt")


        if threat_score >= 5:
            event.severity = EventSeverity.CRITICAL
            event.event_type = EventType.SECURITY_VIOLATION
        elif threat_score >= 3:
            event.severity = EventSeverity.HIGH

        event.details['threat_score'] = threat_score
        event.details['threats'] = threats


        self.event_history.append(event)
```

```
        if len(self.event_history) > self.max_history:
            self.event_history.pop(0)

        return True, event

    def _count_recent_similar_events(self, event: AuditEvent) -> int:

        count = 0
        current_time = datetime.now()

        for hist_event in self.event_history[-100:]:
            hist_time = datetime.fromisoformat(hist_event.timestamp)
            if (current_time - hist_time).seconds <= 60:
                if (hist_event.user_id == event.user_id and
                    hist_event.event_type == event.event_type):
                    count += 1

        return count


class ClassificationFilter(Filter):

    def __init__(self):
        super().__init__("ClassificationFilter")
        self.categories = {
            'system': ['system32', 'windows', 'etc', 'bin', 'sbin'],
            'executable': ['.exe', '.bat', '.cmd', '.sh', '.dll'],
            'config': ['.conf', '.config', '.ini', '.xml'],
            'data': ['.doc', '.pdf', '.txt', '.xlsx']
        }

    def process(self, event: AuditEvent) -> tuple[bool, AuditEvent]:
        obj_category = self._classify_object(event.object_path)
        event.details['object_category'] = obj_category

        access_type = self._classify_access(event.operation)
        event.details['access_type'] = access_type

        if obj_category == 'system' and access_type in ['write', 'delete']:
            if event.severity.value < EventSeverity.HIGH.value:
                event.severity = EventSeverity.HIGH

        if obj_category == 'executable' and event.operation == 'execute':
```

```
            event.details['requires_verification'] = True

        return True, event

    def _classify_object(self, path: str) -> str:
        path_lower = path.lower()

        for category, patterns in self.categories.items():
            for pattern in patterns:
                if pattern in path_lower:
                    return category

        return 'unknown'

    def _classify_access(self, operation: str) -> str:
        operation_lower = operation.lower()

        if any(op in operation_lower for op in ['read', 'open', 'view']):
            return 'read'
        elif any(op in operation_lower for op in ['write', 'modify', 'update']):
            return 'write'
        elif any(op in operation_lower for op in ['delete', 'remove']):
            return 'delete'
        elif any(op in operation_lower for op in ['execute', 'run']):
            return 'execute'

        return 'other'


class OutputFilter(Filter):

    def __init__(self):
        super().__init__("OutputFilter")

    def process(self, event: AuditEvent) -> tuple[bool, AuditEvent]:
        event.details['filter_chain_completed'] = True
        event.details['output_timestamp'] = datetime.now().isoformat()

        event_hash = self._compute_event_hash(event)
        event.details['event_hash'] = event_hash

        if event.severity.value >= EventSeverity.HIGH.value:
            event.details['alert_required'] = True
```

```
        event.details['log_format_version'] = '1.0'

        return True, event

    def _compute_event_hash(self, event: AuditEvent) -> str:
        data = f"{event.timestamp}{event.event_type}{event.user_id}{event.object_path}"
        return hashlib.sha256(data.encode()).hexdigest()[:32]


class AuditConfiguration:

    def __init__(self):
        self.enabled_event_types = set(EventType)
        self.min_severity = EventSeverity.LOW
        self.monitored_users = set()  # Порожній = всі користувачі
        self.excluded_users = {'backup', 'monitoring'}
        self.monitored_paths = []  # Порожній = всі шляхи
        self.max_events_per_minute = 1000

    def should_audit(self, event: AuditEvent) -> bool:
        if event.event_type not in self.enabled_event_types:
            return False

        if event.severity.value < self.min_severity.value:
            return False

        # Перевірка користувача
        if event.user_id in self.excluded_users:
            return False

        if self.monitored_users and event.user_id not in self.monitored_users:
            return False

        return True


class AuditLogger:

    def __init__(self, db_path: str = "audit_logs.db"):
        self.db_path = db_path
        self._init_database()
```

```
        self.lock = threading.Lock()

    def _init_database(self):
        conn = sqlite3.connect(self.db_path)
        cursor = conn.cursor()

        cursor.execute("""
            CREATE TABLE IF NOT EXISTS audit_events (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                timestamp TEXT NOT NULL,
                event_type TEXT NOT NULL,
                severity INTEGER NOT NULL,
                user_id TEXT NOT NULL,
                process_id INTEGER,
                process_name TEXT,
                object_path TEXT,
                operation TEXT,
                result TEXT,
                details TEXT,
                event_hash TEXT,
                created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            )
        """)

        cursor.execute("""
            CREATE INDEX IF NOT EXISTS idx_timestamp ON audit_events(timestamp)
        """)
        cursor.execute("""
            CREATE INDEX IF NOT EXISTS idx_user ON audit_events(user_id)
        """)
        cursor.execute("""
            CREATE INDEX IF NOT EXISTS idx_severity ON audit_events(severity)
        """)

        conn.commit()
        conn.close()

    def log_event(self, event: AuditEvent):
        """Збереження події в журнал"""
        with self.lock:
            conn = sqlite3.connect(self.db_path)
            cursor = conn.cursor()
```

```
            cursor.execute("""
                INSERT INTO audit_events
                (timestamp,    event_type,    severity,    user_id,    process_id,
process_name,
                 object_path, operation, result, details, event_hash)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
            """, (
                event.timestamp,
                event.event_type.value,
                event.severity.value,
                event.user_id,
                event.process_id,
                event.process_name,
                event.object_path,
                event.operation,
                event.result.value,
                json.dumps(event.details),
                event.details.get('event_hash', '')
            ))

            conn.commit()
            conn.close()

    def get_recent_events(self, limit: int = 100) -> List[Dict]:
        """Отримання останніх подій"""
        conn = sqlite3.connect(self.db_path)
        cursor = conn.cursor()

        cursor.execute("""
            SELECT * FROM audit_events
            ORDER BY id DESC
            LIMIT ?
        """, (limit,))

        columns = [description[0] for description in cursor.description]
        events = []

        for row in cursor.fetchall():
            event_dict = dict(zip(columns, row))
            event_dict['details'] = json.loads(event_dict['details'])
            events.append(event_dict)

        conn.close()
```

```
    return events

def search_events(self, user_id: str = None, event_type: str = None,
                  severity: int = None, start_time: str = None) -> List[Dict]:
    """Пошук подій за критеріями"""
    conn = sqlite3.connect(self.db_path)
    cursor = conn.cursor()

    query = "SELECT * FROM audit_events WHERE 1=1"
    params = []

    if user_id:
        query += " AND user_id = ?"
        params.append(user_id)

    if event_type:
        query += " AND event_type = ?"
        params.append(event_type)

    if severity:
        query += " AND severity >= ?"
        params.append(severity)

    if start_time:
        query += " AND timestamp >= ?"
        params.append(start_time)

    query += " ORDER BY id DESC LIMIT 1000"

    cursor.execute(query, params)
    columns = [description[0] for description in cursor.description]
    events = []

    for row in cursor.fetchall():
        event_dict = dict(zip(columns, row))
        event_dict['details'] = json.loads(event_dict['details'])
        events.append(event_dict)

def stop(self):
    self.running = False
    if self.worker_thread:
        self.worker_thread.join()
    print("[AuditSystem] Система аудиту зупинена")
```

```
    def _process_events(self):
        while self.running:
            try:
                event = self.event_queue.get(timeout=1)
                self._process_single_event(event)
                self.event_queue.task_done()
            except queue.Empty:
                continue
            except Exception as e:
                print(f"[AuditSystem] Помилка обробки події: {e}")

    def _process_single_event(self, event: AuditEvent):
        for filter_obj in self.filters:
            should_continue, event = filter_obj.process(event)

            if not should_continue:
                print(f"[{filter_obj.name}] Подія відхилена")
                return

        self.logger.log_event(event)

        if event.severity.value >= EventSeverity.HIGH.value:
            self._alert_critical_event(event)

    def _alert_critical_event(self, event: AuditEvent):
        print(f"\n{'='*80}")
        print(f"[КРИТИЧНА ПОДІЯ] {event.event_type.value.upper()}")
        print(f"Час: {event.timestamp}")
        print(f"Користувач: {event.user_id}")
        print(f"Процес: {event.process_name} (PID: {event.process_id})")
        print(f"Об'єкт: {event.object_path}")
        print(f"Операція: {event.operation}")
        print(f"Результат: {event.result.value}")
        print(f"Загрози: {event.details.get('threats', [])}")
        print(f"{'='*80}\n")

if __name__ == "__main__":
    main()
```

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
здобувача вищої освіти
Савчук Владислав Вікторович
студента ФІТ, 2 курсу, групи КБЗІм-24-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений. Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений. Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1.12.25
дата


підпис

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 7%**

	Document		Sum coincidence on the DB	
ID: 251495 Title: "Метод захисту вебзстосунків від завантаження і виконання підозрілих файлів " Added in a DB: 2025-12-03 Authors: Савчук Владислав Вікторович Heads: Муляр І.В. Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	107840	1623	799 (1%)	12 (1%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Савчук Владислав Вікторович

Співавтор:

Назва: Метод захисту вебзстосунків від завантаження і виконання підозрілих файлів

Науковий керівник: Муляр Ігор Володимирович

Підрозділ: Кафедра кібербезпеки

Коефіцієнт подібності 1:2.2%

Коефіцієнт подібності 2:0.2%

Мікропробіли: 0

Заміна букв: 2

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-12-03 22:48:06.0

Після аналізу Звіту подібності констатую наступне:

☐ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата експерт

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи: Метод захисту вебзстосунків від завантаження і виконання підозрілих файлів

Автор: Савчук Владислав Вікторович

Освітня програма: Кібербезпека та захист інформації

Рівень вищої освіти: другий (магістерський)

Спеціальність: 125 – Кібербезпека та захист інформації

Науковий керівник: Муляр І.В. , к.т.н., доц.

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту	відповідає
1.2	Виявлені запозичення не є академічним 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	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою StrikePlagiarism складає 97.8%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism складає 100%

Згідно з Положенням про систему забезпечення академічної доброчесності у ХНУ (https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-systemu-zabezpechennya-akademichnoyi-dobrochesnosti.pdf, Додаток В) кваліфікаційна робота, виконана за освітньо-професійною програмою, кількісні показники рівня унікальності тексту у відсотках до загального обсягу матеріалу в якій складає 75-100 %, визнається роботою з високим рівнем унікальності тексту: «Текст вважається унікальним і не потребує додаткових дій щодо запобігання неправомірним запозиченням».

Дата: 03.12.2025

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

Гарант освітньої програми _____ Віра ТІТОВА

Керівник кваліфікаційної роботи _____ Ігор МУЛЯР

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «магістр»

Студент Савчук Владислав Вікторович

Тема Метод захисту вебзстосунків від завантаження і виконання підозрілих файлів

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека та захист інформайції

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «магістр»:

кількість листів креслень - ; кількість сторінок записки 86

1. Короткий зміст кваліфікаційної роботи та прийнятих рішень В рамках роботи досліджено фактори, що впливають на ефективність функціонування вебдодатків, а також проаналізовано критерії та існуючі методи їх оцінювання. розроблено моделі системи захисту вебресурсів, 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 у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині.

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 розглядаються питання особливостей функціонування корпоративних мереж в. Наступні розділи присвячені розробці та реалізації алгоритму та методу тестування безпеки вебресурів

4. Позитивні сторони роботи Кваліфікаційна робота містить ряд інноваційних рішень, зокрема запропонований метод забезпечує комплексний підхід до діагностики безпеки вебдодатків, поєднуючи можливості розмежування доступу, політики безпеки,

5. Негативні сторони роботи Впровадження розробленої моделі та методу ускладняється масштабними та складними топологіями.

6. Оцінка графічного оформлення та пояснювальної записки роботи
Пояснювальна записка відповідає нормам для її оформлення.

7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує позитивної оцінки. Але матеріал роботи не достатньо структурований, чіткий та послідовний. Усі розділи роботи відповідають завданню, що дозволяє розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Робота має наукову новизну, але вона не достатью висвітлена, та практичну цінність.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «задовільно» 65 балів

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
Підченко Сергій Костянтинович, професор кафедри ТМІТ, доктор технічних наук

«8» 12 2025. (підпис)