Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Зацепіна Костянтина Олександровича

на здобуття ступеня вищої освіти магістра

Метод тестування безпеки програмного забезпечення для захисту інформації в корпоративній мережі

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

Шифр КРМКБЗІ. 2301139.23.01.10 ПЗ

Виконав студент 2 курсу група КБЗІм-23-1 Костянтин ЗАЦЕПІН
Керівник канд. техн. наук, доцент Ігор МУЛЯР
Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

16 12 2024 р.

Хмельницький 2024

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

2 09 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Зацепіну Костянтину Олександровичу

1 Тема роботи Метод тестування безпеки програмного забезпечення для захисту інформації в корпоративній мережі
Керівник роботи канд.техн.наук, доцент Ігор МУЛЯР
Затверджено наказом ректора університету від 26 08 2024 № 60
2 Строк подання студентом кваліфікаційної роботи на кафедру 2.12.2024
3 Вихідні дані до роботи програмне забезпечення корпоративної мережі, GERT-мережа, засоби тестування

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Провести аналіз чинників, що впливають на ефективне функціонування мережі, проаналізувати методології тестування безпеки програмного забезпечення корпоративної , розробити математичні моделі процесу тестування корпоративної мережі, розробити метод тестування безпеки програмного забезпечення мережі, перевірити результати моделювання

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 2 09 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Ґрунтовне ознайомлення та дослідження предметної галузі	14.09.2024	Виконано
Визначення змісту, структури кваліфікаційної роботи	22.09.2024	Виконано
Підготовка першого розділу кваліфікаційної роботи	29.09.2024	Виконано
Підготовка другого розділу кваліфікаційної роботи	11.10.2024	Виконано
Підготовка третього розділу кваліфікаційної роботи	20.10.2024	Виконано
Підготовка статті/тези за темою кваліфікаційної роботи	4.11.2024	Виконано
Підготовка четвертого розділу кваліфікаційної роботи	16.11.2024	Виконано
Підготовка та оформлення ілюстративного матеріалу	24.11.2024	Виконано
Оформлення кваліфікаційної роботи	24.11.2024	Виконано
Попередній захист кваліфікаційної роботи	27.11.2024	Виконано
Захист кваліфікаційної роботи на засіданні ЕК	17.12.2024	Виконано

Студент _____ Костянтин ЗАЦЕПІН

Керівник кваліфікаційної роботи _____ Ігор МУЛЯР

АНОТАЦІЯ

Тема кваліфікаційної роботи: Метод тестування безпеки програмного забезпечення для захисту інформації в корпоративній мережі

Автор роботи: Зацепін Костянтин Олександрович

Керівник роботи: к.т.н., доц. Муляр Ігор Володимирович

Загальний обсяг роботи: 82 сторінок, 26 рисунків, 2 додатки, 61 посилань.

Ключові слова: Захист інформації, тестування програмного забезпечення.

Метою кваліфікаційної роботи є визначення оптимального рівня захищеності програмного забезпечення в корпоративній мережі через створення та впровадження ефективних методів його тестування.

Було створено математичну модель кібератаки, яка дає змогу дослідити ключові етапи проведення атаки та розробити практичні рекомендації щодо захисту мережі від подібних загроз. Удосконалено також математичну модель діагностики системи управління мережевими ресурсами, що сприяло підвищенню ефективності тестування безпеки програмного забезпечення корпоративної мережі. Результати моделювання продемонстрували можливість покращення ефективності тестування безпеки ПЗ на 4%.

2.12.2024

ANNOTATION

Theme of qualification work: Software security testing method for protecting information in a corporate network

Author of the work: Zatsepin Kostiantyn

Mentor: Ph.D. Muliar Ihor

Total volume of work: 82 pages, 26 figures, 2 appendices, 61 links.

Keywords: information security, software testing.

The purpose of the qualification work is to determine the optimal level of software security in a corporate network through the creation and implementation of effective methods for testing it.

A mathematical model of a cyberattack was created, which allows us to study the key stages of the attack and develop practical recommendations for protecting the network from such threats. The mathematical model for diagnosing the network resource management system was also improved, which helped to increase the efficiency of testing the security of corporate network software. The modelling results demonstrated the possibility of improving the efficiency of software security testing by 4%.

2.12.2024

ЗМІСТ

ВСТУП

Гарантування захищеності даних виступає фундаментальним аспектом для корпорацій та освітніх закладів у їхній щоденній роботі. Один із дієвих методів аналізу рівня захисту та мінімізації загроз - це проведення випробувань на вторгнення (penetration testing або pentest) [1]. Такий аудит являє собою санкціонований експеримент із доступу до комп'ютерних систем задля виявлення слабких місць [2]. При цьому застосовуються техніки, аналогічні діям зловмисників під час реальних атак. Подібний підхід допомагає вчасно впровадити необхідні механізми захисту, перш ніж уразливості будуть знайдені та використані небажаними особами.

Фахівці з кібербезпеки розрізняють декілька ключових різновидів тестування: зовнішнє дослідження периметру мережі, внутрішній аудит локальної інфраструктури та оцінка стійкості веб-додатків. Кожен тип має власні методології та інструменти виявлення потенційних проблем. Результати таких перевірок дозволяють створити комплексну картину наявних ризиків та розробити ефективну стратегію посилення захисту критично важливих ресурсів організації.

Особливу увагу варто приділяти документуванню знайдених вразливостей та наданню чітких рекомендацій щодо їх усунення. Це допомагає технічним командам швидко реагувати на виявлені недоліки та впроваджувати відповідні контрзаходи. Регулярне проведення подібних тестів також сприяє підтримці належного рівня обізнаності персоналу щодо актуальних кіберзагроз.

У сучасному світі кожна організація усвідомлює важливість перевірки захищеності інформаційних систем. Найбільшу увагу привертають програмні рішення для інфраструктури стратегічного значення. Їхня надійність має підтверджуватися регулярними випробуваннями впродовж усього життєвого циклу. Однак брак стандартизованих протоколів оцінки та уніфікованих підходів до аудиту суттєво знижує результативність захисних заходів, що призводить до зростання успішних кіберінцидентів [3]. Статистика демонструє тривожну

тенденцію - понад 70% підприємств зазнали негативного впливу від несанкціонованого доступу до власних активів, що спричинило значні матеріальні збитки та репутаційні втрати [4].

Додаткові складнощі створюють механізми захисту програмних продуктів від аналізу - компресія, криптографічні перетворення, динамічна модифікація коду тощо. Це породжує дисонанс між очікуваним рівнем захищеності та реальними можливостями наявного інструментарію для його оцінювання.

Науковці активно працюють над вдосконаленням методологій тестування безпеки. Зокрема, розробляються нові підходи до автоматизованого виявлення вразливостей, створюються інтелектуальні системи аналізу поведінки застосунків, впроваджуються передові практики безперервної інтеграції засобів безпеки у процес розробки. Важливим напрямком є також розвиток технологій деобфускації та аналізу захищеного коду.

Окрема увага приділяється створенню галузевих стандартів та методичних рекомендацій щодо проведення комплексного тестування безпеки. Це включає розробку типових сценаріїв перевірки, критеріїв оцінки результатів та механізмів документування виявлених проблем. Такий системний підхід дозволить підвищити якість та ефективність процесів забезпечення кібербезпеки в організаціях різного профілю.

Наукові дослідження у сфері кібербезпеки та методологій тестування активно розвиваються як в Україні, так і за кордоном, залучаючи все більше талановитих дослідників.

В українському науковому просторі помітний значний внесок професора Олександра Корченка з Національного авіаційного університету, який розробив фундаментальні методи оцінювання ризиків інформаційної безпеки та створив потужну наукову школу захисту інформації [5]. Важливі дослідження проводить Володимир Бурячок у Київському університеті імені Бориса Грінченка, фокусуючись на технологіях проникнення та методах виявлення вразливостей. У Національному університеті "Львівська політехніка" плідно працюють Микола Карпінський, який розвиває методи криптографічного захисту та аналізу безпеки

мереж, та Валерій Дудикевич, котрий зробив значний внесок у розвиток технічного захисту інформації. Вагомі результати у дослідженні методів тестування безпеки програмного забезпечення демонструє Юрій Щебланін з Харківського національного університету радіоелектроніки.

На світовій арені визначні досягнення належать Bruce Schneier, всесвітньо відомому криптографу та експерту з комп'ютерної безпеки зі США. Значний вплив на розвиток теорії інформаційної безпеки здійснив Ross Anderson з Кембриджського університету. Надійність комп'ютерних систем та мережеву безпеку грунтовно досліджує Peter G. Neumann з SRI International. Dorothy Denning з Джорджтаунського університету вважається піонером у галузі комп'ютерної безпеки, а Eugene Spafford з Університету Пердью зробив вагомий внесок як експерт з інформаційної безпеки та комп'ютерної криміналістики [6].

Сучасні дослідження цих та інших науковців охоплюють розробку інноваційних методів виявлення та запобігання кібератакам, створення інтелектуальних систем аналізу загроз, вдосконалення технологій тестування на проникнення. Особлива увага приділяється розвитку методів захисту від соціальної інженерії, покращенню алгоритмів криптографічного захисту та розробці стандартів оцінки безпеки [7].

Результати досліджень регулярно представляються науковій спільноті через публікації у провідних фахових виданнях та на престижних міжнародних конференціях, таких як Black Hat, DEF CON та RSA Conference. Галузь кібербезпеки динамічно розвивається, постійно залучаючи нових талановитих дослідників та формуючи потужні наукові школи, які суттєво впливають на розвиток методів забезпечення інформаційної безпеки у глобальному масштабі.

Актуальність дослідження зумовлена стрімким зростанням кількості кіберзагроз та необхідністю забезпечення надійного захисту інформаційних систем організацій. У сучасному цифровому середовищі питання інформаційної безпеки набуває критичного значення для функціонування корпоративних мереж.

Метою кваліфікаційної роботи є визначення оптимального рівня захищеності програмного забезпечення в корпоративній мережі через створення

та впровадження ефективних методів його тестування.

Для досягнення мети в кваліфікаційній роботі потрібно вирішити наступні завдання:

– провести оцінку ймовірності успішного проникнення в систему, враховуючи рівень захисту та існуючі механізми безпеки;

– моделювання різноманітних сценаріїв атак, що дає можливість оцінити ефективність існуючих заходів безпеки в умовах реальних загроз;

– обрати метод аналізу коду для виділення алгоритму;

– змоделювати початковий етап кібератаки до ресурсів корпоративної мережі;

– перевірити ефективності розроблених моделей.

Об'єктом дослідження є процес забезпечення безпеки програмного забезпечення корпоративної мережі шляхом його тестування. Предметом дослідження виступають моделі та методології тестування, спрямовані на захист інформаційних ресурсів у корпоративному середовищі.

Наукова новизна одержаних результатів полягає в тому, що в магістерській роботі:

– створено математичну модель, яка деталізує ключові етапи кібератаки, дозволяючи аналізувати її механізми та розробляти практичні рекомендації для попередження подібних загроз

– удосконалено метод тестування безпеки програмного забезпечення за рахунок використання моделі діагностики системи управління мережевими ресурсами.

Практичне значення результатів полягає у можливості адаптувати процес тестування безпеки для захисту інформаційних систем, а також у впровадженні запропонованого методу в рамках гнучких підходів до розроблення програмного забезпечення, забезпечуючи підвищення стійкості мереж до кіберзагроз.

Публікації. За матеріалами кваліфікаційної роботи опубліковано 1 статтю у фаховому виданні та 1 теза доповіді на міжнародній науковій конференції.

1 ДОСЛІДЖЕННЯ ПІДХОДІВ ДО ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЯКЕ ФУНКЦІОНУЄ В КОРПОРАТИВНІЙ МЕРЕЖІ

1.1 Огляд методології для тестування безпеки мережі

В умовах сучасної цифрової трансформації захист корпоративних інформаційних систем набуває першочергового значення. Стрімке збільшення кількості та складності кібератак створює серйозні виклики для безпеки програмного забезпечення організацій різного профілю. Особливо гостро постає питання захисту критичної інфраструктури та систем обробки конфіденційних даних.

Проблема тестування безпеки програмного забезпечення корпоративних мереж є комплексною та потребує системного підходу до її вирішення. Статистичні дані свідчать про зростання кількості успішних кібератак, що призводять до значних фінансових втрат та репутаційних ризиків. Водночас наявні методології тестування безпеки не завжди відповідають сучасним вимогам та викликам.

Актуальність дослідження посилюється через відсутність уніфікованих стандартів та методик проведення тестування безпеки, що знижує ефективність захисних заходів. Додаткові складнощі створюють механізми захисту програмного забезпечення від аналізу, такі як обфускація, шифрування та динамічна модифікація коду [8].

Зворотний інжиніринг - це процес аналізу готової програми, спрямований на вивчення її структури, функцій та принципів роботи. Однак він не обов’язково передбачає доступ до вихідного коду. Найчастіше аналіз здійснюється через дослідження скомпільованого коду, поведінки програми в різних умовах або навіть через декомпіляцію [9].

У контексті тестування зворотний інжиніринг дозволяє тестувальникам виявити потенційні слабкі місця, недоліки чи вразливості програми. Це може включати перевірку на безпеку, оцінку ефективності або пошук прихованих

функцій. Така техніка часто використовується для забезпечення надійності та відповідності програми очікуваним стандартам.

Сучасна практика тестування інформаційної безпеки базується на кількох фундаментальних методологіях, кожна з яких має свої особливості та сфери застосування.

Penetration Testing (тестування на проникнення) представляє собою комплексний підхід до оцінки захищеності систем шляхом симуляції реальних атак. Ця методологія передбачає активне тестування захисних механізмів та пошук вразливостей, які можуть бути використані зловмисниками. Процес включає збір інформації про цільову систему, виявлення потенційних точок входу, спроби експлуатації знайдених вразливостей та документування результатів. Особлива цінність цього методу полягає в отриманні практичних доказів існування проблем безпеки [10].

Security Assessment (оцінка захищеності) фокусується на систематичному аналізі всіх компонентів системи безпеки. В рамках цієї методології проводиться детальний аналіз архітектури системи, конфігурацій безпеки, політик та процедур захисту інформації. Оцінюється відповідність впроваджених механізмів захисту встановленим вимогам та галузевим стандартам. Результатом стає комплексний звіт про стан системи безпеки з рекомендаціями щодо її вдосконалення [11].

Vulnerability Assessment (оцінка вразливостей) зосереджується на виявленні, класифікації та пріоритизації вразливостей в інформаційних системах. Ця методологія передбачає використання спеціалізованих інструментів сканування та аналізу, які допомагають ідентифікувати потенційні слабкі місця в системі захисту. Особлива увага приділяється оцінці критичності виявлених вразливостей та розробці рекомендацій щодо їх усунення.

Security Audit (аудит безпеки) представляє собою формалізований процес перевірки відповідності системи безпеки встановленим нормам та стандартам. В рамках аудиту проводиться документована оцінка всіх аспектів інформаційної безпеки, включаючи технічні засоби захисту, організаційні процедури та

людський фактор. Результати аудиту дозволяють оцінити загальний рівень захищеності організації та визначити напрямки вдосконалення системи безпеки.

Compliance Testing (тестування на відповідність) фокусується на перевірці дотримання нормативних вимог та галузевих стандартів безпеки. Ця методологія особливо важлива для організацій, які працюють у регульованих галузях та повинні відповідати специфічним вимогам до захисту інформації. Процес включає детальну перевірку всіх аспектів безпеки на відповідність встановленим нормам.

Red Team Assessment (оцінка червоної команди) представляє собою розширений варіант тестування на проникнення, який імітує дії реальних зловмисників в умовах максимально наближених до реальних. Ця методологія дозволяє оцінити не тільки технічні аспекти захисту, але й готовність організації до виявлення та реагування на складні, багатовекторні атаки.

Source Code Review (аналіз вихідного коду) фокусується на виявленні вразливостей на рівні програмного коду. Ця методологія передбачає детальний аналіз вихідного коду застосунків для виявлення потенційних проблем безпеки ще на етапі розробки. Особлива увага приділяється пошуку типових помилок програмування, які можуть призвести до вразливостей

Configuration Review (аналіз конфігурацій) зосереджується на перевірці налаштувань систем безпеки та мережевого обладнання. В рамках цієї методології проводиться детальний аналіз конфігураційних файлів, параметрів безпеки та системних налаштувань для виявлення потенційних проблем та невідповідностей встановленим вимогам [12-15].

Кожна з описаних методологій має свої переваги та обмеження, тому на практиці найбільш ефективним є комплексний підхід, який поєднує елементи різних методологій залежно від конкретних потреб та особливостей організації. Такий підхід забезпечує найбільш повну оцінку стану інформаційної безпеки та дозволяє розробити ефективні заходи щодо її вдосконалення.

Серед найбільш поширених методологій тестування на проникнення першочергове місце займає OSSTMM (Open Source Security Testing Methodology

Manual) [16]. Ця методологія забезпечує комплексний підхід до оцінки безпеки через визначення п'яти основних каналів тестування: фізичного, людського, телекомунікаційного, дротових мереж та бездротових комунікацій. OSSTMM відрізняється детальним описом процедур тестування та чіткими метриками для оцінки результатів.

OWASP Testing Guide представляє собою фреймворк, орієнтований переважно на тестування веб-додатків. Методологія охоплює різні аспекти безпеки, починаючи від конфігурації серверів та закінчуючи бізнес-логікою додатків. Особлива увага приділяється виявленню поширених вразливостей веб-додатків та методам їх експлуатації [17].

NIST SP 800-115 є стандартизованою методологією, розробленою Національним інститутом стандартів і технологій США. Документ надає детальні рекомендації щодо планування та проведення тестування безпеки інформаційних систем. Методологія акцентує увагу на важливості попереднього планування, оцінці ризиків та документуванні результатів [18].

PTES (Penetration Testing Execution Standard) пропонує структурований підхід до проведення тестування на проникнення через сім основних етапів: збір попередньої інформації, розвідка, моделювання загроз, аналіз вразливостей, експлуатація, пост-експлуатація та звітування. Кожен етап супроводжується детальними технічними рекомендаціями [19].

ISSAF (Information Systems Security Assessment Framework) надає комплексний підхід до оцінки безпеки інформаційних систем. Методологія розділяє процес тестування на фази планування, оцінки та звітування, з детальним описом процедур та інструментів для кожного етапу.

Методологія CEH (Certified Ethical Hacker) фокусується на практичних аспектах тестування безпеки. Вона охоплює широкий спектр технік та інструментів, використовуваних для виявлення та експлуатації вразливостей, включаючи методи соціальної інженерії.

CBEST представляє собою методологію, розроблену Bank of England для тестування критичних фінансових систем. Особливістю є використання розвідувальних даних про реальні загрози для формування сценаріїв тестування.

Методологія TIBER-EU, розроблена Європейським центральним банком, призначена для проведення контрольованих кібератак на критичні системи фінансового сектору. Вона передбачає використання актуальної інформації про загрози та техніки атак.

Практичне застосування цих методологій часто потребує їх адаптації під конкретні умови та вимоги організації. Важливим аспектом є врахування специфіки галузі, розміру організації, наявних ресурсів та регуляторних вимог. Найбільш ефективним вважається комбінований підхід, який поєднує елементи різних методологій для досягнення максимальної повноти тестування.

Кожна з описаних методологій має свої переваги та обмеження. OSSTMM та PTES надають найбільш повний технічний підхід, OWASP ідеально підходить для веб-додатків, а NIST SP 800-115 забезпечує відповідність державним стандартам. CBEST та TIBER-EU особливо ефективні для фінансового сектору.

Сучасні тенденції розвитку методологій тестування на проникнення спрямовані на інтеграцію з процесами безперервної розробки та впровадження (CI/CD), автоматизацію процесів тестування та адаптацію до нових видів загроз. Особлива увага приділяється розробці методів тестування хмарних інфраструктур та Інтернету речей.

Дослідження показують, що найбільшу ефективність демонструє комбінований підхід, який поєднує елементи різних методологій. Статистика свідчить, що організації, які використовують комплексне тестування, на 60% рідше стають жертвами успішних кібератак.

Значний вплив на ефективність має періодичність проведення тестування. Компанії, що впровадили регулярне тестування безпеки, демонструють зниження кількості інцидентів безпеки на 45% порівняно з тими, хто проводить тестування епізодично [20].

Сучасні методології тестування стикаються з численними викликами, серед яких складність аналізу захищеного програмного забезпечення, що часто ускладнює виявлення прихованих вразливостей. Зростання кількості нових видів загроз вимагає постійної адаптації підходів та інструментів тестування. Реалізація комплексного тестування є фінансово витратною, що створює додатковий тиск на бюджети компаній. Окрім цього, ринок відчуває дефіцит кваліфікованих спеціалістів, здатних працювати з новітніми технологіями та методами аналізу. Відсутність єдиних стандартів у сфері тестування також ускладнює впровадження ефективних рішень, адже кожна організація змушена розробляти власні підходи, що може впливати на якість і надійність процесу.

1.2 Класифікація існуючих методів тестування

Розглядаючи методи за обсягом наданої інформації, можна виділити три основні підходи. При тестуванні за методом Black Box тестувальник працює без попередньої інформації про систему, що максимально наближає умови до реальної атаки зловмисника. У випадку Grey Box фахівець отримує обмежену інформацію про систему, що дозволяє зробити тестування більш цілеспрямованим. White Box передбачає повний доступ до всієї інформації про систему, включаючи вихідний код та архітектурну документацію [21].

Щодо способу проведення тестування, існує кілька варіантів. Ручне тестування виконується фахівцем особисто та дозволяє виявити складні та нестандартні вразливості, які можуть бути пропущені автоматизованими інструментами. Автоматизоване тестування використовує спеціалізовані інструменти та скрипти для швидкого виявлення відомих вразливостей. Гібридне тестування поєднує переваги обох підходів, використовуючи як автоматизовані інструменти, так і експертизу фахівця.

За місцем проведення тестування поділяється на зовнішнє та внутрішнє. Зовнішнє тестування імітує атаку з боку віддаленого зловмисника та фокусується

на периметрі мережі. Внутрішнє тестування проводиться з середини мережі організації та дозволяє оцінити захищеність від внутрішніх загроз [22].

При виборі методу важливо враховувати декілька ключових факторів: цілі тестування, наявний бюджет, часові обмеження, критичність системи, що тестується, вимоги регуляторів та технічні можливості команди, яка буде проводити тестування. Правильний вибір методу значно впливає на ефективність всього процесу тестування та якість отриманих результатів (рис 1.1).

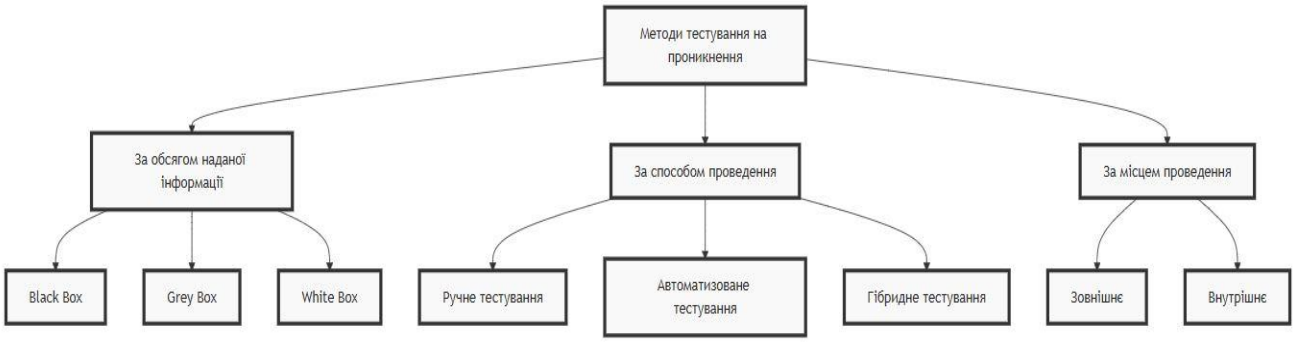


Рисунок 1.1 – Методи тестування на проникнення

Реверс інжиніринг є важливим технічним методом аналізу, який дозволяє зрозуміти принципи роботи досліджуваної системи через вивчення її внутрішньої структури. У контексті тестування на проникнення цей підхід використовується для детального дослідження програмного забезпечення та виявлення потенційних вразливостей.

При статичному аналізі тестувальник досліджує програмне забезпечення без його запуску. Це включає дизасемблювання виконуваних файлів для отримання асемблерного коду, декомпіляцію для відновлення вихідного коду високорівневою мовою програмування та аналіз бінарного коду для пошуку характерних патернів вразливостей. Такий підхід дозволяє виявити потенційні проблеми у логіці програми, небезпечні функції та вразливі місця в реалізації алгоритмів безпеки.

Динамічний аналіз передбачає дослідження програми під час її виконання. Використовуючи налагоджувачі, тестувальник може відстежувати поведінку

програми, аналізувати потік даних та контролювати виконання коду. Моніторинг поведінки програми дозволяє виявити аномалії у роботі та потенційні місця для експлуатації вразливостей. Аналіз мережевої активності допомагає зрозуміти, як програма взаємодіє з мережею та виявити небезпечні мережеві операції.

Особливу увагу при реверс інжинірингу приділяють аналізу механізмів захисту, таких як системи автентифікації, шифрування даних та перевірки цілісності. Розуміння їх реалізації допомагає виявити потенційні слабкості та способи їх обходу. Важливим аспектом є також аналіз обробки помилок та виняткових ситуацій, оскільки неправильна реалізація цих механізмів часто створює вразливості.

Після виявлення потенційних вразливостей проводиться їх валідація для підтвердження можливості експлуатації. Всі знайдені проблеми ретельно документуються з описом технічних деталей, потенційних ризиків та рекомендацій щодо усунення.

При проведенні реверс інжинірингу важливо дотримуватися правових аспектів та умов ліцензійних угод. Тестування має проводитися лише в межах узгодженого з замовником обсягу робіт та з дотриманням відповідних процедур безпеки (рис. 1.2).

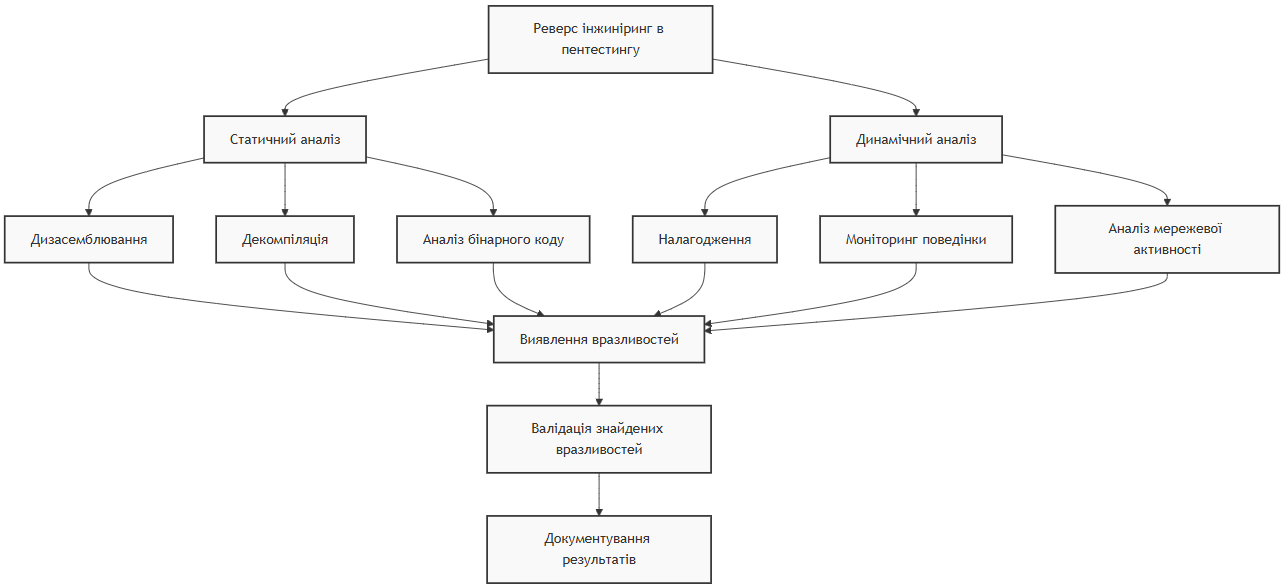


Рисунок 1.2 – Методи реверс інжинірингу

Розглянемо ключові інструменти та техніки, які застосовуються при реверс інжинірингу під час тестування на проникнення.

Дизасемблери є фундаментальними інструментами у реверс інжинірингу. IDA Pro вважається індустріальним стандартом завдяки потужним можливостям аналізу та підтримці широкого спектру процесорних архітектур. Ghidra, розроблена NSA, надає схожу функціональність та є безкоштовною альтернативою. Radare2 популярний серед досвідчених фахівців завдяки гнучкості та можливості автоматизації через скриптинг.

У сфері декомпіляції JD-GUI незамінний для аналізу Java-додатків, дозволяючи відновлювати вихідний код з байт-коду. DotPeek спеціалізується на .NET застосунках та надає зручний інтерфейс для навігації по коду. Hopper особливо ефективний при роботі з macOS та iOS додатками.

Серед налагоджувачів x64dbg є потужним інструментом для Windows-платформи, надаючи можливості трасування, встановлення точок зупину та аналізу пам'яті. GDB, будучи стандартом у Unix-подібних системах, підтримує віддалене налагодження та роботу з різними мовами програмування. WinDbg від Microsoft надає глибокий доступ до системних структур Windows.

Аналізатори бінарного коду як Binwalk допомагають досліджувати вбудовані системи та знаходити приховані дані у файлах. Binary Ninja пропонує потужну платформу для автоматизованого аналізу та візуалізації потоку виконання програми.

Для аналізу мережевої взаємодії Wireshark є незамінним інструментом, що дозволяє детально вивчати мережевий трафік. Burp Suite фокусується на веб-додатках та надає комплексні можливості для тестування веб-безпеки [23].

Розглядаючи техніки аналізу, варто відзначити такі основні підходи як патерн-матчинг для пошуку відомих сигнатур у коді, аналіз потоку даних для відстеження обробки критичної інформації, та символьне виконання для виявлення потенційних шляхів експлуатації. Важливими є також техніки обходу захисних механізмів, таких як обфускація, антидебаг та антидизасемблювання.

При роботі з зашифрованими даними застосовуються методи криптоаналізу

та пошуку ключів у пам'яті. Для розуміння складних алгоритмів використовується техніка трасування виконання з візуалізацією графу викликів функцій.

Впровадження ризик-орієнтованого підходу до планування тестування дозволяє оптимізувати використання ресурсів, зосередитись на критичних загрозах та підвищити ефективність виявлення вразливостей. Створення стандартизованих методик проведення тестів забезпечує не лише послідовність і повноту перевірок, а й сприяє уніфікації процесів між командами, що підвищує узгодженість результатів. Розвиток систем автоматизованого моніторингу безпеки сприяє своєчасному виявленню потенційних загроз і дозволяє оперативно реагувати на нові виклики, знижуючи ймовірність критичних збоїв. Інтеграція цих підходів у загальну стратегію тестування допомагає створити більш стійке до ризиків середовище та забезпечує надійність програмних рішень.

1.3 Особливості двійкових файлів, та інструментарію для їх аналізу

Двійкові файли - унікальні цифрові контейнери, що репрезентують інформаційні масиви через послідовність бінарних значень. На відміну від текстових документів, вони становлять особливий клас носіїв даних, призначених для безпосереднього сприйняття комп'ютерними системами [24].

Внутрішня архітектура таких файлів базується на принципі послідовного розміщення бітів, де кожен елемент може набувати значення нуль або одиниця. Це дозволяє максимально ущільнювати інформаційний простір, забезпечуючи високу щільність збереження різноманітних даних.

Особливістю двійкових файлів є їхня здатність зберігати надскладні структуровані масиви: від мультимедійного контенту до виконуваних інструкцій операційних систем. Графічні зображення, музичні композиції, відеоролики та службові програмні компоненти - все це може бути представлено у бінарному форматі.

Процес роботи з такими файлами вимагає спеціалізованих підходів. Програмісти використовують спеціальні бібліотеки та методики для коректного зчитування, інтерпретації та модифікації бінарних даних. Кожен байт може нести унікальне смислове навантаження, що потребує точного алгоритму декодування.

Важливою характеристикою є незалежність від текстових кодувань. На відміну від символьних послідовностей, бінарні дані не прив'язані до конкретної мови чи набору символів, що робить їх універсальним способом збереження інформації.

Технічні обмеження таких файлів пов'язані з потребою спеціального апаратного та програмного забезпечення для коректної інтерпретації. Людина не здатна безпосередньо прочитати вміст без застосування додаткових перетворень та утиліт.

Сучасні операційні системи та прикладні середовища активно використовують бінарні формати для оптимізації процесів зберігання, передачі та обробки інформаційних масивів. Кожен такий файл являє собою складну екосистему цифрових інструкцій та даних.

У світі реверс-інжинірингу дослідники використовують потужні інструментальні комплекси для глибокого аналізу бінарних файлів. Серед провідних засобів виділяються спеціалізовані програмні продукти, що дозволяють декомпозувати складні виконувані модулі та системні компоненти (рис. 1.3).

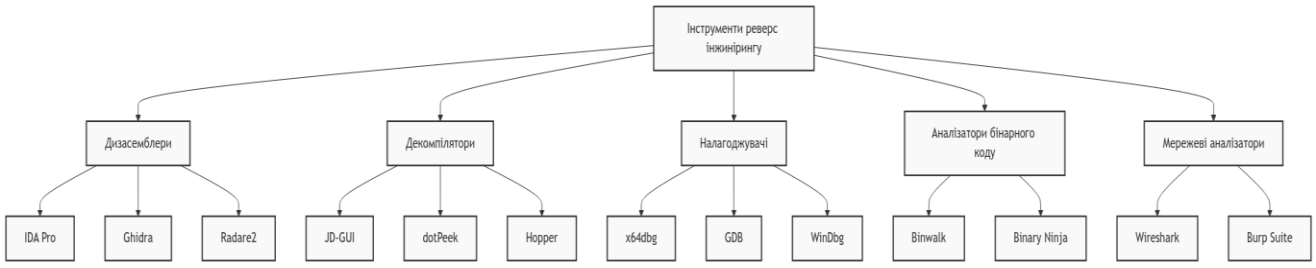


Рисунок 1.3 – Інструменти татехніки реверс інжинірингу

Одним з найпотужніших інструментів є IDA Pro - професійне середовище для дизасемблювання та статичного аналізу машинного коду. Цей інструмент надає унікальні можливості структурного декодування бінарних послідовностей, дозволяючи фахівцям досліджувати внутрішню логіку програмних додатків [25].

Не менш важливим є Ghidra - потужний фреймворк, розроблений спеціалістами Агентства nacional безпеки США. Інструмент містить розширений функціонал для декомпіляції, графічної візуалізації алгоритмів та автоматизованого аналізу криптографічних конструкцій.

Hexeditors представляють окрему категорію інструментів, що дозволяють здійснювати низькорівневе маніпулювання бінарними даними. Binary Ninja та Hopper слугують додатковими платформами для глибинного вивчення структури виконуваних модулів, забезпечуючи високоточну інтерпретацію машинних інструкцій.

Динамічний аналіз реалізується через спеціалізовані налагоджувачі на кшталт OllyDbg, x64dbg та WinDbg. Ці інструменти дозволяють здійснювати покрокове виконання коду, аналізувати стан процесора, досліджувати механізми взаємодії між програмними модулями.

Важливу нішу займають спеціалізовані фреймворки для поведінкового аналізу, такі як Radare2 та Binary Analysis Platform. Вони надають комплексні механізми автоматизованого дослідження потенційно шкідливих компонентів, виявлення прихованих алгоритмів та небезпечних включень.

Суттєвим доповненням слугують мережеві аналізатори та системи віртуальної емуляції. За допомогою таких інструментів фахівці можуть моделювати поведінку бінарних об'єктів у контрольованому середовищі, не ризикуючи безпекою реальних інформаційних систем.

Реверс-інжиніринг вимагає глибоких знань архітектури комп'ютерних систем, розуміння низькорівневих механізмів взаємодії апаратних та програмних компонентів. Сучасні інструментальні комплекси постійно еволюціонують, надаючи дослідникам дедалі витонченіші методи декомпозиції та аналізу бінарних структур.

Аналіз двійкових файлів у корпоративній мережі вимагає комплексного підходу та дотримання певних безпекових протоколів [26].

Підготовчий етап включає створення ізольованого середовища для безпечного аналізу. Використовують віртуальні машини або спеціальні пісочниці, які повністю відокремлені від основної корпоративної інфраструктури. Це дозволяє мінімізувати ризики потенційного інфікування або несанкціонованого впливу на робочі системи.

Первинна діагностика передбачає збирання метаданих про файл: його походження, час створення, цифрові сертифікати, хеш-суми. Фахівці використовують спеціалізовані утиліти для перевірки репутації файлу в глобальних базах даних відомих загроз. Важливо перевірити підпис розробника та проаналізувати можливі відхилення від стандартної поведінки.

Статичний аналіз здійснюється без безпосереднього запуску файлу. Застосовують дизасемблери та декомпілятори для вивчення внутрішньої структури та логіки виконання. Професійні інструменти дозволяють виявити приховані алгоритми, потенційні закладки або підозрілі послідовності машинного коду.

Динамічний аналіз передбачає контрольований запуск файлу в ізольованому середовищі. Спеціалісти моніторять усі системні виклики, мережеву активність, зміни в реєстрі та файловій системі. Особливу увагу приділяють виявленню нестандартної поведінки, спроб несанкціонованого доступу або втручання в системні процеси.

Мережевий аналіз включає детальне дослідження мережевої активності файлу. Використовують спеціалізовані снíфери та аналізатори трафіку для виявлення підозрілих комунікацій, спроб віддаленого підключення або передачі даних. Критично важливо ідентифікувати всі зовнішні точки взаємодії.

Криптографічний аналіз дозволяє виявити приховані механізми шифрування, алгоритми обфускації та потенційні канали витоку інформації. Фахівці досліджують внутрішні бібліотеки, механізми генерації ключів та алгоритми криптографічного захисту.

Результати дослідження оформлюють у детальний звіт, який містить повний опис виявлених особливостей, потенційних ризиків та рекомендацій. Важливо забезпечити конфіденційність та строго документувати кожен етап дослідження.

Етичні та юридичні аспекти такого аналізу вимагають обов'язкового дотримання внутрішніх корпоративних політик, законодавчих норм та погодження з відповідальними підрозділами безпеки.

Успішний аналіз двійкових файлів потребує поєднання технічних навичок, глибокого розуміння архітектури комп'ютерних систем та постійного оновлення методологічного інструментарію.

1.4 Постановка задачі

Проведений аналіз довів, що існує широкий спектр підходів до розробки та використання тестування безпеки ПЗ для корпоративних мереж. Ці підходи можуть суттєво відрізнятися залежно від технологій впровадження, вартості, а також інших тактико-технічних характеристик, таких як ефективність, точність і масштабованість методів. Вибір конкретного методу тестування залежить від особливостей мережі, її компонентів і вимог безпеки, визначених організацією.

Метою розробки методу тестування безпеки ПЗ для корпоративних мереж є створення, вдосконалення та вибір моделей і методів, які забезпечують найвищий рівень безпеки. Важливою частиною цього процесу є адаптація методів тестування до конкретних умов і потреб мережі, що дає змогу більш точно оцінити потенційні вразливості та загрози.

Один із ключових етапів полягає в розробці математичних моделей для тестування на проникнення. Це дозволяє автоматизувати виявлення вразливостей, зменшуючи вплив людського фактора і підвищуючи точність результатів. Математичні моделі допомагають оцінити ймовірність виникнення певних загроз, враховуючи різноманітні сценарії атак.

Зокрема, важливо виділити кілька завдань, які необхідно вирішити при розробці:

– проведено оцінку ймовірності успішного проникнення в систему, враховуючи рівень захисту та існуючі механізми безпеки;

– змоделювано сценарії атак, що дає можливість оцінити ефективність існуючих заходів безпеки в умовах реальних загроз;

– обрано метод аналізу коду для виділення алгоритму;

– змодельовано початковий етап кібератаки до ресурсів корпоративної мережі;

– перевірено ефективності розроблених моделей.

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

2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСУ ТЕСТУВАННЯ КОРПОРАТИВНОЇ МЕРЕЖІ

2.1 Комплексна оцінка ризиків для безпеки корпоративної мережі

Ризик у сфері інформаційної безпеки визначається як потенційний негативний вплив, що виникає внаслідок реалізації вразливості. Він охоплює дві основні складові: ймовірність настання небажаної події та масштаб її наслідків. Управління ризиками, таким чином, є систематичним процесом, що включає ідентифікацію, оцінку та мінімізацію ризиків до прийнятного рівня [27].

Для забезпечення ефективного керування ризиками організаціям доцільно використовувати стандартизовані підходи. Однією з найкращих методологій у цій сфері є OWASP Risk Assessment Methodology (OWASP-RAM) [28] На рисунку 2.1 наведено приклад оцінки можливості використання шкідливого програмного забезпечення (ШПЗ).

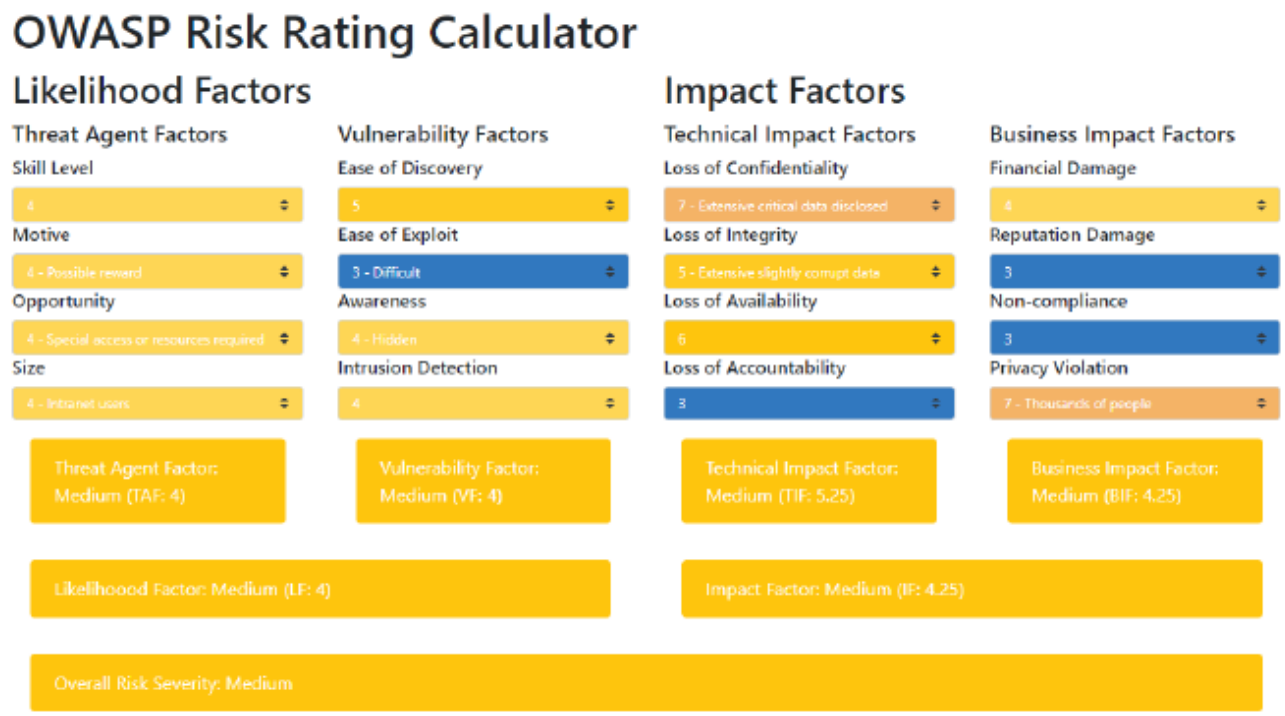


Рисунок 2.1 – OWASP Risk Assessment Methodology (ШПЗ)

OWASP-RAM є ефективним інструментом для оцінки ризиків, пов'язаних з використанням шкідливого програмного забезпечення. Ця методологія допомагає

системно аналізувати вразливості та загрози з урахуванням ймовірності їх реалізації та можливих наслідків, що дозволяє розробити обґрунтовані заходи для мінімізації ризиків.

Перший крок - ідентифікація загроз. У контексті шкідливого програмного забезпечення це може включати визначення типів шкідливих програм, які становлять найбільший ризик для організації: віруси, трояни, шпигунські програми, програми-вимагачі або руткіти. Також беруться до уваги можливі способи впровадження шкідливого коду, такі як фішингові атаки, використання незахищених протоколів, зараження через підключені пристрої або соціальна інженерія.

Другий етап - оцінка вразливості. Це аналіз систем, які можуть бути атаковані шкідливим програмним забезпеченням, наприклад, вразливості програмного забезпечення, відсутність належного антивірусного захисту, ненадійні механізми оновлення або недостатня обізнаність співробітників. Методологія OWASP допомагає виявити слабкі місця, які сприяють успішному впровадженню шкідливого програмного забезпечення.

Наступним кроком є аналіз ризиків, який враховує два основні фактори: ймовірність проникнення шкідливого програмного забезпечення та його вплив на організацію. Для цього використовується матриця ризиків OWASP, яка допомагає класифікувати загрози за ступенем серйозності, беручи до уваги такі аспекти, як рівень автоматизації атак, частота цілеспрямованих спроб, ймовірність виявлення та можливий вплив на конфіденційність, цілісність і доступність даних.

Завершальним етапом є розробка та впровадження заходів зі зниження ризиків. Використовуючи результати оцінки, організація може визначити пріоритетність заходів безпеки. Наприклад, впровадити проактивний моніторинг мережі для виявлення підозрілої активності, оновити програмне забезпечення до найновіших версій, навчити співробітників протидіяти фішинговим атакам або налаштувати політики доступу, щоб обмежити поширення шкідливого програмного забезпечення в разі порушення.

В результаті методологія надає структуровану рамку для визначення, класифікації та оцінки ризиків, з акцентом на аналіз імовірності та потенційних наслідків реалізації загроз. Методологія також дозволяє інтегрувати отримані дані в загальну стратегію керування ризиками, сприяючи більш точному визначенню пріоритетів та оптимізації використання ресурсів.

Основною метою керування ризиками є створення умов, за яких організація може досягати своїх стратегічних цілей, забезпечуючи належний рівень захисту IT-систем, що зберігають, обробляють або передають важливу організаційну інформацію. Процес керування ризиками дозволяє керівництву приймати обґрунтовані рішення щодо витрат на заходи безпеки, оптимізуючи IT-бюджет і забезпечуючи баланс між вартістю захисту та рівнем залишкового ризику.

Також OWASP-RAM надає структурований підхід до оцінки ризиків, що виникають через невиправлені вразливості у програмному забезпеченні чи інфраструктурі. Ця методологія дозволяє оцінити ймовірність використання таких вразливостей зловмисниками, а також масштаб можливих наслідків, що допомагає визначити пріоритети для усунення або мінімізації ризиків (рис.2.2).

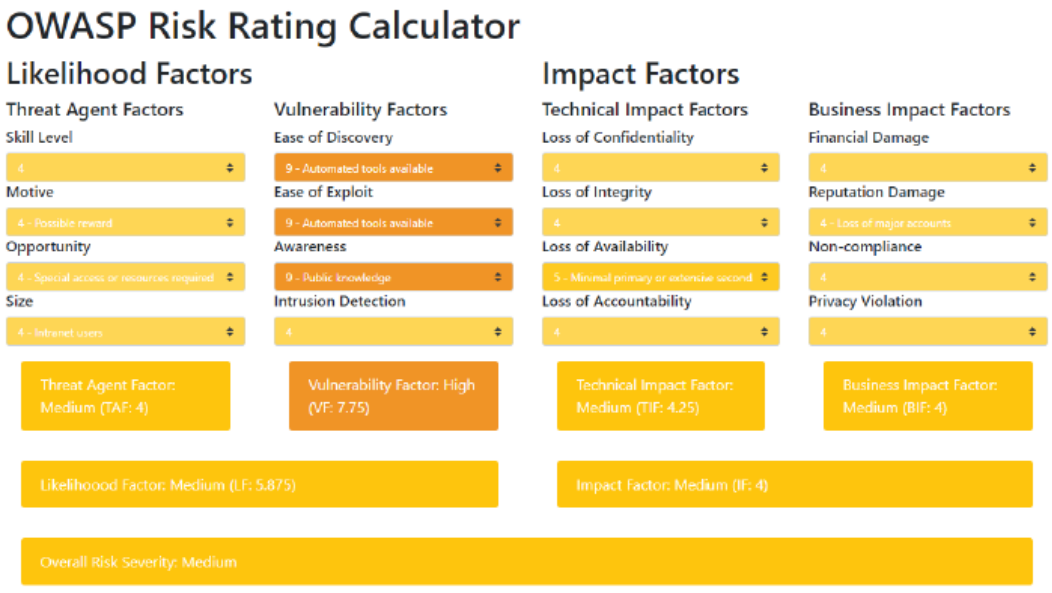


Рисунок 2.2 – Оцінки ризиків, які виникають через невиправлені вразливості у програмному забезпеченні чи інфраструктурі

Першим кроком є виявлення невиправлених вразливостей. Це передбачає аналіз існуючих систем за допомогою автоматизованих сканерів вразливостей, ручного тестування або доступу до баз даних відомих вразливостей, таких як CVE (Common Vulnerabilities and Exposures) або NVD (National Vulnerability Database) [29-30]. Особлива увага приділяється критичним компонентам систем, таким як програми із застарілими бібліотеками, ненадійними конфігураціями або відсутніми патчами.

Наступним кроком є оцінка ймовірності використання вразливостей. Методологія OWASP враховує такі фактори, як наявність експлойтів у відкритих джерелах, складність використання вразливості, рівень технічної експертизи, необхідний для атаки, та наявність активних загроз. Наприклад, вразливості з опублікованими експлоітами, що підтверджують концепцію, мають більшу ймовірність бути використаними.

Наступним кроком є аналіз потенційного впливу. Він оцінює, як використання вразливості може вплинути на конфіденційність, цілісність та доступність системи. Наприклад, невиправлена вразливість у веб-додатку може дозволити SQL-ін'єкцію, що призведе до крадіжки конфіденційних даних. Уразливості в мережевих протоколах можуть сприяти DoS-атакам або несанкціонованому доступу до критично важливих систем.

Оцінюючи ризики, OWASP-RAM використовує матрицю ризиків, яка допомагає класифікувати вразливості за ступенем їхньої критичності, визначаючи найбільш небезпечні та нагальні для усунення.

Заключним етапом є розробка стратегій зменшення ризиків. Залежно від оцінки, організація може обрати відповідні заходи, такі як:

– виправлення (підтримка системи в актуальному стані за допомогою регулярних оновлень і патчів);

– впровадження захисних заходів, таких як брандмауери вебдодатків (WAF), для блокування використання вразливостей;

– ізоляція вразливих компонентів (наприклад, сегментація мережі);

– розробка процедур реагування на інциденти для швидкого усунення

наслідків успішних атак.

Інтеграція керування ризиками в загальну систему безпеки забезпечує зниження вразливостей, підвищення стійкості організації до зовнішніх і внутрішніх загроз, а також сприяє дотриманню нормативних вимог. Це дозволяє не лише мінімізувати втрати від потенційних інцидентів, а й зміцнити довіру клієнтів і партнерів до безпеки даних організації [31].

OWASP-RAM також сприяє розвитку культури проактивного керування ризиками. Він дозволяє організаціям не тільки зосередитися на виправленні вразливостей, але й будувати стратегії для запобігання подібним проблемам у майбутньому, інтегруючи підходи до безпеки на всіх етапах життєвого циклу програмного забезпечення.

Оцінка ризиків, пов'язаних з невідомими вразливостями в програмному забезпеченні або прикладних програмних інтерфейсах (API), є одним з найскладніших завдань у сфері інформаційної безпеки [32]. Методологія оцінки ризиків OWASP надає інструменти для системного підходу до аналізу таких ризиків, навіть коли точні деталі вразливостей залишаються невідомими (рис.2.3).

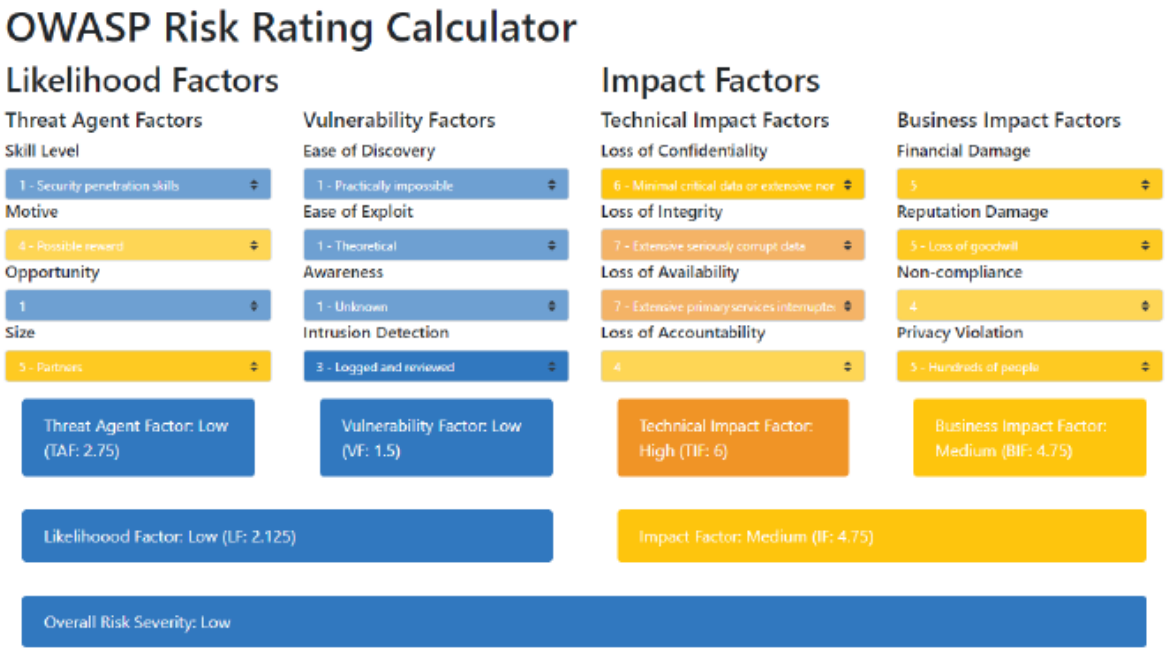


Рисунок 2.3 – Оцінки ризиків, які пов'язані з невідомими вразливостями в програмному забезпеченні чи API

Процес починається з визначення потенційних зон ризику. Це включає аналіз складних програмних компонентів або API, які часто є джерелом прихованих помилок. До таких областей належать частини систем, які працюють з критично важливими даними, використовують складні алгоритми, взаємодіють з іншими системами або залежать від сторонніх бібліотек. Також беруться до уваги області, де складність коду ускладнює його повноцінне тестування.

На другому етапі оцінюється ймовірність невідомих помилок. Методологія OWASP передбачає наступні підходи:

– аналіз попередніх інцидентів (наприклад, чи є в організації історія виявлення подібних помилок);

– оцінка складності коду (більш складні системи зазвичай мають більше потенційних вразливостей);

– використання статистичних моделей, які враховують типові патерни помилок у певних технологіях.

Далі оцінюється вплив невідомих багів. Хоча конкретні сценарії експлуатації можуть бути невідомими, OWASP-RAM пропонує підходи до моделювання можливих впливів на основі характеру даних, що обробляються, типу API (наприклад, відкритий чи закритий), ролі системи в інфраструктурі та здатності взаємодіяти з критичними компонентами. Вплив оцінюється з точки зору потенційної шкоди для конфіденційності, цілісності та доступності.

Методологія також підтримує аналіз ймовірності виявлення. Це дозволяє зрозуміти, наскільки ймовірно, що невідома помилка може бути знайдена зловмисниками раніше, ніж розробниками або системами безпеки. Якщо інтерфейс або система відкриті для зовнішнього доступу, ймовірність такого виявлення зростає.

У сучасних умовах високий рівень захисту мережі, забезпечений передовими засобами безпеки, може створити значні труднощі для пентестерів, які намагаються обійти захисні механізми. У таких випадках завдання зводиться до пошуку альтернативних шляхів доступу до даних, одним із яких є фізичний доступ до елементів мережі. Цей підхід передбачає, що спеціаліст з безпеки

отримує можливість безпосередньо взаємодіяти з обладнанням, наприклад, робочими станціями чи мережевими пристроями, потрапивши до приміщення.

Фізична атака є не лише тестом на стійкість мережевих систем, а й засобом перевірки ефективності організаційних заходів безпеки, таких як контроль доступу до приміщень, політики поводження з носіями даних та загальна інфраструктура фізичного захисту. Її проведення доречно лише за умов, коли в організації вже існує відпрацьована політика безпеки, впроваджені необхідні процедури та проведене навчання персоналу [33].

Соціальна інженерія є ключовим елементом такого тестування. Застосовуючи її методи, пентестери оцінюють рівень підготовки співробітників до можливих маніпуляцій, таких як видавання себе за технічний персонал, використання підроблених перепусток або отримання доступу через людський фактор. Це дозволяє визначити слабкі місця у системі фізичного захисту, які зловмисники можуть використати для доступу до критичних даних [34].

Перевагою фізичної атаки є її здатність виявляти загрози, що зазвичай залишаються поза увагою суто технічного тестування. Наприклад, недостатньо захищені роз'єми пристроїв, доступні USB-порти, незахищені комутаційні шафи або відсутність відеоспостереження в ключових зонах.

Важливим аспектом таких перевірок є відповідність правовим і етичним нормам. Проведення фізичних атак має бути санкціоноване керівництвом організації, а також враховувати можливі ризики для співробітників та обладнання.

Таким чином, тестування із використанням фізичного доступу є важливим доповненням до стандартних методів пентестингу, дозволяючи оцінити не лише технічні, а й організаційні заходи безпеки. Це підвищує загальну стійкість системи до потенційних загроз і сприяє розвитку комплексного підходу до захисту інформаційних активів.

Взявши до уваги результати аналізу, загальну схему процесу тестування зображено на рисунку 2.4.

Рисунок 2.4 – Схема процесу тестування

Тестування безпеки програмного забезпечення є важливим компонентом виявлення та усунення вразливостей, які можуть бути використані зловмисниками для проникнення в систему або пошкодження її компонентів. Воно складається з декількох основних етапів: виявлення вразливостей, тестування на проникнення та тестування на експлуатацію. Кожен з цих етапів включає різні методи, які забезпечують комплексний підхід до безпеки програмного забезпечення.

Одним з методів тестування є аналіз архітектури та дизайну програмного забезпечення. Цей етап проводиться на ранніх стадіях розробки і передбачає перевірку архітектурних рішень і дизайну програми на відповідність основним принципам безпеки. Зокрема, цей аналіз допомагає виявити потенційні проблеми, такі як надмірна поверхня атаки або використання ненадійних алгоритмів шифрування. Раннє виявлення цих проблем може зменшити витрати на виправлення вразливостей на більш пізньому етапі.

Іншим важливим методом є побудова моделі загрози. Цей підхід передбачає створення теоретичної моделі, яка допомагає зрозуміти можливі сценарії атак на систему. Оцінка потенційних загроз дозволяє передбачити, які частини програмного забезпечення можуть стати мішенню для зловмисників, а також оцінити наслідки цих атак для організації. Побудова моделі загроз зазвичай передбачає вивчення найбільш вразливих компонентів системи та оцінку

ймовірності їхньої експлуатації.

Пентестування допомагає виявити слабкі місця в системі, які можуть бути використані для проникнення в систему. Такий підхід дозволяє оцінити ефективність існуючих інструментів безпеки і виправити знайдені недоліки.

Тестування на основі ризиків дозволяє зосередитися на найбільш критичних компонентах системи, оцінюючи ймовірність і вплив вразливостей. Цей метод допомагає ефективно розподіляти ресурси тестування, знижуючи ймовірність серйозних інцидентів безпеки.

Нечітке тестування - це техніка, яка передбачає подачу випадкових або некоректних даних у програмне забезпечення для виявлення помилок. Цей метод ефективний для виявлення невідомих вразливостей, оскільки він може імітувати багато непередбачуваних сценаріїв, які не були враховані під час звичайного тестування.

Перевірка процесу розгортання на відповідність вимогам також є важливою частиною тестування безпеки програмного забезпечення. Цей метод передбачає оцінку того, як система розгортається в реальному середовищі і чи відповідає процес конфігурації вимогам безпеки. Сюди входить перевірка налаштувань доступу, конфігурації сервера та використання криптографії для забезпечення захисту даних під час обробки та зберігання [35].

Перевірка архітектури та дизайну програмного забезпечення є важливою складовою аналізу безпеки, оскільки вона дозволяє оцінити відповідність системи вимогам безпеки ще на етапі її проектування. Як зазначено в працях вчених, цей аналіз має охоплювати кілька ключових аспектів, таких як розгортання системи, інфраструктура, а також загальна архітектура додатків і їх дизайн.

Важливим є те, що розгляд архітектури програмного забезпечення повинен проводитися в контексті інфраструктури та сценаріїв її розгортання. Це дозволяє оцінити, як саме система буде функціонувати в реальному середовищі, що включає не лише технологічні, а й організаційні аспекти. Врахування специфіки інфраструктури дає змогу виявити потенційні слабкі місця в системі, пов'язані з її інтеграцією та взаємодією з іншими компонентами.

Щодо дизайну програмного забезпечення, він має бути оцінений з урахуванням усіх категорій вразливостей, визначених на попередніх етапах роботи над проєктом. Це означає, що розробники та фахівці з безпеки повинні враховувати виявлені вразливості при проектуванні кожного компонента системи. Такі вразливості можуть бути технічними або організаційними, і вони потребують комплексного підходу до їх усунення ще на етапі розробки.

Поетапний компонентний аналіз програмного забезпечення передбачає глибоке дослідження кожного з ключових елементів системи на предмет забезпечення безпеки [32-34]. Це включає в себе перевірку механізмів аутентифікації, авторизації, шифрування даних та інших елементів, які можуть бути вразливими до різних видів атак.

Для кожного компонента необхідно розглядати механізми захисту, які забезпечують не тільки правильну роботу, а й захист від можливих атак (рис.2.5).

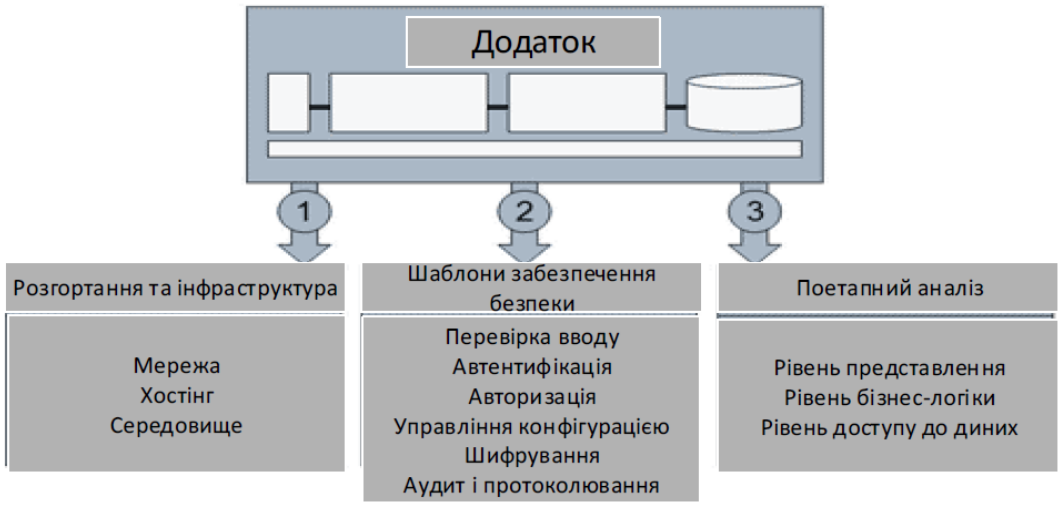


Рисунок 2.5 – Особливості перевірки архітектури ПЗ

Отже, перевірка дизайну програмного забезпечення є необхідним етапом для забезпечення надійного захисту від потенційних загроз і забезпечення його стійкості до атак. Цей підхід дозволяє на ранніх етапах виявляти можливі проблеми, що дає змогу їх усунути до розгортання системи в реальному середовищі [35].

Як видно з наведеного рисунку, цей метод аналізу розглядає основні питання, що стосуються розгортання додатків у складі корпоративних мереж. Проте цей метод не дозволяє оцінити більш детально аспекти, які пов'язані з методиками та алгоритмами розробки ПЗ. Оцінка архітектури та дизайну зосереджена переважно на загальних аспектах інфраструктури і взаємодії компонентів системи, тоді як глибший аналіз процесів розробки та впровадження програмного забезпечення потребує інших підходів, таких як застосування моделей ризиків.

Побудова моделей ризиків є важливою складовою передпроектної розробки, оскільки він дає змогу не тільки прогнозувати можливі загрози, а й описати вимоги до забезпечення безпеки. Це дозволяє системно підходити до виявлення вразливостей на ранніх етапах проекту. Моделі загроз допомагають не лише описати потенційні загрози та вразливості, але й розробити ефективні стратегії для їх запобігання.

Отже процес моделювання загроз має п'ять основних етапів (рис. 2.7).

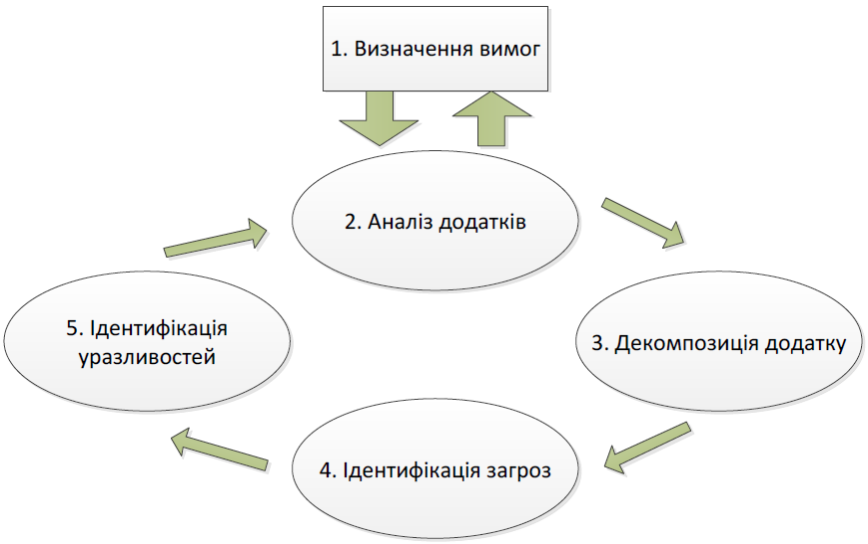


Рисунок 2.6 – Підхід до моделювання загроз

Розглянуті методи взаємно доповнюють один одного, забезпечуючи більш комплексний підхід до керування безпекою програмного забезпечення.

2.2 Моделювання процесу початку кібератаки

Оцінка безпеки мережі є ключовим компонентом забезпечення захисту інформаційних систем. Цей процес імітує можливі дії зловмисників, спрямовані на компрометацію систем з мережі, і дозволяє оцінити стійкість інфраструктури до кібератак. Методика тестування включає моделювання реальних загроз, таких як атаки троянами, DDoS-атаки чи інші види вторгнень, що можуть спричинити втрату, маніпуляцію або знищення даних та обладнання [36].

Результатом оцінки є детальний звіт, який містить виявлені вразливості, оцінку їхнього впливу та рекомендації щодо їх усунення. Зокрема, тест дозволяє ідентифікувати слабкі місця у мережевій безпеці, наприклад, недостатній захист від несанкціонованого доступу, ненадійні механізми автентифікації, некоректно налаштовані брандмауери або застаріле програмне забезпечення.

Оцінка також сприяє підвищенню загальної безпеки шляхом впровадження проактивних заходів. До них належать розробка оновлених політик захисту, налаштування моніторингових систем для виявлення аномалій та регулярне тестування на стійкість до сучасних загроз. Таким чином, оцінка безпеки мережі не лише виявляє наявні проблеми, але й формує основу для створення довгострокової стратегії захисту інформаційних активів.

Відповідно до принципів системного підходу до захисту інформації, сукупність взаємопов'язаних компонентів, що працюють для забезпечення безпеки функціонування системи, утворює структуру захисту даних. Ці компоненти включають математичні, технічні та програмні рішення, а також людські ресурси.

Проблеми при розбудові системи захисту інформації можуть виникати не лише через складність її структури, але й через вплив зовнішніх факторів або нестандартні дії зловмисників. Оскільки хакери все частіше звертають увагу на електронні інформаційні ресурси, спостерігається розширення різноманітних тактик атак, що відображає збільшення варіативності поведінки кіберзлочинців. Зростаючий інтерес до нових можливостей для кібератак стимулює еволюцію

стратегій, використовуваних хакерами, що, в свою чергу, робить захист інформації більш складним і багатогранним завданням.

Для реалізації кібератаки хакер дійсно повинен змоделювати подію, враховуючи численні фактори, що можуть вплинути на успіх атаки. Існує ряд математичних моделей, які використовуються для аналізу та моделювання кіберзагроз, допомагаючи як зловмисникам, так і фахівцям з безпеки вивчати й оцінювати ймовірність успішних атак.

Стохастичні моделі є важливими для оцінки ризику і ймовірності успішної кібератаки. Ці моделі дозволяють описати атаки як процеси з випадковими результатами, де ймовірність того, що певна вразливість буде використана або що система буде скомпрометована, залежить від ряду змінних факторів. Для аналізу ефективності атак може бути побудована модель, яка враховує ймовірність проникнення в систему на основі типів атак, захисту та вразливостей [36].

Моделі атак з множинними етапами описують кібератаки як серію етапів, кожен з яких має свою ймовірність успіху або невдачі. Це дозволяє зловмисникам (або захисникам) прораховувати ймовірність того, що атака на одному етапі успішно пройде і буде рухатися далі. Це може включати етапи розвідки, проникнення, ескалацію прав, використання шкідливого програмного забезпечення та отримання доступу до критичних даних.

Ігрові теорії широко використовуються для моделювання стратегічних ситуацій, де учасники мають різні інтереси та стратегії. У контексті кібератак, це може бути застосовано для моделювання протиборчих сторін: хакерів та захисників. У такій моделі можна аналізувати стратегії обох сторін, визначаючи оптимальні стратегії для зловмисників, які хочуть проникнути в систему, та для захисників, які намагаються запобігти атакі [37].

Моделі на основі оцінки вразливості, такі як CVSS (Common Vulnerability Scoring System), дозволяють оцінювати рівень ризику, пов'язаний з конкретною вразливістю. Зловмисники можуть використовувати ці моделі для визначення найбільш ефективних вразливостей для атак, враховуючи фактори, як серйозність уразливості, її експлуатаційні можливості та потенційний вплив на систему [38].

Моделі атак на основі мережевої топології моделюють мережу як набір вузлів і зв'язків між ними, дозволяючи визначити найслабші точки в мережевій архітектурі. Хакери використовують такі моделі для оцінки того, як краще атакувати мережу, враховуючи потенційні точки входу, шляхи для переміщення по мережі та способи ескалації прав.

Моделі аналізу динаміки інформаційних систем використовують системи диференціальних рівнянь або інших методів для моделювання поведінки інформаційних систем під час атак. Вони дозволяють зрозуміти, як зміни в системі (наприклад, уразливості, атаки або зміни в поведінці користувачів) можуть вплинути на її загальну безпеку.

Враховуючи, що значна частина успішних кібератак здійснюється через маніпуляції з користувачами, соціальна інженерія та моделі, засновані на психології людей, також є важливими інструментами [39]. Моделі соціальної інженерії намагаються передбачити ймовірність того, як користувачі можуть бути обмануті або маніпульовані для надання доступу до конфіденційної інформації або виконання шкідливих дій.

Всі ці моделі мають своє застосування не тільки для розробки ефективних стратегій атак, а й для створення більш захищених систем. Вони допомагають фахівцям з кібербезпеки передбачати можливі сценарії атак та оптимізувати заходи безпеки для зменшення ймовірності успішної реалізації кіберзагроз.

Аналіз наявних математичних моделей для опису кібератак дозволяє дійти висновку, що використання стохастичних підходів є найбільш ефективним, оскільки вони враховують випадковість і невизначеність подій. Особливий інтерес представляє метод графічного відображення подій, відомий як GERT-мережі [40-42]. Цей метод дає змогу моделювати ймовірності, залежності між етапами, а також визначати можливі сценарії розвитку атак.

Даний розділ присвячений розробці математичної моделі, що базується на GERT-мережах, для аналізу початкового етапу генерації коду, який використовується в кібератаках. У фокусі - взаємодія між цим процесом і системою керування ресурсами корпоративної мережі. Така модель допомагає

детально описати початкові умови атаки, включаючи характеристики вразливостей, доступність елементів мережі й імовірнісний розподіл успішного виконання кожного з кроків.

Особливу увагу в моделі приділено опису вузлів і зв’язків між ними, які відображають різні етапи атакувальних дій, а також визначенню випадкових змін у параметрах мережі. Це дає змогу моделювати варіативність поведінки мережі й оцінювати ефективність протидії на різних рівнях.

Створення такої моделі стане основою для прогнозування потенційних загроз, оцінки ризиків і підвищення ефективності заходів, спрямованих на захист інформаційної інфраструктури організації.

Аналіз математичних моделей, призначених для опису дій, пов’язаних із кібератаками, свідчить про доцільність застосування підходів, що враховують випадкові фактори й невизначеність. Серед таких методів найбільш перспективним є підхід, що дозволяє відображати події за допомогою графічної оцінки з використанням GERT-мереж. Він забезпечує аналіз імовірнісних характеристик процесу, дозволяючи враховувати взаємозв’язки між різними етапами й моделювати можливі сценарії атакувальних дій.

Цей розділ розглядає створення математичної моделі, заснованої на концепції GERT-мереж, яка використовується для оцінки дій на ранніх етапах формування коду атаки. Окремий акцент зроблено на тому, як такі дії впливають на ресурси корпоративної мережі й на процеси керування ними. Розроблена модель дає змогу аналізувати вхідні параметри, що визначають можливості атакуючих, включно з характеристиками мережевих слабких місць і ймовірностями реалізації кожного з можливих варіантів.

Особливістю представленої моделі є відображення ключових вузлів і залежностей між ними, які відповідають за динаміку розвитку кібератаки. Випадкові зміни характеристик елементів мережі також враховані, що дозволяє оцінити варіативність їхньої поведінки та ефективність впроваджених заходів захисту.

Запропонований підхід забезпечує інструмент для прогнозування ризиків і виявлення слабких місць, що можуть бути використані в атаках, дозволяючи підвищити рівень інформаційної безпеки корпоративної інфраструктури.

Схема кібератаки, розроблена на основі GERT-мереж, включає послідовність взаємопов'язаних етапів, кожен із яких має свої ймовірнісні характеристики. Першим етапом є розвідка, коли зловмисник аналізує елементи корпоративної мережі, збираючи дані про IP-адреси, відкриті порти, типи протоколів та можливі вразливості. На підставі зібраної інформації здійснюється перехід до наступного етапу, який передбачає пошук недоліків у системі безпеки, наприклад, у вигляді застарілого програмного забезпечення чи помилок у конфігурації.

Далі здійснюється створення зловмисного програмного коду, призначеного для експлуатації знайдених вразливостей. Цей код впроваджується у цільові елементи мережі, причому методи його доставки можуть включати соціальну інженерію або використання програмних помилок. Після успішного впровадження код активується, виконуючи завдання, такі як створення прихованого доступу, організація DoS-атак або отримання несанкціонованого доступу до конфіденційних даних.

Якщо права доступу зловмисника обмежені, далі може здійснюватися ескалація привілеїв, що надає ширший контроль над системою. Після цього реалізується основна мета кібератаки, що може включати викрадення, зміну або знищення даних, а також саботаж. Завершальним етапом зазвичай є приховування слідів своєї діяльності або створення прихованих механізмів для подальшого доступу (рис.2.7).

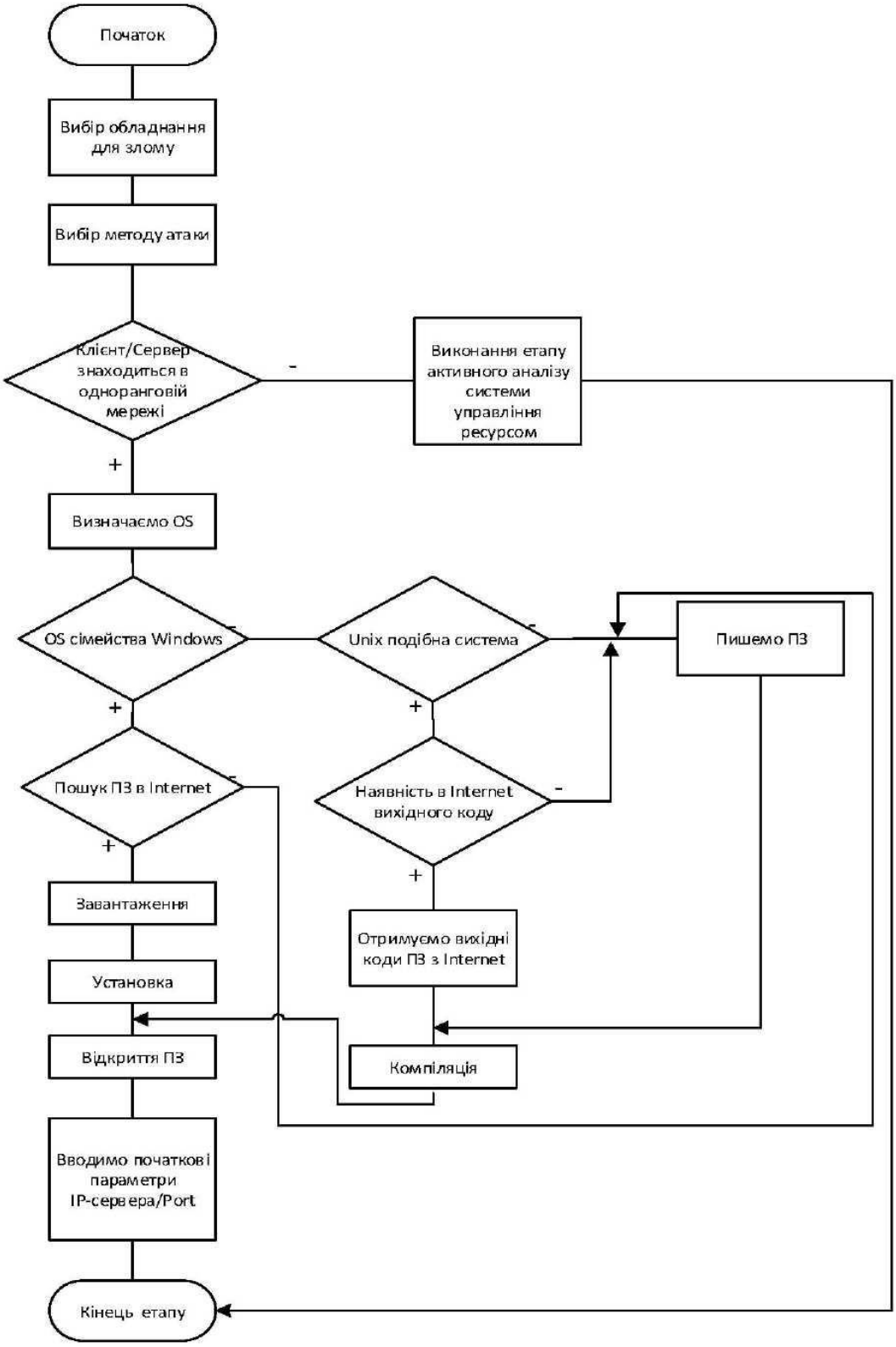


Рисунок 2.7 – Процес кібератаки

Модель GERT дозволяє чітко візуалізувати зв’язки між етапами нападу, враховуючи ймовірність успішного виконання кожного з них. Такий підхід сприяє оцінці ризиків, створенню планів захисту та підвищенню ефективності захисних

заходів. За необхідності можна запропонувати графічне представлення схеми для більш детального аналізу.

Перший етап злочинного впливу на корпоративну мережу зазвичай полягає у здійсненні кібератаки, спрямованої на її ресурси. У зв'язку з цим виникає потреба у проведенні детального аналізу характеристик таких атак і створенні відповідної моделі на основі GERT-методу. Такий підхід дозволяє ефективно моделювати й оцінювати різні етапи атаки, враховуючи їх ймовірність, залежності та вплив на загальну безпеку мережі.

Використання аналітичних інструментів і моделей на основі штучного інтелекту є доцільним при побудові моделі кібератаки на корпоративну мережу з кількох причин. По-перше, завдяки можливості аналізувати великі обсяги інформації, ШІ здатний враховувати різноманітні сценарії атак, потенційні загрози та вразливості, що знижує ризик упущення критичних деталей [43].

По-друге, такі системи дозволяють автоматизувати процес створення моделі, економлячи час і ресурси, які в іншому випадку були б витрачені на ручний аналіз складної мережевої інфраструктури. Моделі, створені за допомогою ШІ, можуть оперативно оновлюватися та адаптуватися до нових загроз, забезпечуючи актуальність аналізу.

ШІ може інтегруватися з існуючими методологіями, зокрема GERT-мережами, для створення комплексної, багаторівневої моделі, яка враховує ймовірнісні залежності між етапами атаки. Це дозволяє ефективніше оцінити вплив кожного компонента мережі на загальну безпеку [44].

Крім того, використання ШІ сприяє підвищенню точності моделювання, адже він здатний знаходити закономірності у складних даних, а також враховувати історичні дані про атаки та прогнозувати ймовірні майбутні загрози.

На основі такого підходу можна не лише моделювати кібератаки, а й формувати ефективні заходи протидії, сприяючи посиленню інформаційної безпеки корпоративної мережі (рис.2.8).

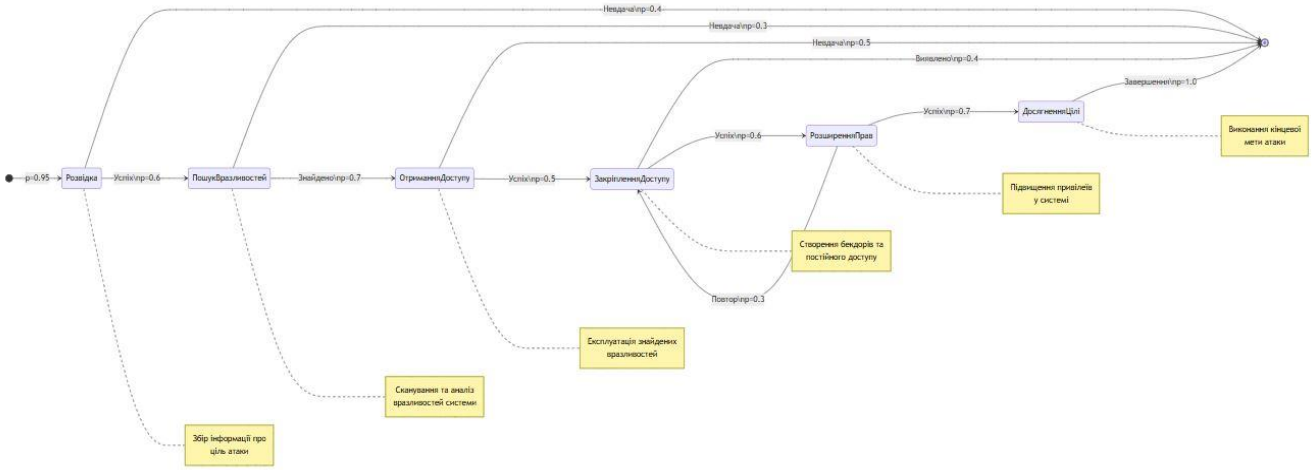


Рисунок 2.8 – Модель кібератаки, запропонована ШІ

У розробленій GERT-моделі відображено комплексний процес реалізації кібератаки на корпоративну мережу. Початковим етапом є розвідка, під час якої зловмисники збирають інформацію про ціль з ймовірністю успіху 0.6. У разі невдачі на цьому етапі (ймовірність 0.4) атака припиняється.

При успішній розвідці атакуючі переходять до пошуку вразливостей, де проводиться сканування та аналіз можливих точок входу в систему. Цей етап має ймовірність успіху 0.7, а у випадку невдачі (0.3) атака також завершується. Наступним кроком є отримання доступу через експлуатацію знайдених вразливостей, що має рівну ймовірність успіху та невдачі по 0.5.

Після успішного отримання доступу зловмисники намагаються закріпитися в системі, створюючи бекдори та забезпечуючи постійний доступ. На цьому етапі існує ймовірність виявлення 0.4, що призводить до блокування атаки. При успішному закріпленні (ймовірність 0.6) атакуючі переходять до розширення прав у системі.

Процес розширення прав має цікаву особливість - можливість циклічного повернення до етапу закріплення доступу з ймовірністю 0.3, що відображає ітеративний характер підвищення привілеїв у системі. При успішному розширенні прав (ймовірність 0.7) атакуючі переходять до фінального етапу - досягнення цілі атаки.

Фінальний етап має гарантоване завершення з ймовірністю 1.0, що означає або успішне виконання зловмисних намірів, або виявлення та блокування атаки системами захисту. Модель також включає детальні пояснювальні нотатки для кожного етапу, що описують конкретні дії та процеси, які відбуваються на відповідному кроці атаки (рис. 2.9).

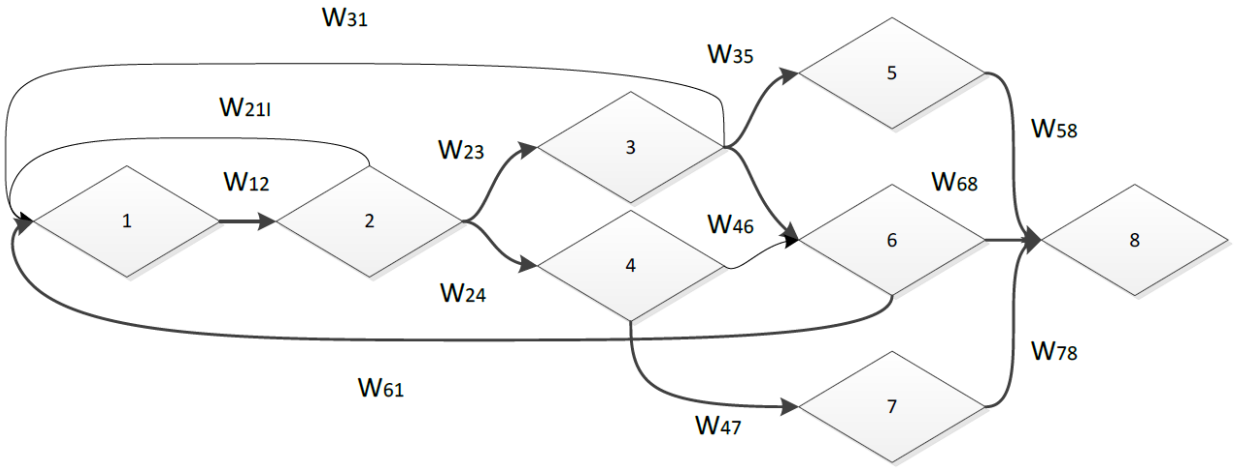


Рисунок 2.9 – Стохастична мережа, що ввідповідає алгоритму

Ця GERT-модель дозволяє краще зрозуміти структуру та ймовірнісні характеристики процесу кібератаки, що є важливим для розробки ефективних механізмів захисту та протидії на кожному з етапів несанкціонованого доступу до корпоративної мережі (рис. 2.10).

Якщо атакуючому не вдалося здійснити операцію налагодження свого ПЗ під час атаки, тоді виконується перехід (6,1). Запуск ШПЗ та введення параметрів IP-серверу жертви відображають переходи до вершини 8.

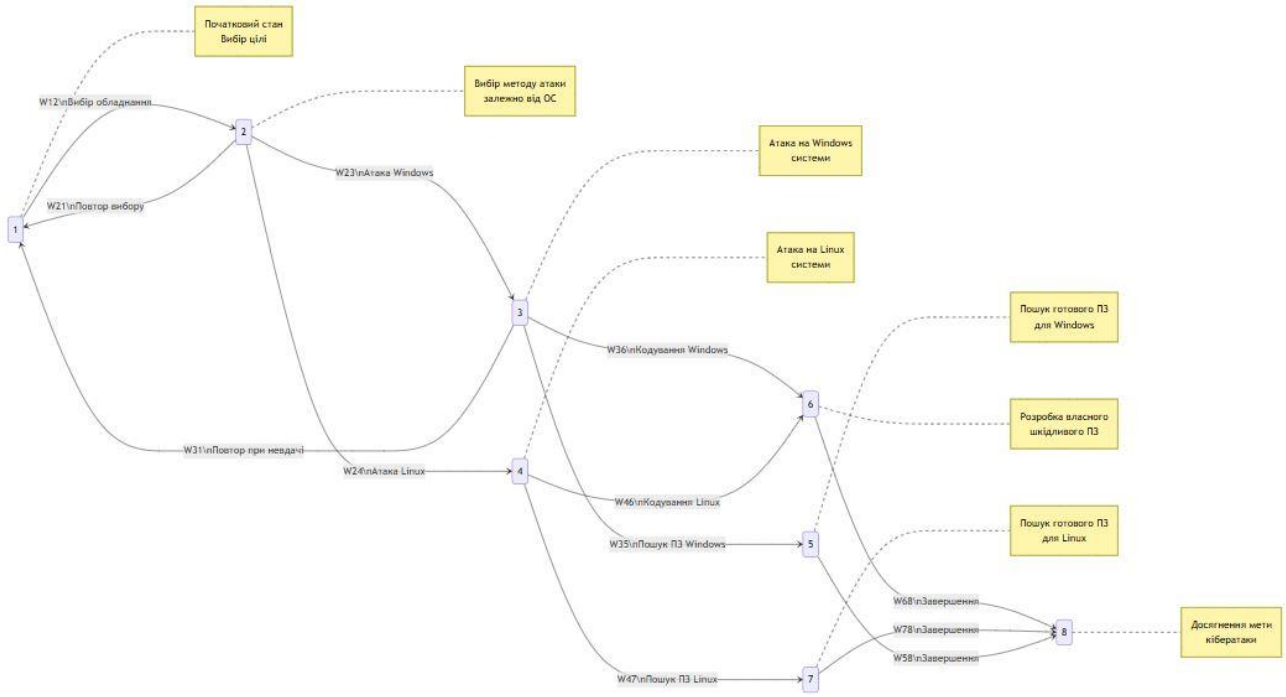


Рисунок 2.10 – Вдосконалена стохастична мережа проведення кібератаки

Процес кібератаки починається з вибору обладнання в мережі (перехід W12). Модель демонструє два основні напрямки атаки залежно від операційної системи цілі: Windows (перехід W23) або Linux (перехід W24). У разі невдалого вибору методу атаки передбачено повернення до початкового етапу через переходи W21 та W31.

Для кожної операційної системи існують два можливі шляхи реалізації атаки: пошук готового шкідливого програмного забезпечення в мережі Internet (переходи W35 для Windows та W47 для Linux) або розробка власного ПЗ (переходи W36 та W46 відповідно).

Кожен з цих шляхів веде до фінального стану (стан 8) через відповідні переходи (W58, W68, W78). Особливістю моделі є наявність циклічних переходів, які відображають можливість повторного вибору методу атаки у випадку невдачі, що робить модель більш реалістичною.

Всі переходи характеризуються випадковими величинами з певними законами розподілу, що відображає стохастичну природу процесу кібератаки. Це дозволяє більш точно моделювати реальні сценарії атак та розробляти ефективні методи протидії на кожному етапі.

На основі структурної схеми ми бачимо, що перехід W61 відображає повернення до початкового стану у випадку невдалого кодування та налагодження програмного забезпечення. Переходи W58, W68 та W78 відображають відкривання злочинного програмного забезпечення і ввід параметрів IP-серверу комп'ютера-жертви, що веде до завершального стану 8.

Таким чином, кожний перехід в GERT-моделі характеризується набором параметрів, що описують як ймовірнісні характеристики виконання операцій, так і їх часові показники. Це дозволяє створити повну математичну модель процесу кібератаки та проаналізувати різні сценарії її розвитку. На основі цих характеристик можна розробляти ефективні методи протидії кібератакам, враховуючи найбільш вірогідні шляхи їх реалізації та критичні точки в процесі атаки.

Взявши до уваги характеристику GERT- функцію часу атаки на ресурси корпоративної мережі *W*- можна представити як:

$$W_E(s)=\frac{\left(\begin{array}{l}W_{12}W_{23}W_{35}W_{58}+W_{12}W_{23}W_{36}W_{68}+W_{12}W_{24}W_{46}W_{68}+\\+W_{12}W_{24}W_{47}W_{78}\end{array}\right)}{1-W_{12}W_{21}-W_{12}W_{23}W_{31}-W_{12}W_{23}W_{36}W_{61}-W_{12}W_{24}W_{46}W_{61}}=$$

$$=\frac{\left(\begin{array}{l}p_1\left(\lambda_1/\lambda_1-s\right)\left(\begin{array}{l}p_2p_4^2\lambda_2^3\left(\lambda_1-s\right)\left(\lambda_4-s\right)^2\left(\lambda_5-s\right)+\\+p_1p_2p_6q_2\lambda_1\lambda_2\lambda_4\lambda_5\left(\lambda_2-s\right)^2\left(\lambda_4-s\right)+\\+p_1p_5p_6q_1\lambda_1\lambda_4^2\lambda_5\left(\lambda_2-s\right)^3+\\+p_1q_1q_3^2\lambda_1\lambda_2^2\lambda_4\left(\lambda_2-s\right)\left(\lambda_5-s\right)\end{array}\right)\end{array}\right)}{\left(\begin{array}{l}\left(\lambda_1-s\right)\left(\lambda_2-s\right)\left(\lambda_3-s\right)\left(\lambda_4-s\right)^2\left(\lambda_6-s\right)-\\-p_1p_2\lambda_1\lambda_3\left(\lambda_2-s\right)\left(\lambda_4-s\right)^2\left(\lambda_6-s\right)-\\-p_1p_2p_3\lambda_1\lambda_2\lambda_3\left(\lambda_4-s\right)^2\left(\lambda_6-s\right)-\\-p_1p_2q_1q_2p_3\lambda_1\lambda_2\lambda_4\lambda_6\left(\lambda_4-s\right)\left(\lambda_3-s\right)-\\-p_1p_5q_1q_4p_3\lambda_1\lambda_4^2\lambda_6\left(\lambda_2-s\right)\left(\lambda_3-s\right)\end{array}\right)}{\left(\lambda_1-s\right)\left(\lambda_2-s\right)\left(\lambda_3-s\right)\left(\lambda_4-s\right)^2\left(\lambda_6-s\right)}. \quad (2.1)$$

де $q_1=1-p_2-p_3;\ q_2=1-p_3-p_4;\ q_3=1-p_5;\ q_4=1-p_6.$

Розроблена стохастична GERT-модель процесу кібератаки демонструє суттєві інноваційні характеристики порівняно з існуючими моделями. Її унікальність полягає насамперед у реалізації двонаправленого підходу до проведення кібератаки через врахування специфіки операційних систем Windows та Linux, що відображається у паралельних гілках процесу атаки з відповідними переходами.

Щільність розподілу ймовірності виникнення кібератаки, буде визначатися

$$\phi(\mathrm{x}) = \frac{1}{2\pi i}\int_{-i\infty}^{i\infty} e^{zx}\frac{-\mathrm{yz}^6 + \mathrm{bz}^5 - tz^4 + \mathrm{uz}^3 - kz^2 + wz - h}{\left(-z^6 + jz^5 - dz^4 + gz^3 - vz^2 + rz - c\right)\times}, \qquad (2.2)$$
$$\times\left(\lambda_1 + z\right)\left(\lambda_2 + z\right)\left(\lambda_5 + z\right)$$

Важливою особливістю моделі є включення можливості як використання готових рішень через пошук в Internet, так і розробки власного шкідливого програмного забезпечення, що робить модель більш гнучкою та наближеною до реальних сценаріїв кібератак. Модель також враховує ітеративний характер процесу через наявність зворотних зв'язків при невдалому виборі методу атаки та при невдалому кодуванні програмного забезпечення.

Суттєвою перевагою розробленої моделі є опис процесу за допомогою стохастичних характеристик, що включають ймовірності виконання кожного переходу, часові параметри операцій, твірні функції моментів та W-функції для кожного переходу. Це дозволяє проводити глибокий кількісний аналіз процесу кібератаки та оцінювати ймовірності успішної реалізації різних сценаріїв.

Практична цінність моделі реалізується через можливість визначення критичних точок та вразливих місць у процесі кібератаки, що є основою для розробки ефективних механізмів захисту. Розуміння повного циклу атаки, включаючи етапи генерації коду та пошуку рішень в Internet, дозволяє створювати більш досконалі системи кіберзахисту.

Таким чином, розроблена модель надає комплексний інструментарій для дослідження процесів кібератак з урахуванням сучасних методів їх реалізації, що робить її ефективним засобом для розробки систем кіберзахисту та протидії несанкціонованому доступу до корпоративних мереж.

2.3 Висновки

У цьому розділі було проведено аналіз моделювання процесу реалізації кібератаки, орієнтованої на корпоративні мережі. Визначено, що забезпечення безпеки інформаційної інфраструктури неможливе без ретельного попереднього аналізу, який здійснюється за допомогою методів пентестингу для виявлення вразливих точок у системі.

Розроблені математичні моделі зосереджуються на початкових етапах створення шкідливих кодів для реалізації кібератак та на діагностиці системи керування ресурсами мережі. Перша модель дозволяє детально дослідити процес створення коду атаки, зокрема ідентифікувати ключові етапи, які потребують особливого захисту. Це дає змогу формувати рекомендації для мінімізації ризиків і підвищення стійкості до атак.

Друга, удосконалена модель, спрямована на аналіз і діагностику керування ресурсами корпоративної мережі, демонструє значний потенціал у підвищенні точності тестування безпеки програмного забезпечення. Вона дозволяє виявляти проблемні ділянки в системі, оцінювати ефективність механізмів захисту та оптимізувати процеси керування.

Ця модель забезпечує візуальне представлення послідовності станів корпоративної мережі в контексті потенційних кібератак, дозволяючи розглядати можливі сценарії та розробляти заходи захисту, адаптовані до конкретних умов. Ця модель є цінним інструментом для оцінки ефективності реалізованих механізмів безпеки, оскільки враховує ймовірнісний характер подій і взаємодій між елементами системи. Його застосування допоможе створити стратегію тестування, яка адаптується до змін у кіберсередовищі, підвищуючи здатність

організації виявляти атаки та протидіяти їм. У майбутньому ця розробка може бути інтегрована в комплексні системи моніторингу та аналізу безпеки для зниження ризиків і підвищення стійкості мережі до зовнішнього впливу. Отримані результати є основою для подальшого розвитку методів тестування безпеки, що враховують динамічні зміни в архітектурі мережі та нові види загроз. Запропоновані моделі забезпечують більш глибоке розуміння взаємодії між елементами корпоративної системи, що сприяє розробці комплексних рішень для підвищення інформаційної безпеки. У майбутньому такі підходи можуть бути інтегровані в автоматизовані системи моніторингу для забезпечення своєчасного реагування на загрози.

3 МЕТОД ТЕСТУВАННЯ БЕЗПЕКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МЕРЕЖІ

3.1 Процес діагностики корпоративної мережі

На основі проведеного аналізу літератури щодо тривалості етапів тестування безпеки корпоративних мереж, дійсно можна констатувати, що етап аналізу системи керування ресурсом займає значну частину часу - понад 45% від загальної тривалості процесу тестування (рис. 3.1) [45-48] .

Рисунок 3.1 – Розподіл часу тестування мережі

Така значна тривалість етапу аналізу системи керування ресурсом обумовлена кількома ключовими факторами. По-перше, необхідністю проведення безпечного тестування, що вимагає ретельного планування та виконання тестів без ризику порушення працездатності мережі. По-друге, варіативністю реакції мережі на однакові тестові впливи, що потребує множинних повторень тестів для отримання достовірних результатів.

Процес аналізу системи керування ресурсом включає декілька підетапів: встановлення безпечного з'єднання з мережею, формування адаптивних тестових

векторів, аналіз реакції мережі на тестові впливи, оцінка поточного стану системи. Кожен з цих підетапів вимагає часу на виконання та аналіз результатів.

Важливим фактором, що впливає на тривалість, є необхідність врахування стохастичного характеру реакції мережі. Оскільки однакові тестові впливи можуть викликати різні реакції системи, потрібен додатковий час на статистичну обробку результатів та формування достовірних оцінок стану безпеки.

Математична модель процесу тестування повинна враховувати часові характеристики через функцію тривалості τі для кожного етапу:

$$T = \sum \tau i(Ti, Ri, Si), \quad (3.1)$$

де:

– Ti - тестовий вектор;

– Ri - реакція системи;

– Si - стан системи.

Така значна часова складова аналізу системи керування ресурсом підкреслює необхідність оптимізації процесу тестування та розробки методів скорочення тривалості цього етапу без втрати якості діагностики безпеки мережі. Це може бути досягнуто через впровадження автоматизованих засобів аналізу, паралельного виконання тестів та використання адаптивних алгоритмів формування тестових впливів.

Неспецифічність тестових переборів є дійсно важливою особливістю процесу аналізу системи керування ресурсом мережі. Це обумовлено тим, що пентестер, імітуючи дії потенційного зловмисника, використовує різноманітні підходи до тестування безпеки мережевих ресурсів (рис. 3.2).

Ключова проблема полягає в тому, що тестові сценарії не завжди можуть повністю відтворити реальну атаку через:

– різноманітність можливих векторів атаки;

– комбінування різних методів проникнення;

– адаптивність дій зловмисника;

– обмеження на безпечність тестування;

– варіативність реакції системи.

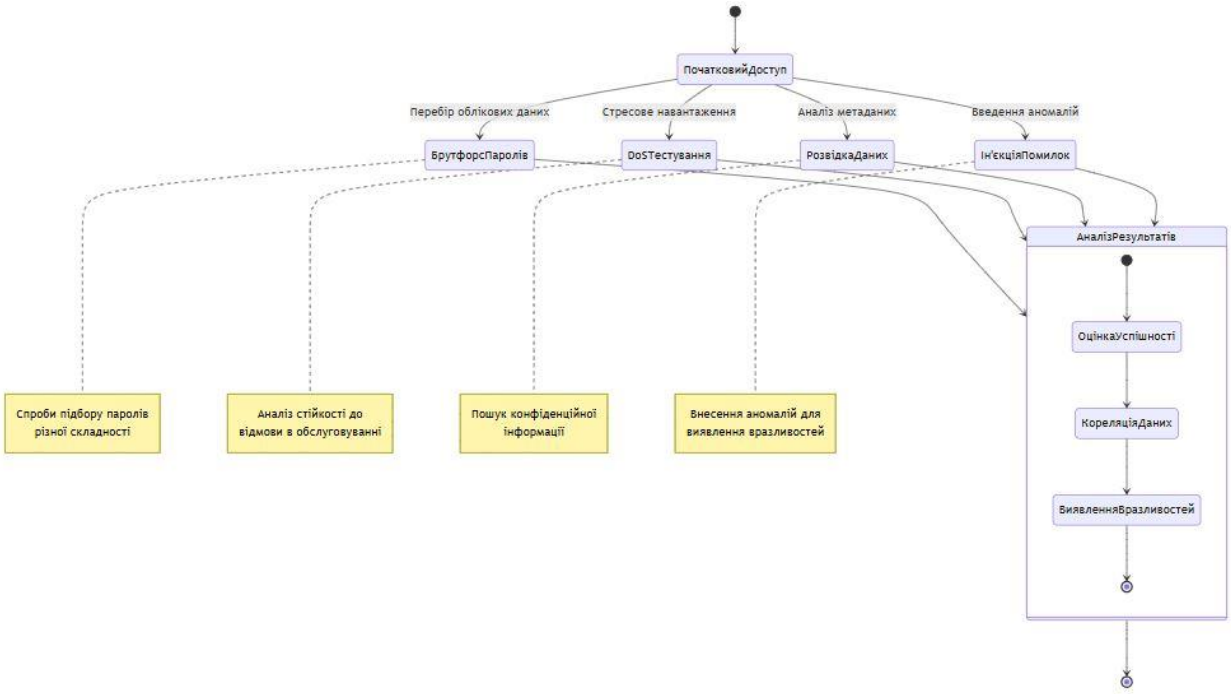


Рисунок 3.2 – Модель різновекторної атаки при тестувані

Це створює необхідність розробки комплексного підходу до тестування, який би враховував:

– множинність векторів атаки;

– взаємозв'язок різних методів тестування;

– адаптивність тестових сценаріїв;

– реалістичність імітації атак;

– безпечність проведення тестів.

Така модель дозволяє більш точно відобразити реальні сценарії атак при проведенні тестування безпеки корпоративної мережі, хоча і не може гарантувати повного покриття всіх можливих варіантів атак.

На основі вказаних вихідних даних можна сформулювати математичний опис алгоритму аналізу діагностики системи керування ресурсом мережі з урахуванням особливостей безпечного тестування (рис.3.3).

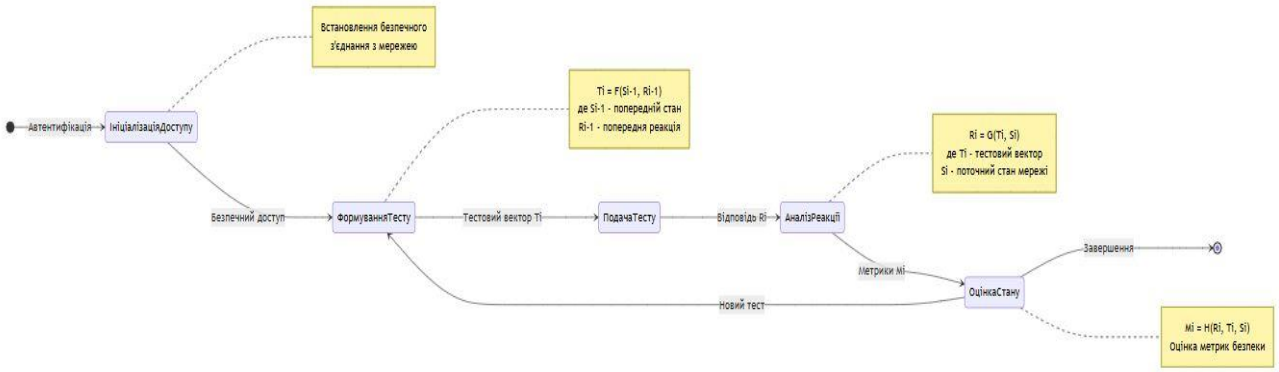


Рисунок 3.3 – Математична модель процесу діагностики корпоративної мережі

Дана математична модель враховує ключові аспекти безпечного тестування мережі. Функція формування тестів *F* адаптується до попередніх результатів, забезпечуючи безпечність впливу. Функція реакції *G* відображає варіативність відгуку мережі на однакові тести в різні моменти часу. Функція оцінки *H* забезпечує комплексний аналіз стану безпеки системи.

Важливою особливістю моделі є врахування ймовірнісного характеру реакції мережі на тестові впливи та наявність обмежень, що гарантують безпечність процедури тестування. Інтегральний показник *Q* дозволяє визначити момент досягнення необхідного рівня діагностики безпеки системи.

Запропонована модель дозволяє формалізувати процес тестування безпеки мережі з урахуванням вимог щодо збереження її працездатності та варіативності реакції на тестові впливи.

У представленій моделі перший вузол відображає початковий стан корпоративної мережі перед впливом алгоритму, що виконує аналіз системи керування ресурсами. Другий вузол демонструє стан, за якого неможливо отримати доступ до ресурсів, необхідних для керування мережею. Третій вузол, навпаки, ілюструє ситуацію, коли такі ресурси успішно перехоплено для цілей керування мережею. Четвертий вузол описує процес усунення виявлених помилок, які можуть виникнути в результаті впливу. П’ятий вузол характеризує ситуацію, коли алгоритм аналізу системи керування ресурсами виконується частково. Шостий вузол символізує стан завершеної перевірки корпоративної мережі, яка охоплює всі необхідні аспекти аналізу (рис. 3.4).

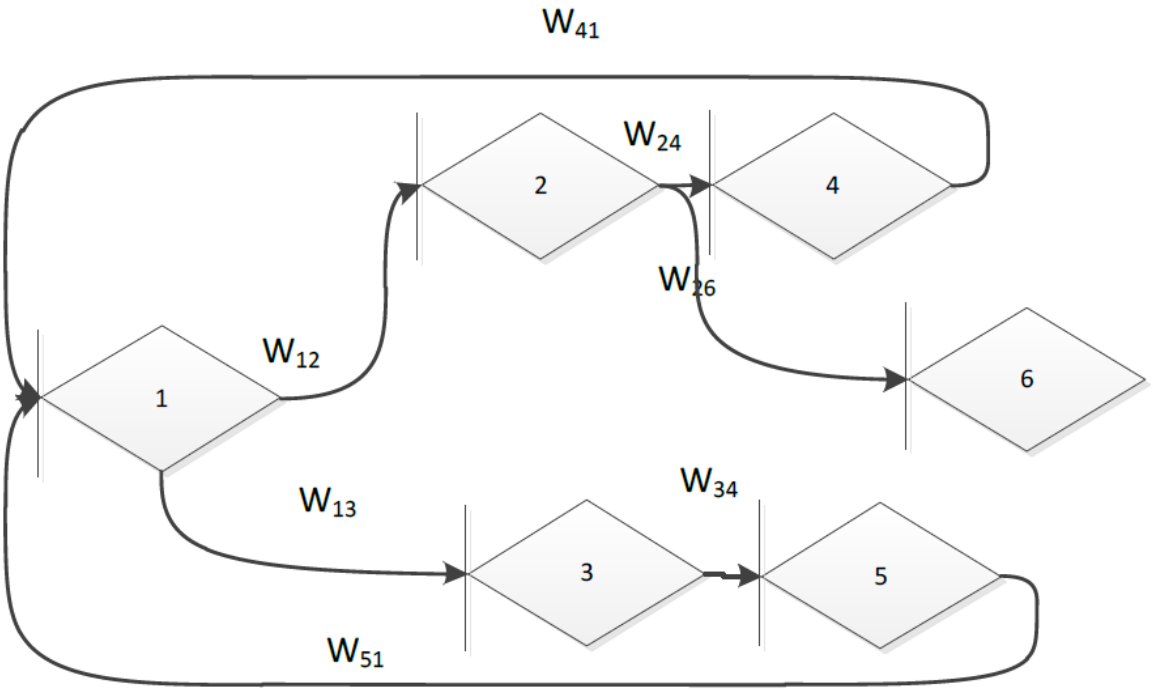


Рисунок 3.4 – GERT модель процесу діагностики корпоративної мережі

Ребро (1-2) показує вплив на корпоративну мережу описаного алгоритму, який призвів атакуючого до позитивного результату. За коригування помилок відповідає ребро (2-4). Ребро (4-1) зображає повернення в початковий стан корпоративної мережі. Ребро (2-6) описує потрапляння шкідливого коду в мережу. Ребро (1-3) зображає вплив на корпоративну мережу описаного алгоритму, який не дав позитивного результату. Ребро (3-5) описує оцінку повноти аналізу, при застосуванні певного алгоритму.

Відповідно до характеристик ребер та вершин GERT-мережі еквівалентну W-функцію застосування алгоритмів системи діагностики корпоративної мережі можна зобразити як:

$$W_E(s)=\frac{W_{12}W_{26}+W_{12}W_{24}W_{41}W_{12}W_{26}+W_{13}W_{35}W_{51}W_{12}W_{26}}{1-W_{12}W_{24}W_{41}W_{45}W_{51}-W_{13}W_{35}W_{51}}=$$

$$=\left(\frac{\begin{pmatrix}\left(p_1q_1\lambda_1\lambda_3(\lambda_1-s)(\lambda_2-s)(\lambda_4-s)(\lambda_5-s)(\lambda_6-s)\right)+\\+\left(p_1^2p_2p_3q_1\lambda_1^2\lambda_2\lambda_3\lambda_4(\lambda_5-s)(\lambda_6-s)\right)+\\+\left(p_1p_4^2q_1q_2\lambda_1^2\lambda_3\lambda_5\lambda_6(\lambda_2-s)(\lambda_4-s)\right)\end{pmatrix}}{(\lambda_1-s)^2(\lambda_2-s)(\lambda_3-s)(\lambda_4-s)(\lambda_5-s)(\lambda_6-s)}}{1-\left(\frac{\begin{pmatrix}p_1p_2p_3\lambda_1\lambda_2\lambda_4(\lambda_5-s)(\lambda_6-s)+\\+p_4^2q_2\lambda_1\lambda_5\lambda_6(\lambda_2-s)(\lambda_4-s)\end{pmatrix}}{(\lambda_1-s)(\lambda_2-s)(\lambda_4-s)(\lambda_5-s)(\lambda_6-s)}\right)}\right) \quad (3.2)$$

Отже, створено математичну модель, яка описує процес керування ресурсами корпоративної мережі. Ця модель може бути застосована для вивчення аспектів тестування безпеки мережі з метою мінімізації її слабких місць. Використання цієї моделі дозволяє проводити більш точний аналіз потенційних загроз і вразливостей, а також розробляти заходи, спрямовані на підвищення рівня захисту корпоративної інфраструктури.

3.2 Виділення алгоритму з бінарного коду

IxChariot - це програмний продукт, розроблений компанією Ixia (тепер є частиною Keysight Technologies), що використовується для моделювання, аналізу та тестування мережевої продуктивності. Це потужний інструмент, який дозволяє оцінювати швидкість передачі даних, затримки, втрати пакетів і інші параметри роботи мережі, а також ефективність роботи додатків у різних умовах [48, 49].

Ця версія IxChariot забезпечує можливість створювати реалістичні моделі мережевого трафіку, дозволяючи симулювати тисячі різних користувачів і додатків одночасно. Система підтримує перевірку продуктивності як дротових, так і бездротових мереж (Wi-Fi5G, LTE) (рис. 3.5).

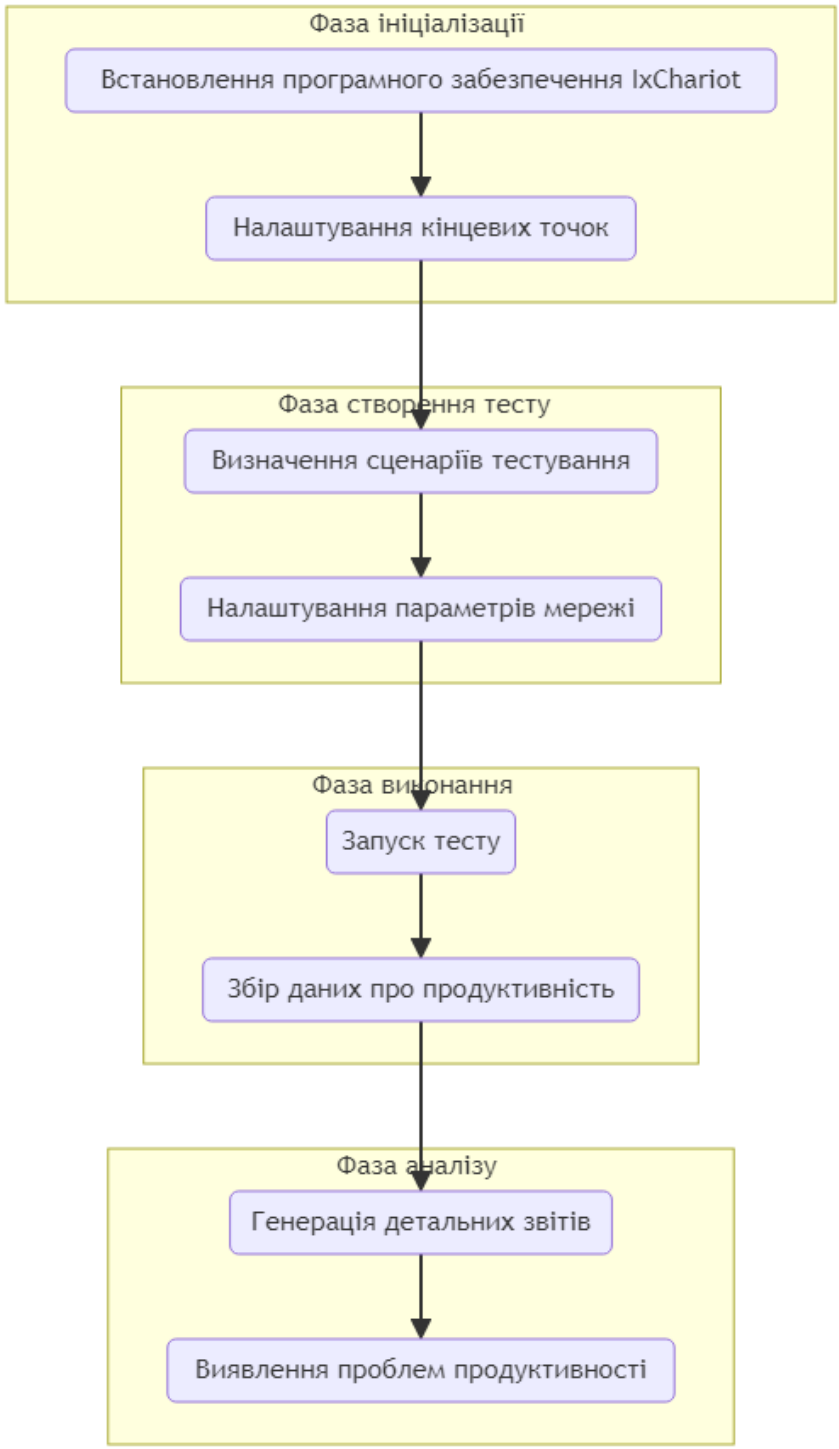


Рисунок 3.5 – Процес використання IxChariot для діагностування мережі

Процес моделювання мережевої діагностики починається з налаштування середовища IxChariot, що передбачає розгортання консолі управління та встановлення кінцевих точок на ключових вузлах мережевої інфраструктури.

Ключові особливості IxChariot включають:

– підтримку різних протоколів і сценаріїв, що дозволяє адаптувати тести під конкретні потреби користувача;

– можливість тестування як локальних мереж, так і великих розподілених систем;

– підтримку інтеграції з автоматизованими системами управління мережею;

– використання бібліотеки шаблонів трафіку для моделювання додатків, таких як відеоконференції, потокове передавання медіа, веб-додатки та інше;

– зрозумілий інтерфейс із можливістю генерування звітів у реальному часі для аналізу отриманих результатів.

На цьому кроці відбувається початкове налаштування системи та перевірка з’єднань між компонентами. Після завершення налаштування переходимо до налаштування тестових сценаріїв. Тут визначаються параметри тестування, включаючи типи мережевого трафіку, інтенсивність навантаження та тривалість тесту. IxChariot дозволяє емулювати різні мережеві протоколи та служби, що забезпечує комплексну оцінку продуктивності.

У процесі генерації навантаження система створює синтетичний трафік відповідно до заданих параметрів. Паралельно відбувається постійне вимірювання показників продуктивності мережі, таких як пропускна здатність, затримки, втрати пакетів та джиттер. При виявленні відхилень від очікуваних значень параметри тестування можуть автоматично коригуватися.

Моніторинг здійснюється в режимі реального часу з постійною перевіркою досягнення порогових значень. Система відстежує динаміку змін продуктивності та фіксує будь-які аномалії у роботі мережі. Особлива увага приділяється критичним показникам, які можуть вплинути на якість обслуговування.

Етап обробки результатів включає глибокий аналіз зібраних даних з використанням вбудованих інструментів IxChariot. Проводиться співставлення отриманих метрик з еталонними значеннями, виявляються вузькі місця та потенційні проблеми в роботі мережі.

Завершальною фазою стає формування детального звіту, який містить візуалізацію результатів тестування, статистичні дані та рекомендації щодо оптимізації мережевої інфраструктури. Цей звіт служить основою для прийняття рішень щодо модернізації або реконфігурації мережі.

IxChariot використовується мережевими адміністраторами, постачальниками послуг і виробниками мережевого обладнання для тестування продуктивності мережевої інфраструктури перед її впровадженням або оновленням. Завдяки своїм можливостям інструмент також знаходить застосування в індустрії безпеки, де можливості мережі перевіряються в стресових умовах, таких як високий рівень трафіку або DDoS-атаки.

Такий підхід до діагностування забезпечує комплексну оцінку стану мережевої інфраструктури та дозволяє своєчасно виявляти потенційні проблеми до їх критичного впливу на роботу системи.

Процес починається із завантаження двійкового коду та його базової трансформації через дизасемблювання та декомпіляцію. На цьому етапі код перетворюється у більш зрозумілий формат для подальшого аналізу. Наступним кроком відбувається пошук характерних векторів ознак у трансформованому коді.

При успішному виявленні векторів ознак запускається глибинний аналіз, який включає дослідження виявлених шаблонів, вивчення потоків керування та структур даних. У випадку, коли вектори ознак не вдається виявити одразу, застосовується комплексне дослідження через структурний та функціональний аналіз коду (рис.3.6).

Обидва шляхи аналізу ведуть до етапу відновлення логіки, де відбувається реконструкція алгоритмічної структури програми. Після цього проводиться ретельна валідація отриманих результатів. Якщо перевірка виявляє помилки, процес повертається до етапу базової трансформації для повторного аналізу [49].

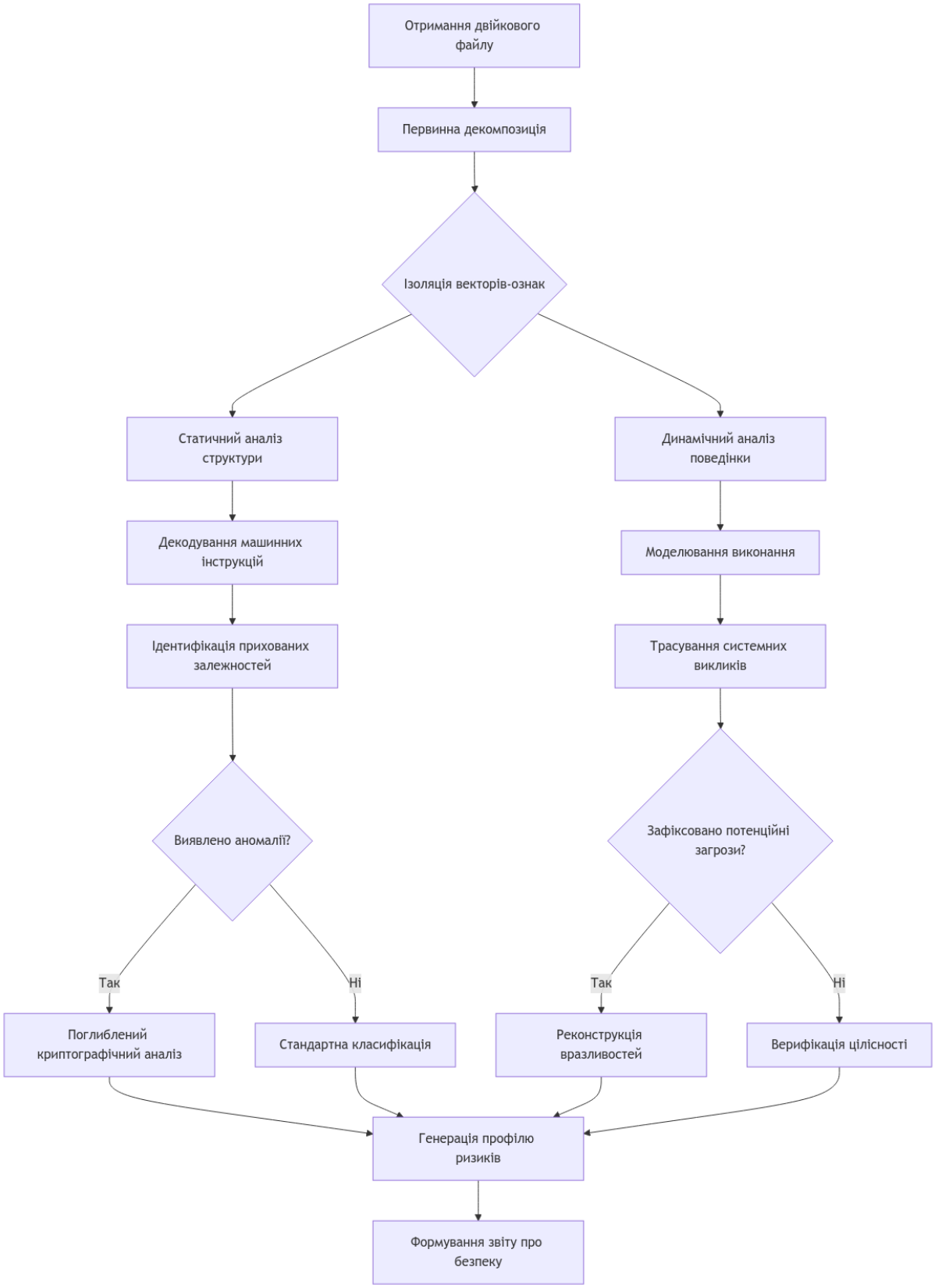


Рисунок 3.6 – Проникнення в мережу через машинно-незалежне відновлення алгоритму

Процес розпочинається з накопичення та попередньої обробки масиву двійкових векторів-ознак. На етапі препроцесингу відбувається нормалізація даних, видалення шумів та підготовка до структурного аналізу. Ключовим є забезпечення максимальної чистоти та релевантності вихідних даних.

Наступний етап - комплексний аналіз структурних патернів, де здійснюється пошук закономірностей та нехарактерних послідовностей у бінарному коді. У разі виявлення стійких патернів запускається процес машинно-незалежної реконструкції алгоритму, який дозволяє відновити логіку виконання незалежно від конкретної апаратної архітектури.

Паралельно проводиться контекстуальне дослідження внутрішніх зв'язків між виявленими структурами. Це дає змогу зрозуміти глибинні взаємодії компонентів та виявити потенційні точки вразливості.

Результатом аналізу стає картування можливих вразливостей з подальшою оцінкою ризиків. Залежно від рівня потенційної загрози обираються різні стратегії дослідження: від поглибленої діагностики критичних компонентів до цільового аналізу окремих підсистем.

Ключовим моментом є відновлення повної логіки виконання коду, що дозволяє максимально точно змоделювати поведінку програмного забезпечення. На цьому етапі здійснюється пошук шкідливого коду, який може бути прихованим комп'ютерним вірусом або недокументованим функціоналом.

У випадку виявлення загрози запускається процедура термінової ізоляції з подальшим формуванням детального звіту безпеки. Якщо жодних небезпечних компонентів не знайдено, проводиться додаткова перевірка відповідності системи встановленим стандартам безпеки.

Унікальність методу полягає в його здатності працювати на рівні машинно-незалежного аналізу, що дозволяє виявляти вразливості незалежно від конкретної реалізації або архітектури системи.

Ключовою особливістю методології є комплексна оцінка, яка дозволяє не лише виявити приховані вразливості, але й змоделювати потенційні сценарії мережевого впливу.

Блок-схема отримання алгоритму з файлу зображен на рисунку 3.7.

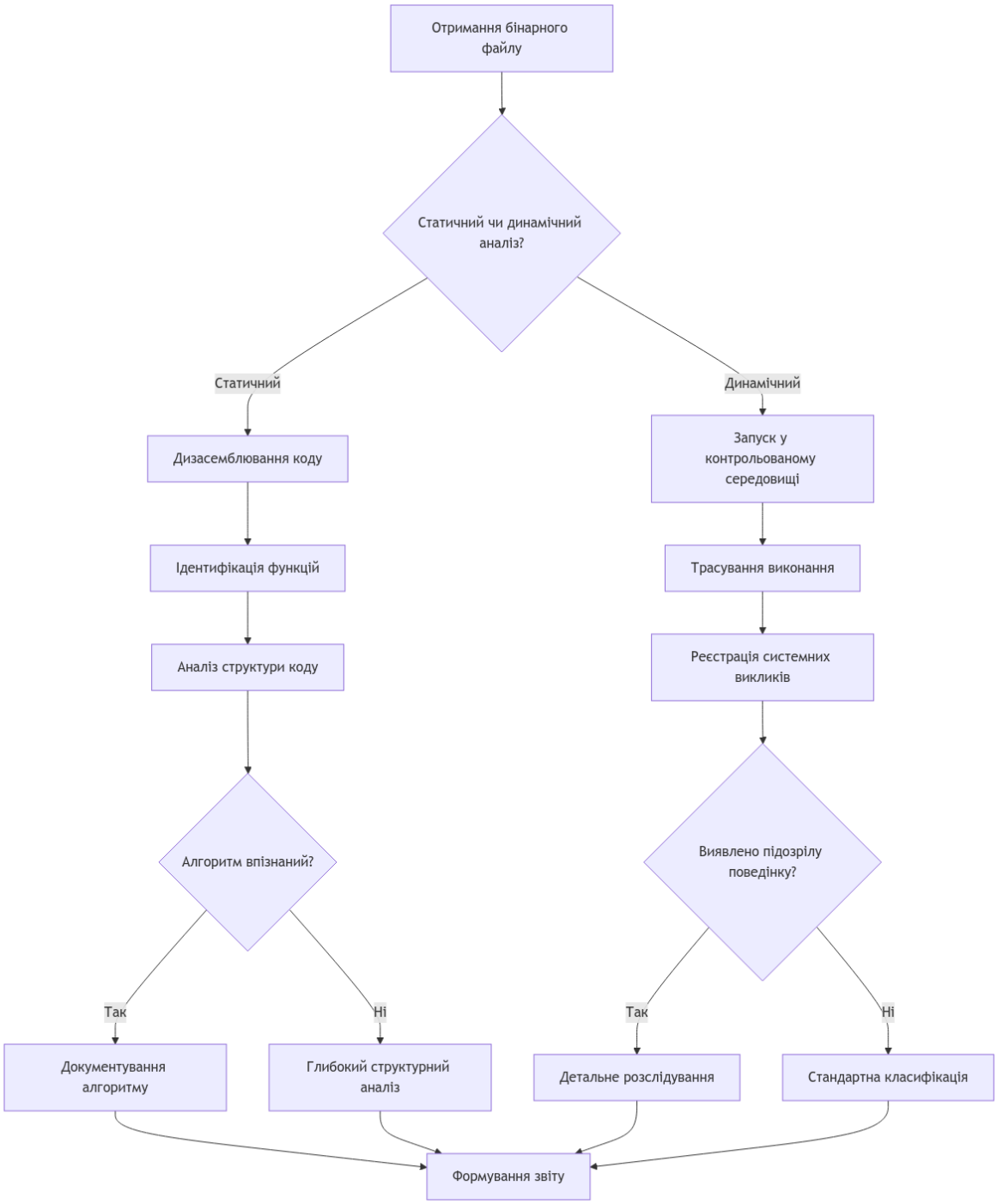


Рисунок 3.7 – Отримання алгоритму з бінарного коду

Результатом аналізу є формування детального профілю ризиків та звіту про безпеку, що надає вичерпну інформацію про можливі загрози в досліджуваному програмному забезпеченні.

При успішній валідації результати документуються з детальним описом відновленого алгоритму та його параметрів. Завершальним етапом стає безпековий аудит, під час якого проводиться аналіз потенційних вразливостей та формуються рекомендації щодо захисту.

Важливо зазначити, що цей метод дозволяє ефективно досліджувати приховані алгоритми в бінарному коді, що критично важливо для забезпечення безпеки корпоративних мереж. Особлива увага приділяється валідації на кожному етапі, що гарантує точність та надійність результатів аналізу (3.7).

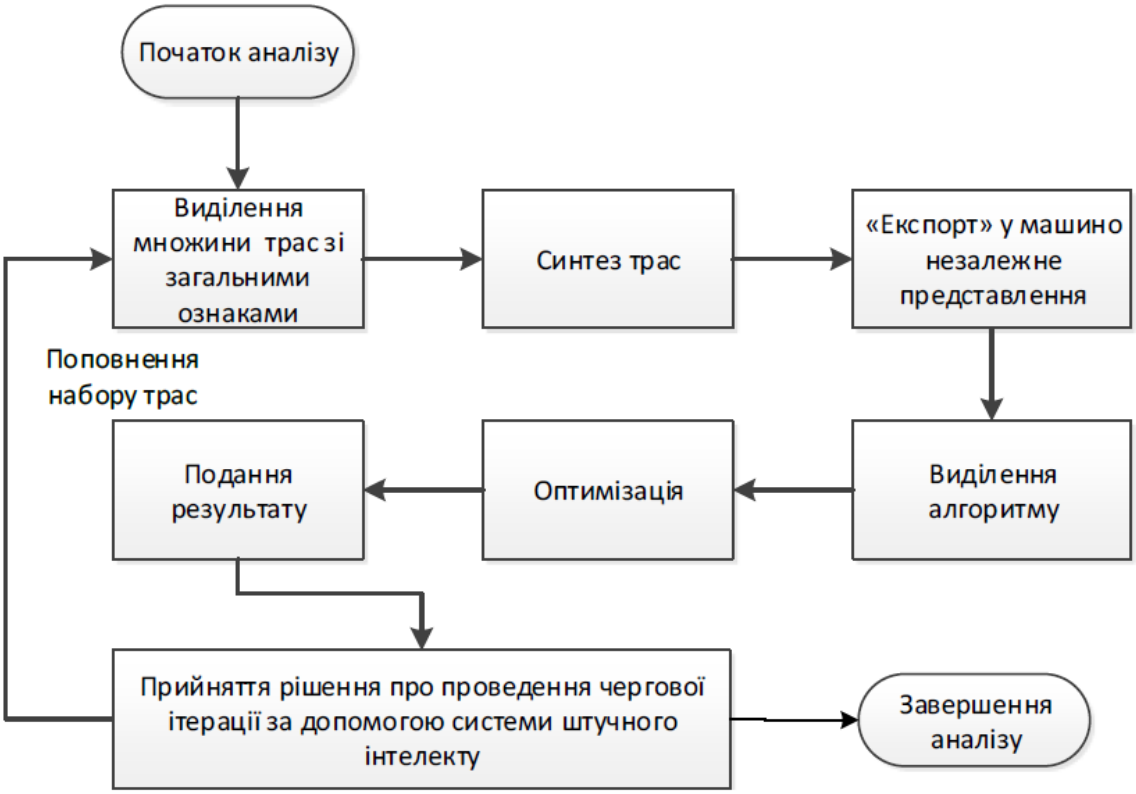


Рисунок 3.7 – Блок-схема методу пошуку алгоритму з бінарного коду

У разі аналізу простих програм достатньо одноразового проходження схеми, тоді як для складних програм може виникнути необхідність у повторенні алгоритму для досягнення необхідної точності. Схема пошуку алгоритму, як видно з рисунка, розпочинається з виділення множини трас – векторів з подібними ознаками, що утворюють основу для подальшого аналізу. На наступному етапі

відбувається синтез трас, де інформація про об’єкт дослідження представляється у вигляді графів, що спрощує подальшу обробку.

Отримані дані експортуються у машино-незалежне представлення, після чого виділяється алгоритм – конкретний код, що піддається аналізу. Наступним кроком є оптимізація для спрощення та узагальнення отриманих результатів, що полегшує їхнє використання. Завершальний етап – блок подання результату, який забезпечує зручний формат представлення коду для пентестера або для подачі в систему штучного інтелекту. Цей блок також визначає, чи потрібна додаткова ітерація аналізу, чи процес можна вважати завершеним. Такий підхід дозволяє гнучко адаптувати процес до складності аналізованої програми та вимог до кінцевого результату.

Особливістю методології є можливість багаторазового проходження циклу аналізу для складних програм. Кожна ітерація передбачає поступове уточнення та деталізацію інформації про досліджуваний алгоритм.

Ключові етапи включають:

- виділення множини трас;
- графове представлення;
- машинно-незалежне перетворення;
- виділення цільового алгоритму;
- оптимізацію результатів.

Впровадження штучного інтелекту дозволяє динамічно приймати рішення про продовження або завершення аналізу, що робить процес більш гнучким та інтелектуальним.

3.3 Висновки

У дослідженні впроваджено інноваційний методологічний підхід до розкриття внутрішньої структури програмних компонентів через деталізований алгоритмічний аналіз двійкового коду в контексті мережевої безпеки.

Процес декомпозиції складається з послідовних взаємопов'язаних етапів трансформації інформаційних потоків. Початкова стадія передбачає ідентифікацію сукупності трасувальних векторів, що містять характерні інформативні ознаки досліджуваного об'єкта.

Наступним кроком виступає графова репрезентація отриманих даних, де кожен вектор перетворюється на структурований вузол з притаманними взаємозв'язками. Такий методологічний прийом дозволяє представити складну інформаційну модель у вигляді логічно вибудованої графічної конструкції.

Принципово важливим етапом є трансформація інформаційного масиву в машинно-незалежне представлення, що забезпечує абстрагування від специфічних архітектурних особливостей первинного коду. У межах цієї стадії здійснюється цільова ізоляція алгоритмічного контуру, який підлягає подальшому аналізу.

Оптимізаційна фаза спрямована на редукцію отриманих результатів, їх змістовне очищення та приведення до формату, максимально зручного для сприйняття експертами з кібербезпеки. Додатковим завданням виступає підготовка даних для подальшої машинної обробки системами штучного інтелекту.

Розроблена методика передбачає можливість багатократного ітераційного аналізу, де кожен наступний цикл деталізує та поглиблює первинні дослідницькі гіпотези. Інтелектуальна система приймає рішення про доцільність продовження або завершення аналітичного процесу.

Створений графічний інструментарій дозволяє комплексно досліджувати різні типи кодових структур - як статичні інструкційні послідовності, так і динамічні виконавчі контексти. Це забезпечує універсальність методології та її придатність для широкого спектра дослідницьких завдань у сфері кібербезпеки.

Принципова новизна підходу полягає в інтеграції графового моделювання, машинно-незалежної трансформації та інтелектуальної системи підтримки прийняття рішень в єдиний методологічний алгоритм діагностики потенційних загроз корпоративних інформаційних мереж.

4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ

4.1 Процес виділення множини векторів із загальними ознаками

Двійковий вектор виконання являє собою складну інформаційну структуру, що відображає послідовність машинних інструкцій під час виконання програмного коду [50-52]. Сутність такого вектора полягає в детальному трасуванні кожного кроку виконання команди безпосередньо в симульованому середовищі.

Процес формування двійкового вектора розпочинається з ініціалізації симулятора, який точно відтворює апаратне середовище та системні характеристики цільової платформи. Симулятор забезпечує покрокове виконання інструкцій, фіксуючи найменші зміни в стані процесора, регістрах та оперативній пам'яті.

Кожен крок вектора містить:

- адресу поточної інструкції;
- машинний код команди;
- стан процесорних регістрів до виконання;
- стан процесорних регістрів після виконання;
- зміни в оперативній пам'яті;
- системні виклики;
- контекст виконання.

Унікальність двійкового вектора виконання полягає в його здатності максимально точно відтворити всі внутрішні процеси, що відбуваються під час роботи програми. Це дозволяє дослідникам здійснювати глибокий аналіз поведінки коду без безпосереднього запуску на реальному обладнанні.

Симулятори використовують складні алгоритми декодування та інтерпретації машинних інструкцій, забезпечуючи високу точність трасування. Вони підтримують різні архітектури процесорів, дозволяючи аналізувати код, призначений для різних апаратних платформ.

Практичне застосування двійкових векторів виконання охоплює широкий спектр завдань:

- зворотна розробка програмного забезпечення;
- виявлення вразливостей безпеки;
- аналіз шкідливого коду;
- дослідження алгоритмічної поведінки;
- оптимізація програмних продуктів.

Важливою перевагою є можливість багаторазового відтворення ідентичних умов виконання, що робить двійкові вектори незамінним інструментом для системних дослідників та спеціалістів з кібербезпеки.

Розглянемо простий приклад двійкового вектора виконання для елементарної функції додавання двох чисел на мові Assembly для архітектури x86:

```
mov eax, 5      ; Завантаження числа 5 в регістр eax
mov ebx, 3      ; Завантаження числа 3 в регістр ebx
add eax, ebx    ; Додавання вмісту регістрів
```

Рисунок 4.1 – Фрагмент коду

Двійковий вектор в процесі виконання буде змінюватися в три етапи, напрклад стан регістру eахзміниться з 0x00000000 на першому етапі, до 0x00000005 на другому етапі, і 0x00000008 на останьому, а стан регістру ebx зміниться тільки два рази з 0x00000000 до 0x00000003 на другому, на третьому він змін не зазнає. Також в двійковий вектор виконання описує зміну адреси інструкції, машиного коду, флагів процесора, змінв пам’яті, системних викликів.

Такий деталізований вектор дозволяє повністю реконструювати процес виконання інструкцій, показуючи кожну мікрозміну в системі.

Практичне значення такого аналізу:

- точне відстеження логіки виконання коду;
- виявлення потенційних вразливостей;

– розуміння внутрішньої механіки програми;

– можливість моделювання різних сценаріїв виконання.

Концепція пов'язаних векторів у контексті тестування програмного забезпечення є складним методологічним підходом до комплексного аналізу поведінки системи [53-55].

Основна ідея полягає в тому, що пов'язані вектори формуються з єдиного стану ініціалізації системи, але відрізняються сценаріями виконання та варіативністю вхідних параметрів. Це дозволяє створити багатовимірну модель поведінки досліджуваної програми [56].

Механізм формування пов'язаних векторів передбачає:

– фіксацію первинного стану системи;

– трасування змін у системі для кожного сценарію;

– генерацію різноманітних вхідних впливів;

– збереження послідовності виконаних інструкцій;

– порівняння отриманих результатів.

Для інтерактивних програм вхідними даними можуть слугувати:

– послідовності натискання клавіш;

– маршрути переміщення курсору;

– послідовність натискання кнопок миші;

– взаємодія з графічними елементами інтерфейсу;

– введення текстових даних.

Головна мета методології - отримання набору пов'язаних векторів, що комплексно описують алгоритмічну поведінку системи через множину альтернативних сценаріїв виконання.

Практична цінність такого підходу полягає в можливості:

– виявлення прихованих алгоритмічних залежностей;

– ідентифікації потенційних вразливостей;

– аналізу повноти покриття тестових сценаріїв;

– реконструкції внутрішньої логіки програми;

– прогнозування поведінки в нестандартних ситуаціях.

Набір пов'язаних векторів виступає інформативним результатом дослідження, що дозволяє здійснити глибинний аналіз внутрішньої архітектури програмної системи.

Процес розпочинається з отримання набору пов'язаних векторів виконання, зібраних за допомогою спеціального симулятора з одного ініціалізованого стану формування системи. Кожен вектор являє собою послідовність кроків виконання команд, які реально спостерігалися під час роботи програми.

Далі відбувається трансформація у орієнтований граф $G = (V, C)$, де кожна вершина представляє базовий блок інструкцій - неперервну лінійну ділянку коду. Ребра графа з'єднують базові блоки, відтворюючи лише ті переходи керування, які були зафіксовані в оригінальних векторах виконання.

Особливістю такого графу є можливість роботи з кількома паралельними потоками виконання. У разі наявності множинних потоків, граф буде складатися з незалежних компонент зв'язності, кожна з яких відображає окремий потік виконання.

Для кожної вершини графу накопичується додаткова метаінформація: початковий адрес блоку, повна послідовність команд та номер покоління, що дозволяє простежувати еволюцію коду під час виконання програми. Номер покоління збільшується при кожному перезаписі коду в конкретному потоці.

Принципово важливою характеристикою графу є те, що він містить лише реально спостережені переходи та структури, без додаткових гіпотетичних припущень. Це забезпечує максимальну точність відображення поведінки оригінальної програми.

Крім того, в графі визначаються спеціальні службові вершини для кожного потоку, які мають унікальні властивості наслідування, що дозволяє більш глибоко аналізувати структуру алгоритму.

Кожна така вершина містить початкову адресу, послідовність команд та номер покоління. Номер покоління є цілим числом, яке відображає стан коду

команди на конкретний момент. Це значення забезпечує точне відображення змін у коді, що відбуваються під час його виконання.

У кожному окремому потоці номер покоління збільшується на одиницю при кожному перезаписі коду цього потоку. Такий механізм дозволяє фіксувати зміни та забезпечувати коректне відображення поточного стану команд, що є важливим для аналізу динаміки змін коду.

```
# Приклад вхідних двійкових векторів виконання

vectors = [
    {
        "initial_state": "start_program",
        "flow": [
            {"address": 0x1000, "instruction": "mov eax, 5"},     # Базовий блок 1
            {"address": 0x1004, "instruction": "add eax, 3"},
            {"address": 0x1008, "instruction": "cmp eax, 10"},    # Умовний перехід
            {"address": 0x100C, "instruction": "jge greater"}     # Перехід
        ],
        "generation": 1
    },
    {
        "initial_state": "start_program",
        "flow": [
            {"address": 0x1010, "instruction": "mov ebx, 0"},     # Базовий блок 2 (greater)
            {"address": 0x1014, "instruction": "ret"}             # Завершення
        ],
        "generation": 1
    }
]
```

Рисунок 4.2 – Приклад опису отриманого графа

Такий підхід дає змогу поступово реконструювати внутрішню логіку алгоритму, перетворюючи низькорівневий двійковий код на більш зрозумілу та структуровану послідовність логічних переходів та операцій.

Проблеми можуть виникати через поліморфну природу програм, коли кожна нова версія коду базується на попередній. Такий підхід значно ускладнює процес аналізу, оскільки код постійно змінюється. Зловмисники часто використовують цей метод для впровадження шкідливого коду, що дозволяє обійти сигнатурний аналіз, який виконують антивірусні програми [57, 58].

У межах кваліфікаційної роботи не ставиться за мету виявити причини або детально класифікувати динамічні зміни коду. Проте розглядається механізм цих змін. Динамічна зміна коду відбувається за таким сценарієм: новий код генерується на основі існуючого, змінюючи свою структуру таким чином, щоб уникати стандартних методів аналізу та детекції. Ця властивість ускладнює процеси виявлення загроз і вимагає розробки більш універсальних і динамічних підходів до аналізу.

4.2 Дослідження запропонованих моделей

Розроблена стохастична GERT-модель процесу кібератаки має інноваційні переваги порівняно з традиційними підходами. Її ключова особливість полягає в реалізації двонаправленого методу, який враховує специфіку операційних систем Windows і Linux. Це забезпечується через використання паралельних гілок у моделі, що відображають процеси атаки з урахуванням особливостей кожної системи та передбачають відповідні переходи між етапами. Такий підхід дозволяє моделювати складні сценарії кібератак з високою точністю, враховуючи різноманіття вразливостей та можливих векторів впливу.

Проведемо її перевірку. Дослідимо, яким буде розподіл ймовірностей часу тестування даних корпоративної мережі $\varphi(x)$ при таких параметрах гілок:$\lambda_1 = 0{,}9,\ \lambda_2 = 0{,}15,\ \lambda_3 = 0{,}99,\ \lambda_4 = 0{,}1, \lambda_5 = 0{,}9,\ \lambda_6 = 0{,}99,\ p_1 = 0{,}9,\ p_2 = 0{,}5,\ p_3 = 0{,}5,\ p_4 = 0{,}9,\ p_5 = 0{,}8,\ p_6 = 0{,}9.$

Підставивши ці значення у формулу 2.2, отримаємо значення густини розподілу кібератаки і на основі цих значень побудуємо графік (рис. 4.3).

Точність визначення функції щільності розподілу значною мірою залежить від похибки використовуваної інтерполяційної методики. Наприклад, застосування многочлена Лагранжа третього ступеня дозволяє досягти похибки, яка не перевищує 0,002 [59]. Ця точність є цілком достатньою для вирішення задач, поставлених у межах даної кваліфікайійної роботи.

Такий рівень точності забезпечує адекватне наближення експериментальних даних, що дозволяє побудувати функцію густини з необхідною деталізацією. Застосування інтерполяції третього ступеня є оптимальним вибором з точки зору балансу між обчислювальною складністю та точністю результату, особливо в контексті обмеженого обсягу вхідних даних, характерного для практичних досліджень.

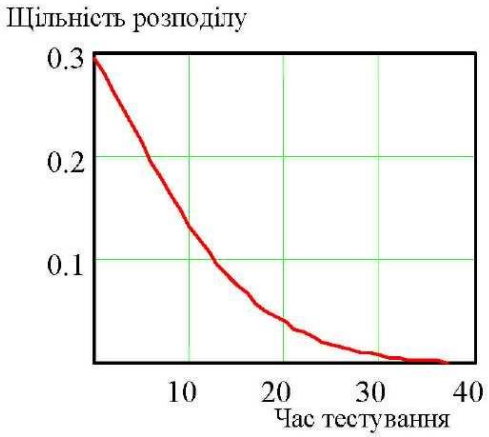


Рисунок 4.3 – Залежність $\varphi(x)$ від часу тестування

Проведемо дослідження математичної моделі процесу діагностики корпоративної мережі. Запропонована математична модель охоплює ключові аспекти безпечного тестування мережі. Функція формування тестів динамічно адаптується до отриманих результатів, забезпечуючи мінімізацію ризиків впливу на систему. Функція реакції моделює змінність відгуку мережі на однакові тестові впливи у різний час. Функція оцінки виконує комплексний аналіз стану безпеки, дозволяючи отримати всебічну картину стійкості системи.

Ключовою особливістю моделі є врахування ймовірнісного характеру реакції мережі на тестові дії, а також впровадження обмежень, що гарантують безпечність процедури тестування. Інтегральний показник визначає момент, коли досягнуто необхідного рівня діагностики безпеки системи [60].

Ця модель формалізує процес тестування безпеки мережі, акцентуючи на збереженні її стабільності під час тестування та врахуванні варіативності поведінки системи у відповідь на тестові впливи. Вона забезпечує ефективне

поєднання гнучкості тестування із суворими вимогами до безпеки мережевої інфраструктури.

В попередніх розділах ми дослідили, що щільність розподілу ймовірностей часу виконання різних алгоритмів системи управління ресурсом мережі обраховується:

$$\phi(\mathrm{x}) = \frac{1}{2\pi i}\int_{-i\infty}^{i\infty} e^{zx}\frac{\mathrm{z}^5 - gz^4 - \mathrm{dz}^3 - cz^2 - bz - a}{\left((\lambda_1 + z)(\lambda_3 + z)\left(\mathrm{z}^5 - rz^4 - \mathrm{hz}^3 - nz^2 - mz - k\right)\right)}dz,$$

Графічне представлення залежності функції розподілу наведено на рисунку 4.4

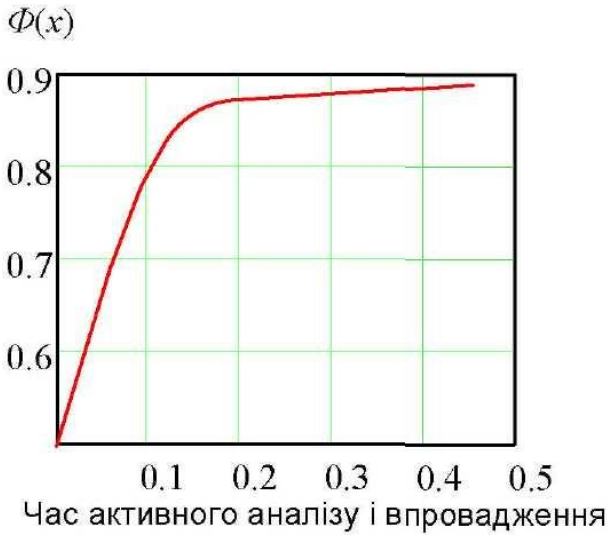
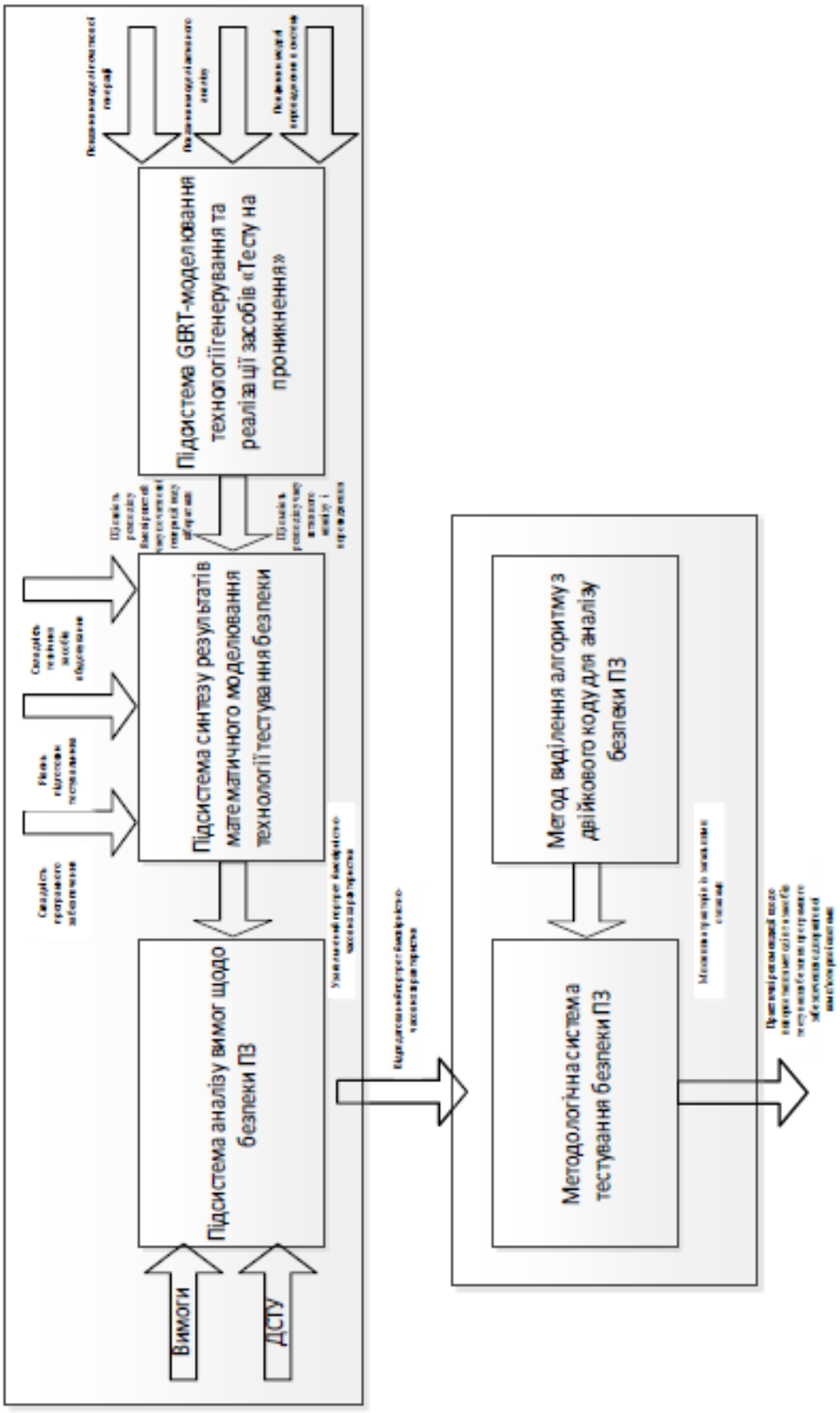


Рисунок 4.4 - Графік залежності функції розподілу

Реалізація методу пошуку алгоритму з двійкового коду для діагностики безпеки корпоративної мережі є складним багатоступеневим процесом.

Процес розпочинається з підготовки спеціалізованого середовища моделювання, де ключову роль відіграє програмний комплекс IxChariot, здатний симулювати як захисні механізми, так і потенційні кібератаки. У середовищі MathCAD створюється математична модель, яка дозволяє абстрагувати та формалізувати процеси мережевої взаємодії (рис. 4.5) [61].



Узагальнена структурна схема імітаційної моделі технології тестування безпеки програмного забезпечення

Рисунок 4.5 – Імітаційна модель процесу тестування

Основний процес тестування включає створення сценаріїв для відправлення та отримання трафіку між кількома кінцевими точками. Для цього використовуються спеціальні узгоджені пристрої, які можуть бути фізичними або віртуальними. Після налаштування всіх параметрів запускається тест, під час якого IxChariot збирає дані про продуктивність мережі, а також про будь-які проблеми, які можуть виникнути в процесі передачі даних.

Після завершення тестування користувач може переглянути звіти, що містять детальну інформацію про тестові сценарії, статистику мережі та різні аналітичні метрики. У цих звітах можна побачити графіки продуктивності, порівняння результатів для різних тестів, а також виявити вузькі місця у мережі, що можуть потребувати оптимізації. Використовуючи ці дані, можна здійснити точну оцінку ефективності мережі та прийняти рішення щодо подальших кроків для її поліпшення.

У процесі тестування використовувалися програми з відкритим кодом для отримання результатів аналізу даних із застосуванням як удосконаленого, так і відомого методу. Існує кілька загальновизнаних методів тестування, які використовуються в різних галузях для перевірки якості програмного забезпечення, мережевих систем або апаратного забезпечення. Результати, представлені на рисунку, демонструють, що удосконалений метод забезпечує приблизно на 4% вищу ефективність за часом у порівнянні з відомим підходом. Це свідчить про підвищення продуктивності та оптимізацію роботи за рахунок запропонованих покращень (рис. 4.6).

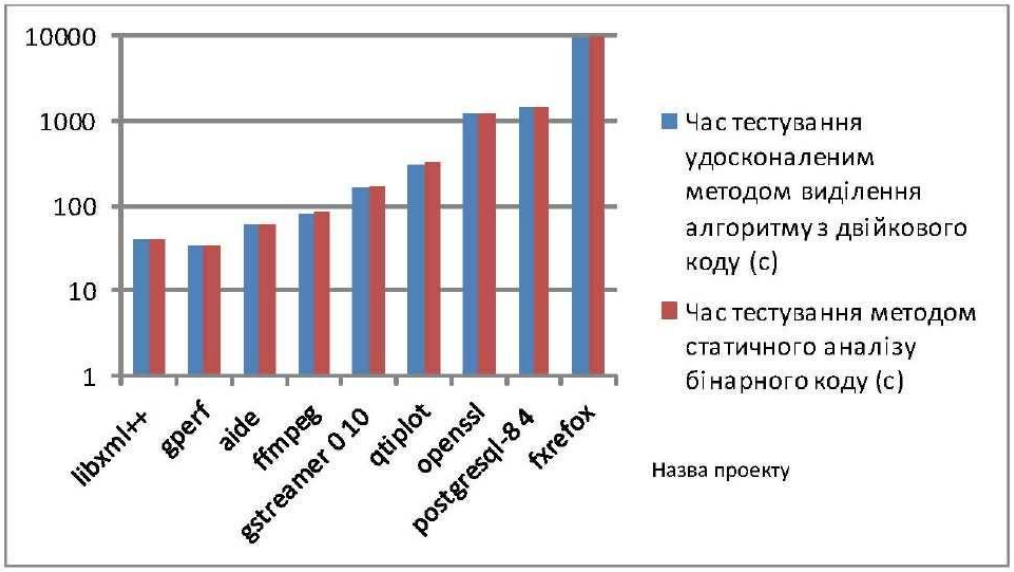


Рисунок 4.6 – Графік порівняльного аналізу часу тестування

Було виконано порівняльний аналіз рівня безпеки між удосконаленим та існуючим методом. Результати свідчать, що удосконалений підхід дозволив досягти зростання рівня захисту даних на 3%. (рис. 4.7)

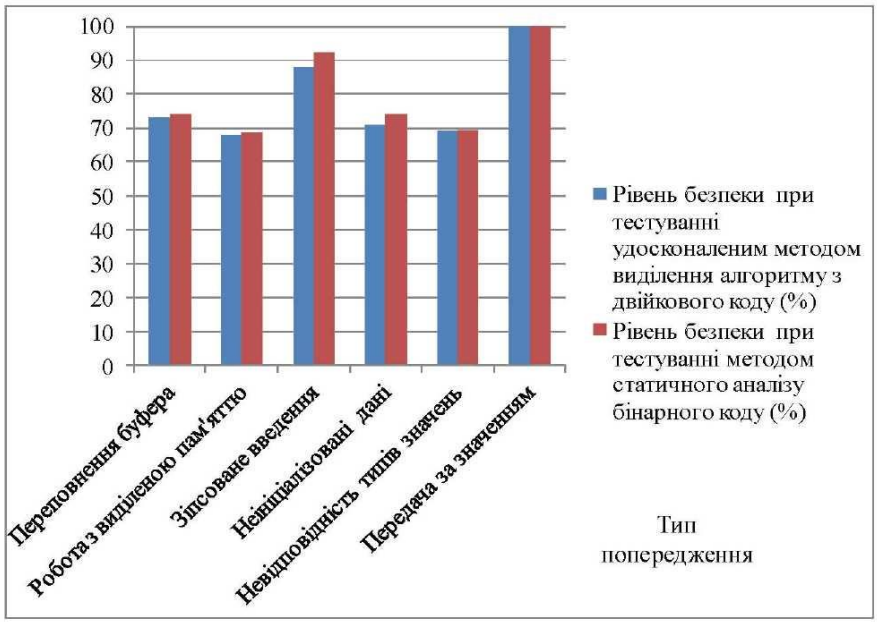


Рисунок 4.6 – Графік порівняльного аналізу рівня безпеки

Це покращення забезпечується за рахунок оптимізації алгоритмів обробки та впровадження додаткових механізмів протидії загрозам, що підвищує загальну стійкість системи до потенційних атак.

4.3 Висновки

У цьому розділі розглянуто методологію отримання набору векторів, що взаємопов'язані, і які всебічно описують алгоритмічну поведінку системи через альтернативні сценарії її виконання. Такий підхід дозволяє поступово відновлювати внутрішню логіку алгоритму, перетворюючи низькорівневий двійковий код на більш структуровану послідовність логічних операцій і переходів.

Моделювання складних сценаріїв кібератак було виконано з високою точністю, з урахуванням різноманіття вразливостей і можливих векторів впливу. Такий рівень деталізації забезпечує точне наближення до реальних експериментальних даних, що дозволяє побудувати функцію густини з необхідною точністю. Застосування інтерполяції третього ступеня було оптимальним рішенням, оскільки воно поєднує високу точність та ефективність з точки зору обчислювальних ресурсів, що особливо важливо для практичних досліджень з обмеженим обсягом вхідних даних.

Для перевірки запропонованого методу пошуку алгоритму з двійкового коду для діагностики безпеки корпоративної мережі було проведено імітаційне моделювання в середовищі MathCAD, використовуючи спеціальну програму IxChariot, яка одночасно може виконувати захист і генерувати кібератаки.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне наукове завдання, пов’язане з розробленням методу тестування безпеки даних, спрямованого на підвищення рівня захисту інформації в корпоративних мережах.

Проведений аналіз основних вимог до захисту даних корпоративної мережі виявив, що існуючі методики тестування безпеки не повністю забезпечують необхідний рівень захисту. Це дозволило обґрунтувати вибір математичних моделей і сформулювати наукове завдання, яке стало основою дослідження в магістерській роботі.

Було розроблено математичну модель кібератаки, яка охоплює ключові етапи атак на корпоративні мережі. Це забезпечує можливість глибокого аналізу процесу атак і формування рекомендацій для їх ефективного запобігання.

У дослідженні впроваджено інноваційний методологічний підхід до розкриття внутрішньої структури програмних компонентів через деталізований алгоритмічний аналіз двійкового коду в контексті мережевої безпеки.

Ключовим етапом запропонованої методики є перетворення інформаційного масиву у машинно-незалежну форму. Це дозволяє уникнути впливу специфічних архітектурних характеристик вихідного коду, забезпечуючи універсальність аналізу. На цьому етапі виділяється основний алгоритмічний контур, що є об’єктом подальшого дослідження.

Етап оптимізації спрямований на скорочення та систематизацію отриманих даних. Він включає їх очистку від зайвих елементів і приведення до форми, що є найбільш інформативною для експертів із кібербезпеки. Додатково забезпечується підготовка інформації для інтеграції в системи штучного інтелекту, що дозволяє автоматизувати процес подальшої обробки.

Запропонована методика також підтримує ітераційний підхід до аналізу, коли кожна нова ітерація доповнює й уточнює початкові гіпотези. Інтелектуальна система аналізує результати кожного циклу, приймаючи рішення щодо

необхідності подальшої деталізації або завершення процесу, що гарантує адаптивність і ефективність роботи методики.

Унікальність запропонованого підходу полягає у синергетичному поєднанні графового моделювання, універсального машинно-незалежного перетворення даних та інтелектуальної системи прийняття рішень. Ця інтеграція створює цілісний методологічний алгоритм, що забезпечує комплексну діагностику можливих загроз для корпоративних інформаційних мереж. Такий підхід дозволяє не лише моделювати потенційні загрози з високою точністю, а й забезпечувати адаптивний аналіз у реальному часі, враховуючи динаміку змін у мережі та варіативність атак.

Таким чином, запропонований метод забезпечує комплексний підхід до діагностики безпеки корпоративної мережі, поєднуючи можливості імітаційного моделювання, низькорівневого аналізу двійкового коду та математичної формалізації мережевих процесів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

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

2. 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. 580с.

3. Інформаційна безпека держави: навчальний посібник/ В.І. Гур'єв, Д.Б. Мехед, Ю.М. Ткач, І.В. Фірсова. – Ніжин: ФОП Лук'яненко В.В. ТПК «Орхідея», 2018. – 166 с.

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

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

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

7. 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.

8. 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.

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

10. Корченко О.Г., Бурячок В.Л., Гнатюк С.О. (2022) "Методологія тестування безпеки інформаційних систем". Київ: НАУ. ISBN: 978-966-598-654-3

11. Kim Peter. The Hacker Playbook 3: Practical Guide To Penetration Testing / Securety planet LLC 2018 359 p.

12. Основи інформаційної безпеки. Конспект лекцій/ Б.А. Заплотинський. – КІІВіП НУ “ОЮА”, 2017. – 128 с.

13. Корченко О.Г., Бурячок В.Л., Гнатюк С.О. (2022) "Методологія тестування безпеки інформаційних систем". Київ: НАУ. ISBN: 978-966-598-654-3

14. Miessler, D. (2023) "Professional Penetration Testing: Creating and Learning in a Hacking Lab", 3rd Edition. Syngress. ISBN: 978-0128242827

15. Engebretson, P. (2023) "The Basics of Hacking and Penetration Testing", 3rd Edition. Syngress. ISBN: 978-0128196342

16. Herzog, P. (2010) "OSSTMM 3 – The Open Source Security Testing Methodology Manual". ISECOM, New York. ISBN: 978-1393667681

17. OWASP Foundation (2023) "OWASP Testing Guide v4". Web Application Security Testing Methodology. URL: https://owasp.org/www-project-web-security-testing-guide/

18. National Institute of Standards and Technology (2020) "Special Publication 800-115: Technical Guide to Information Security Testing and Assessment". NIST, Washington. DOI: 10.6028/NIST.SP.800-115

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

20. Rathore B., Brunner M., Dilaj M. (2019) "Information Systems Security Assessment Framework (ISSAF)". Open Information Systems Security Group. ISBN: 978-1119512943

21. EC-Council (2023) "Certified Ethical Hacker (CEH) Methodology v11". EC-Council Press. ISBN: 978-1133283485

22. Snort. "Network Intrusion Detection and Prevention System". [Електронний ресурс]. Режим доступу: https://www.snort.org/

23. Burp Suite. "Vulnerability scanning and advanced manual tools". [Електронний ресурс]. Режим доступу: https://portswigger.net/burp

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

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

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

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

28. OWASP Testing Guide v4. [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/ OWASP_Testing_Project

29. Common Vulnerabilities and Exposures. [Електронний ресурс]. – Режим доступу: https://cve.mitre.org/

30. National Vulnerability Database. [Електронний ресурс]. – Режим доступу: https://www.nist.gov/programs-projects/national-vulnerability-database-nvd

31. Anderson, R. (2020) "Security Engineering: A Guide to Building Dependable Distributed Systems", 3rd Edition. Wiley. ISBN: 978-1119642787

32. Schneier, B. (2022) "Click Here to Kill Everybody: Security and Survival in a Hyper-connected World". W. W. Norton & Company. ISBN: 978-0393357448

33. Scarfone, K., Souppaya, M., Cody, A., Orebaugh, A. (2021) "Technical Guide to Information Security Testing". NIST Special Publication. DOI: 10.6028/NIST.SP.800-115r1

34. Miessler, D. (2023) "Professional Penetration Testing: Creating and Learning in a Hacking Lab", 3rd Edition. Syngress. ISBN: 978-0128242827

35. Alhusain S. Intelligent Data-Driven Reverse Engineering of Software Design Patterns. A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy. Montfort University, 2017. 220p.

36. Information Systems Security Assessment Framework (ISSAF).

[Електронний ресурс]. – Режим доступу: http://www.oissg.org/files/issaf0.2.1.pdf

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

38. Franck De Goër de Herve. Reverse-engineering of binaries in a single execution. Université Grenoble Alpes, 2017. 252p.

39. Krishnan M. Soumya Software Development Risk Aspects and Success Frequency on Spiral and Agile Model / M. Soumya Krishnan // International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization). -2015. -Vol.3. -№1. -pp.301-310.

40. Прикладне застосування теорії хаотичних систем у телекомунікаціях: монографія / [Ю.Я. Бобало, С.Д. Галюк, М.М. Климаш, Р.Л. Політанський]; Нац. ун-т «Львів. політехніка». – Львів: Коло, 2015. – 178 с.

41. Стефінко Я.Я., Піскозуб А.З. Використання відкритих операційних систем для тестування на проникнення в навчальних цілях/Я.Я. Стефінко, А.З. Піскозуб //Вісник Національного університету “Львівська політехніка” Комп’ютерні системи та мережі. – 2014. - № 806. – С. 258-263.

42. The Penetration Testing Execution Standard (PTES). [Електронний ресурс]. – Режим доступу: http://www.penteststandard. org/index.php/Main_Page

43. Zhang Peng. Security in Network Coding / Peng Zhang, Chuang Lin. - Springer, 2016.-98 p.

44. Зацепін К.О. Використання моделі GPT для автоматизації тестування iot-пристроїв / І.В. Муляр, В.С. Гловюк, К.О. Зацепін, В.С. Чернов // Збірник тез доповідей ХХміжнародної науково-практичноїконференції «Військова освіта і наука:сьогодення та майбутнє», 29листопада 2024 р. - Київ : ВІКНУ,2024. – Т. 1. – С. 49-50

45. Tenable. "Tenable.io Vulnerability Management". [Електронний ресурс]. Режим доступу: https://www.tenable.com/products/tenable-io

46. Зацепін К.О. Модель інформаційної безпеки функціонування програмного забезпечення / Сергій Лєнков, Володимир Джулі, Олександр Яворський, Костянтин Зацепін //SMART TECHNOLOGIES: Industrial and Civil

Engineering, Issue 1(14), 2024, 18-34

47. Дудикевич В.Б., Микитин Г.В. (2023) "Технології аналізу безпеки програмного забезпечення". Львів: Видавництво Львівської політехніки. ISBN: 978-966-941-321-0

48. Information Systems Security Assessment Framework (ISSAF). [Електронний ресурс]. – Режим доступу: http://www.oissg.org/files/issaf0.2.1.pdf (дата звернення: 21.09.2020)

49. IxChariot: Тестування в умовах атак відмови в обслуговуванні (DOS) [Електронний ресурс]. - Режим доступу до ресурсу: http://ixchariot.ru/tests/.

50. Allen, L. (2022) "Advanced Penetration Testing: Hacking the World's Most Secure Networks". Wiley. ISBN: 978-1119367680

51. Методи і алгоритми захисту інформаційних ресурсів комп'ютерних систем: навчальний посібник / В. М. Джулій, Ю. П. Кльоц, І. В. Муляр, В. М. Чешун. Хмельницький: ХмНУ, 2020. 196с.

52. Darktrace. "Self-Learning AI for Cyber Defense". [Електронний ресурс]. Режим доступу: https://www.darktrace.com/en

53. Dang B., Gazet A., Bachaalany E. Practical Reverse Engineering: x86, x64, ARM, Windows. Indianapolis : John Wiley & Sons, Inc. , 2014. 383p.

54. Practical Information Security: A Competency-Based Education Course / [Izzat Alsmadi, Robert Burdwell, Ahmed Aleroud, Abdallah Wahbeh, Mahmoud Ali Al-Qudah, Ahmad Al-Omari]. Cham, Switzerland : Springer International Publishing AG, 2018. 328 p.

55. Selander D. Advanced Apple Debugging and Reverse Engineering. Razeware LLC, 2017. 475p.

56. Sreeram Reddy, M Manzoor Hussain, K Srinivasa Rao. Latest Research on Reverse Engineering Technology. Proceedings of the International conference on Paradigms in Engineering & Technology (ICPET2016). Methodist College of Engineering & Technology, Hyderabad. P. 945-948.

57. Найкращі програми для реверс-інжинірингу. [Електронний ресурс]. Режим доступу: https://spy-soft.net/reverse-engineering- software/

58. Reverse engineering approach for improving the quality of mobile applications. / Eman K. Elsayed, Kamal A., Enas E. Naglaa E. Research article Software Engineering. August 19, 2019. P. 123.

59. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навчальний посібник/ Д.В. Ланде, І.Ю. Субач, Ю.Є. Бояринова. - К.: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2018. - 297 с.

60. Моделювання систем захисту інформації/ А.О. Антонюк. - Ірпінь: Національний університет ДПС України, 2015. - 273 с.

61. Системний аналіз та прийняття рішень в інформаційній безпеці: підручник. / В.Л. Бурячок, С.В.Толюпа, А.О. Аносов, В.А.Козачок, Н.В. Лукова-Чуйко / - К.:ДУТ, 2015. - 345 с.

ДОДАТОК А

(Обов’язковий)

Перелік публікацій за темою кваліфікаційної роботи

Модель інформаційної безпеки функціонування програмного забезпечення

Сергій Ленков[1], Володимир Джулій[2], Олександр Яворський[3], Костянтин Зацепін[4]

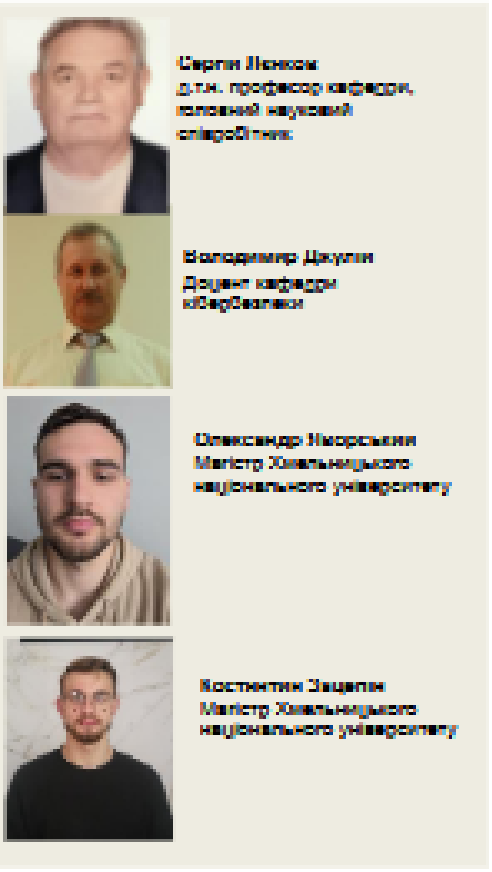
[1] Військовий Інститут Київського національного університету імені Тараса Шевченка
Юлії Здановської, 81, Київ, Україна, 03189
[1] lenkov_s@ukr.net, https://orcid.org/0000-0001-7689-239X,
[2,3,4] Хмельницький національний університет
вул. Інститутська, 11, Хмельницький, Україна, 29000
[2,3,4] dzhulliym@khmnu.edu.ua, https://orcid.org/0000-0003-1878-4301

Received 29.06.2024, accepted 24.10.2024
https://doi.org/10.32347/uwt.2024.15.1202

Анотація. В роботі виконано систематизацію моделей надійного та безпечного функціонування програмного забезпечення. В результаті проведеного дослідження виділено три типи моделей: аналітичні; статистичні; емпіричні [4, 9].

Розглянуто ряд найчастіше застосовуваних моделей, виділено їх недоліки та переваги з погляду розв'язуваної задачі опису безпечного функціонування програмного продукту, та розпізнавання шкідливого програмного забезпечення. За результатами проведених досліджень розглянуті моделі мають переваги у плані простоти їх практичної реалізації, проте водночас виділено наступні недоліки: деякі з розглянутих моделей при реалізації вимагають великого об'єму обчислювальних ресурсів – для аналізу безпеки та забезпечення архівних даних; використання статистичних та імовірнісних моделями припущень про те, що інтенсивність атак/відмов чи кількість помилок у програмному забезпеченні мають задану/сягнуть відомий розподіл (біноміальний, стандартний або пуассонівський), що не завжди вірно для реальних процесів і систем; немає поділу на відмови програмного забезпечення і вихід з ладу внаслідок кібератак, не враховуються також вразливості нульового дня; не аналізуються загрози досліджуваного програмного забезпечення до пам'яті, що могло б дати важливу інформацію про його легітимність чи наявність шкідливих функцій; жодна з розглянутих моделей не забезпечує комплексного представлення про процес функціонування програмного забезпечення, у тому числі, аналіз зі сторони інформаційної безпеки відсутній [1, 8, 16].

Завдання розпізнавання шкідливого програмного забезпечення з кожним роком стає дедалі актуальнішим і складнішим у зв'язку з



Сергій Ленков
д.т.н., професор кафедри, головний науковий співробітник

Володимир Джулій
Доцент кафедри кібербезпеки

Олександр Яворський
Магістр Хмельницького національного університету

Костянтин Зацепін
Магістр Хмельницького національного університету

цифровізацією галузей діяльності людини та використанням програмного забезпечення для виконання бізнес-логіки та технічних процесів у складних системах. Внаслідок цього, чим більший обсяг програмного забезпечення в системі, тим потенційно більше в ньому помилок, при цьому через підключення сучасних систем до мережі Інтернет, програмне

забезпечення часто поширюється по мережі, що дозволяє зловмисникам створити нові вектори кібератак на системи [2, 3, 5, 7].

Запропонована модель усуває наведені недоліки за рахунок того, що вона враховує характерні особливості прояву шкідливого програмного забезпечення на пристроях, а саме вплив шкідливого програмного забезпечення на обчислювальні ресурси системи та роботу з оперативною пам'яттю. Це дозволяє розробленій моделі враховувати як надійність функціонування програмного забезпечення, так і безпеку [9, 11, 19].

У термінах моделі сформульовані критерії безпечного функціонування програмного забезпечення, зроблено висновок про те, що для найбільш ефективної реалізації такої моделі на практиці має бути використаний гіпервізор [14, 17, 20].

Ключові слова: модель, інформаційна безпека, вразливості, атаки, конфіденційні дані, шкідливе програмне забезпечення, безпечне функціонування.

ВСТУП

Число кібератак на найважливіші галузі інфраструктури у всьому світі неухильно зростає з кожним роком. Так, за даними компанії Trellix, вже в першому кварталі 2023 року в Україні було зафіксовано на 22,5% більше кібератак, ніж у останньому кварталі 2022 року. У другому кварталі кількість кібератак зросла ще на 9%, порівняно з першим кварталом 2023 року [5, 7].

Одним із найпоширеніших методів реалізації кібератак є використання шкідливого програмного забезпечення. Так, на початку 2023 року кількість кібератак, реалізованих цим методом, незначно поступилася лише кібератакам методами соціальної інженерії. Частка кібератак з використанням шкідливого програмного забезпечення, за даними Trellix, склала 71% для фізичних осіб та 60% для юридичних осіб [5, 7, 22].

У зв'язку з цим, актуальність завдання розпізнавання шкідливого програмного забезпечення щорічно зростає. Слід також враховувати, що зловмисники при реалізації кібератак комбінують різні типи шкідливого програмного забезпечення – використовують багатофункціональні трояни або завантажують на скомпрометовані об'єкти інфраструктури безліч різних за функціоналом шкідливих програм. Це значно ускладнює процес аналізу безпеки програмного забезпечення. Особливої гостроти проблема набуває останнім часом у зв'язку з об'єктивною необхідністю організації віддаленої роботи та доступу до інформаційних ресурсів компаній, систем управління та моніторингу технологічних процесів – що потребує вирішення завдання захисту «територіально розподіленого периметра». Зростає небезпека атак, що використовують прийоми як явно вираженої, так і прихованої соціальної інженерії, коли для впровадження шкідливих програм застосовуються довірені джерела [4, 6, 7, 20].

Одним із каналів доставки шкідливих програм можуть виступати комплекти та компоненти легітимного програмного забезпечення. У разі відкритого коду користувачам надається можливість ознайомитися з вихідними текстами, та за необхідності виконати оцінку та аналіз як ефективності продукту, так і його безпеки. У разі програмного забезпечення із закритим кодом вихідні тексти або не надаються взагалі, або надаються частково або повністю державним сертифікаційним органам для винесення рішення про допуск до використання таких продуктів у певних мережах та системах. Більшість ліцензійних угод на використання програмного забезпечення містять повідомлення користувача про обмеженість гарантії та відмову від відповідальності при використанні продукту. Таким чином, програмний продукт без вихідного коду є «чорною» скринькою для користувача, і навіть у разі авторства відомого та сумлінного розробника та постачальника може містити: недокументовані можливості; шкідливі компоненти, впроваджені засоби розробки внаслідок атак на сховища; шкідливе програмне забезпечення і «чорні ходи», впроваджені на етапі розробки, компіляції програмного коду або поширення; помилки та недоробки, не

усунені у процесі тестування; шкідливе програмне забезпечення та «чорні ходи», навмисно впроваджені третіми особами у процесі доставки [9, 10, 15].

Таким чином, завдання оцінки безпеки програмного забезпечення не вирішується довірою до програмного продукту, який розроблений відомою компанією, сертифікований та отриманий із джерел, які видаються довіреними.

АНАЛІЗ ОСТАННІХ ДОСЛІДЖЕНЬ ТА ПОСТАНОВКА ЗАДАЧІ

Виявити потенційні загрози можна на підставі аналізу вихідних текстів та динамічного аналізу поведінки у контрольованому середовищі. Однак в умовах відсутності вихідних текстів потрібне використання ще більш технічно складних процедур, у тому числі декомпіляції, реверсі інжинірингу і т.д. На практиці фахівцю із захисту інформації потрібно швидко і ефективно визначити можливість використання програмного забезпечення в інфраструктурі, що захищається, без порушення інформаційної безпеки. При цьому час та обчислювальні ресурси обмежені, а повна автоматизація процесу аналізу безпеки неможлива у зв'язку з високою складністю предметної галузі [6, 17, 18].

На даний час не існує універсального та оптимального вирішення задачі виявлення шкідливого програмного забезпечення. Використання всього спектра наявних засобів дозволяє максимально наблизитися до вирішення завдання виявлення шкідливого коду, визначення його параметрів та поведінки, і навіть ідентифікації методів, а можливо навіть особистості зловмисника, проте вимагає докладання значних зусиль, тимчасових витрат і кваліфікації [5, 7].

Для підвищення ефективності вирішення цього завдання в умовах обмежених часових та обчислювальних ресурсів, а також відсутності відкритих вихідних текстів програмного забезпечення, пропонується підхід, що полягає в автоматизації оцінки інформаційної безпеки програмного забезпечення без вихідних текстів та базується на моделі безпечного функціонування ПЗ та методики оцінки інформаційної безпеки [14, 16].

Безпека ПЗ означає, що в результаті його включення в систему не відбувається порушення стану безпеки, тобто повною мірою зберігаються функціональні характеристики системи, не змінюються регламентовані взаємодії суб'єктів та об'єктів системи. Безпека також означає відсутність дефектів недокументованих можливостей, які можуть спричинити порушення безпеки як власне аналізованого ПЗ, так і системи, в яку воно встановлюється, у тому числі при взаємодії з іншими компонентами системи. Безпека ПЗ також передбачає відсутність у його складі шкідливого коду [4, 10, 20].

Характеристики шкідливого програмного забезпечення наведені в таблиці 1.

Наведена характеристика порівняно умовна, оскільки шкідливе програмне забезпечення, що використовується для здійснення кібератак може мати комбінований функціонал. Найбільш небезпечними є засоби, націлені на приховану присутність у системі з метою викрадення інформації та перехоплення управління [5, 7, 8].

Регламентовані стандартом заходи та процедури також можуть бути використані для побудови системи оцінки безпеки програмного забезпечення, що пропонується сторонніми розробниками. Однак, безпосереднє застосування методик оцінки, побудованих на аналізі вихідного коду, не може бути ефективно здійснено у разі його відсутності. Більша частина програмного забезпечення для систем Windows, що пропонується користувачам, не постачається з вихідними текстами [2, 9, 13].

Таблиця 1. Характеристики шкідливого програмного забезпечення

Table 1. Characteristics of malware

Класи шкідливого програмного забезпечення, середовища оперування, область дії та поширення		Шпигунське	Рекламне	Куки файли	"Чорний хід"	Троянець	Сніфери	Спам	Мережеві боти	Логічні бомби	Черв'яки	Віруси
Технології створення ШПЗ	Патерни	+	+	+	+	+	+	+	+	+	+	+
	Обфузування	+	+	+	+	+	+	+	+	+	+	+
	Поліморфізм	+	+	+	+	+	+	+	+	+	+	+
	Спеціальні інструменти	+	+	+	+	+	+	+	+	+	+	+
Середовище виконання	Мережа	+	+	+	+	-	+	+	+	+	+	-
	Віддалене виконання	+	+	+	+	+	+	+	+	-	-	-
	Робоча станція	-	-	-	-	-	-	-	-	+	+	+
Засоби поширення, середовище доставки	Мережа	+	+	+	+	+	+	+	+	+	+	+
	Змінні диски	+	+	+	+	+	+	+	+	+	+	+
	Завантажувальні файли	+	+	+	+	+	+	+	+	+	+	+
Руйнівний вплив	Порушення конфіденційності	+	-	+	-	+	+	-	-	-	-	-
	Відволікання користувачів	-	+	-	-	-	-	+	-	-	-	-
	Відмова в обслуговуванні	-	-	-	+	-	-	+	+	+	+	+

На етапі впровадження програмного забезпечення, завдання оцінки безпеки стоїть дуже гостро, і має вирішуватися експлуатаційним персоналом достатньо ефективно. В даний час немає єдиного методичного документа, який дозволяв би при його застосуванні персоналом без спеціальних знань приймати рішення про можливість застосування ПЗ у корпоративних мережах. Натомість існує великий інструментарій, безліч методів та підходів для дослідження поведінки ПЗ, шкідливий характер якого відомий, і завданням дослідника є визначення класу шкідливого ПЗ, характерних ознак та поведінки для подальшого використання в системах захисту [8, 20].

Об'єктивні причини появи вразливостей у ПЗ полягають у надзвичайно високій структурній складності програмного коду, динамічності розвитку версій, що випускаються розробником, та легкості самодифікації програм при віддаленому оновленні. До цього можна додати проблему достовірної ідентифікації навмисно створених програмних закладок, недосконаліть нормативно-методичної та відставання інструментальної бази сертифікаційних випробувань [6, 16, 20].

Вразливі сигнатури у вихідних текстах ПЗ шукаються розробниками лише на етапах внутрішнього тестування, основною метою якого є виявлення власних помилок у розробленому програмному забезпеченню. При цьому про оцінку безпеки ПЗ найчастіше не замислюються [6, 12, 22].

Статичний аналіз вихідних текстів здійснюється без реального виконання досліджуваних програм. Залежно від інструмента, що використовується, глибина аналізу може змінюватись від визначення поведінки окремих операторів до дослідження, що включає весь наявний вихідний текст.

Динамічний аналіз вихідних текстів виконується за допомогою запуску програм на реальному чи віртуальному процесорі. Утиліти динамічного аналізу можуть вимагати завантаження спеціальних бібліотек, перекомпіляцію програмного коду. Для більшої ефективності динамічного аналізу потрібно подання тестованій програмі достатньої кількості вхідних даних, щоб отримати повніше покриття коду.

Статичний та динамічний аналізи дозволяють виконати наступні види контролю:

1. Контроль повноти та відсутності надмірності вихідних текстів. Він здійснюється експертом за допомогою програмних засобів, які виявляють можливо надлишкові функціональні об'єкти, що не використовуються.

2. Контроль відповідності вихідних текстів його об'єктному коду, полягає в компіляції вихідних текстів програм і порівнянні отриманих об'єктних (завантажувальних) кодів з аналогічними, наданими з дистрибутивом ПЗ, що перевіряється. В якості програмних засобів для здійснення цієї перевірки використовуються компілятор та програма підрахунку контрольних сум та порівняння файлів.

3. Контролює зв'язки функціональних об'єктів з управління. Ця перевірка полягає у побудові графа взаємодії функціональних об'єктів з управління (граф виклику функцій). Перевірка може бути виконана експертом без залучення додаткових програмних засобів, проте обсяг вихідних текстів та кількість функціональних об'єктів у сучасному програмному забезпечення досить велика.

4. Контроль зв'язків функціональних об'єктів за інформацією побудові графа взаємодії функціональних об'єктів за інформацією (граф передачі змінних між функціями та використання глобальних змінних усередині функцій).

5. Контроль інформаційних об'єктів передбачає побудову переліку інформаційних об'єктів (усі змінні, крім локальних, що не передаються інші функціональні об'єкти).

6. Контроль наявності заданих конструкцій у вихідних текстах полягає у пошуку певних конструкцій у вихідних текстах ПЗ. Пошук здійснюється за базою вразливостей, що містить сигнатури чи відмітні ознаки вразливості.

При контролі виконання функціональних об'єктів відбувається зіставлення фактичних маршрутів їх виконання та маршрутів, побудованих у процесі проведення статичного аналізу [5, 7, 15].

Лексичний аналіз передбачає пошук розпізнавання та класифікацію різних лексем об'єкта дослідження (програмного забезпечення), представленого у виконуваних кодах. У цьому лексемами є сигнатури. В даному випадку здійснюється пошук сигнатур наступних класів: сигнатури вірусів; сигнатури елементів; сигнатури (лексеми) «підозрілих функцій»; сигнатури штатних процедур використання системних ресурсів та зовнішніх пристроїв. Пошук лексем (сигнатур) реалізується за допомогою спеціальних інструментів – сканерів [10, 14, 16, 22].

Синтаксичний верифікаційний аналіз передбачає пошук, розпізнавання та класифікацію синтаксичних структур, а також побудова структурно-алгоритмічної моделі самого продукту [11]. Семантичний аналіз передбачає дослідження функцій (процедур) продукту аспекті операційної середовища інформаційної системи. На відміну від попередніх видів аналізу, заснованих на статичному дослідженні, семантичний аналіз націлений на вивчення динаміки продукту - його взаємодії з довкіллям. Процес дослідження здійснюється у віртуальному операційному середовищі з повним контролем його дій та відстеженням алгоритму його роботи з структурно-алгоритмічної моделі.

Семантичний аналіз є найбільш ефективним видом аналізу, але при цьому найтрудомісткішим. Узагальнена класифікація методів аналізу програмного забезпечення представлена на рис. 1.

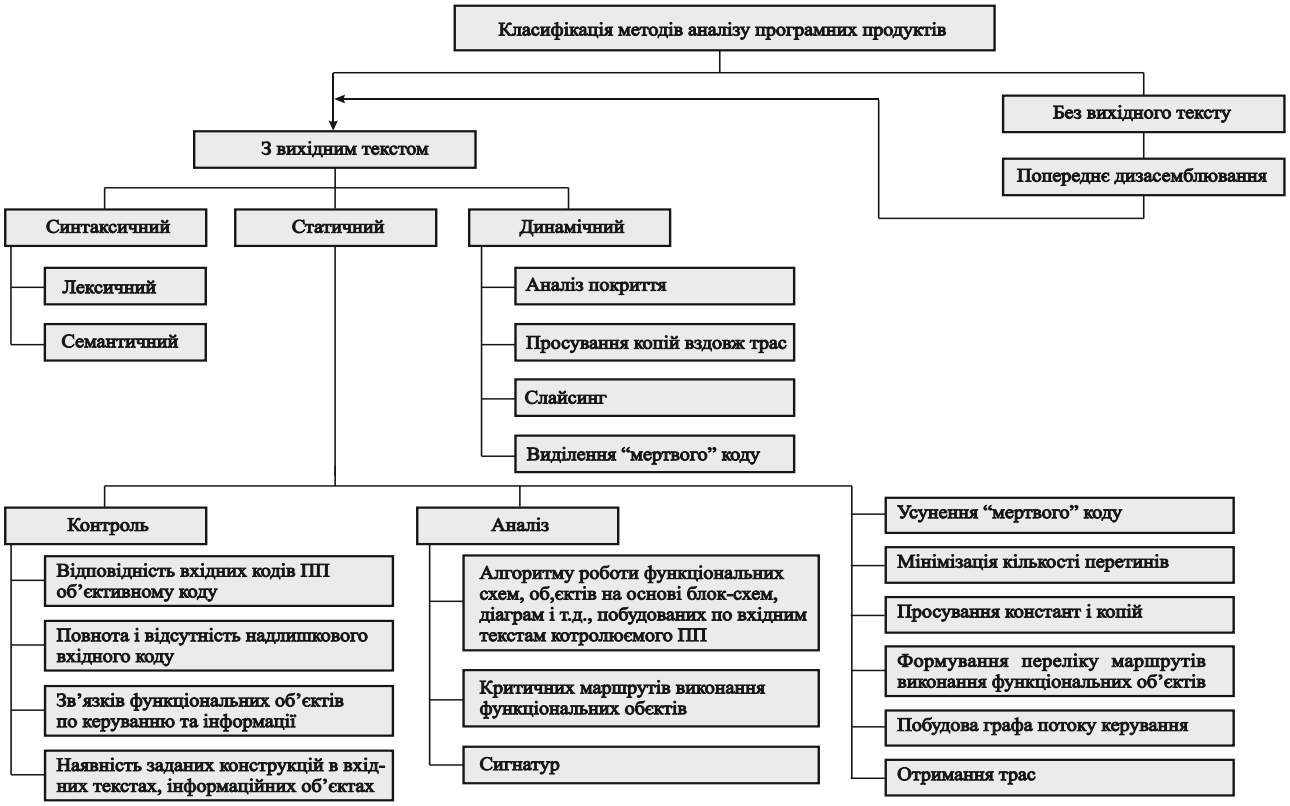


Рис. 1. Узагальнена класифікація методів аналізу програмного забезпечення

Fig. 1. Generalized classification of software analysis methods

Зі зростанням складності ПЗ динамічний аналіз стає нерозв'язним завданням і перетворюється на формальну процедуру. Багато програмних комплексів, які використовуються експертами у повсякденній роботі, мають списки потенційно небезпечних конструкцій, що робить неефективними методи сигнатурного аналізу. Не згадується сигнатурний аналіз у вимогах до випробувань ПЗ нижче другого рівня контролю (тобто, програм, що обробляють секретну та конфіденційну інформацію) [13]. Немає механізмів виявлення помилок кодування, пов'язаних із переповненням буфера, викликом функцій із чужого адресного простору, відсутнє очищення пам'яті тощо. Також відсутні вимоги щодо побудови переліку маршрутів під час виконання функціональних гілок програми, немає механізмів визначення повноти покриття коду та його достатності при динамічному аналізі.

Дослідження вітчизняних та зарубіжних джерел дозволило виділити такі ключові методи, способи та підходи до оцінки інформаційної безпеки ПЗ без вихідних текстів:

1. Перевагою статичного аналізу є повне покриття коду, на відміну методів і способів динамічного аналізу. Виконуючи бінарний аналіз програмного забезпечення, що розробляється, є штатним для середовища функціонування, можна виконати перевірку сторонніх бібліотек, які були використані при розробці. Також можна виконати контрольну перевірку фінальної версії програмного забезпечення шляхом порівняння результатів аналізу вихідного коду та виконуваного коду. Також статичний аналіз може бути успішно використаний при дослідженні стороннього ПЗ, що отримується від постачальників або завантажується з Інтернету.

2. Динамічний аналіз із використанням «пісочниці». Такий вид аналізу являє собою підхід до виявлення підозрілої поведінки у віртуальному ізольованому середовищі. Важливо, що налаштування віртуальних середовищ може бути виконане відповідно до одного з двох підходів.

Оптимальний тут підхід, при якому запуск зразка програмного забезпечення виконується в такому віртуальному середовищі, яке повністю відповідає цільовій системі, якій призначений даний зразок. Однак у такому разі «пісочниця» повинна «на льоту» визначати цільове середовище вузла мережі та запускати потрібну віртуальну машину.

3. Динамічний аналіз із використанням відладчика. Відладники реалізують покрокове виконання програм та встановлення в них точок зупинки. Це може бути реалізовано лише на рівні центрального процесора, мікроконтролера, операційної системи. Ключові функції відладчика – запуск програми, встановлення точок її зупинки, відображення даних, що використовуються програмою, та внесення змін до програми, щоб перевірити, чи можливо таким чином усунути виявлені помилки.

4. Аналіз на імітаційному стенді із використанням форензики. Інтерактивне тестування безпеки додатків є методикою аналізу безпеки ПЗ, переважно спрямовану на дослідження поведінки веб-додатків під час їх роботи [14]. Рішення з інтерактивного тестування зазвичай працюють за рахунок розгортання спеціальних агентів у додатку, що працює. Ці агенти безперервно аналізують те, як взаємодіють додатки (зазвичай ініціювати автоматизованими тестами), щоб виявити вразливості. Деякі рішення щодо інтерактивного тестування безпеки можуть не тільки активно відстежувати вразливості (наприклад, SQL-ін'єкції), але й перевіряти їх, демонструючи їхню придатність для використання зловмисниками. Для аналізу безпеки в такому підході часто використовуються методи форензики, тобто всі методи, пов'язані з розслідуванням вже скоєних комп'ютерних інцидентів.

Таким чином, можна відзначити, що динамічний аналіз більш популярний, ніж статичний, при оцінці безпеки ПЗ без вихідних текстів, однак найкращий результат досягається при їх сукупному використанні, оскільки саме статичний аналіз з використанням дизассемблера дозволяє отримати повне покриття коду та усунути ключовий недолік, властивий методів динамічного аналізу [4, 10, 15].

Слід зазначити, що техніка динамічного аналізу є найбільш популярною, при цьому особливо ефективною вона є в поєднанні з механізмами віртуалізації, оскільки це дає можливість дослідити поведінку зразка ПЗ у середовищі, наближеному за своїми характеристиками до цільової.

Особливої ефективності виявлення шкідливого програмного забезпечення слід очікувати при поєднанні статичного і динамічного аналізу, оскільки статичний аналіз забезпечує максимальне покриття коду. Однак в умовах відсутності вихідних текстів ПЗ це зробити важче, ніж у разі їх наявності, оскільки потрібно використовувати дизасемблер, що дозволяє приблизно відновити високорівневий код програми.

МОДЕЛЮВАННЯ ПРОЦЕСУ ФУНКЦІОНУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Завдання забезпечення безпечного функціонування ПЗ включає: забезпечення інформаційної безпеки для інформаційного середовища під час функціонування ПЗ; забезпечення якісної та надійної реалізації ПЗ своїх функцій.

Програмне забезпечення за функціональним призначенням можна поділити на системне, прикладне та сервісне. Програмне забезпечення є компонентом інформаційної системи (операційної системи, яка своєю чергою може бути частиною програмно-апаратного комплексу, розподіленої обчислювальної мережі).

Оцінка надійності ПЗ пов'язана з можливістю впровадження тексту програмних засобів на етапі розробки та/або модифікації програмних закладок, використовуючи які, зловмисник реалізує створені вразливості. У зв'язку з цим, необхідність сертифікації та перевірки ПЗ на відповідність вимогам чинних нормативних документів протягом усього

його життєвого циклу, не викликає сумнівів [4, 5, 10].

Для оцінки надійності програмних продуктів, що є складовою інформаційних систем, окремий інтерес представляють рандомізовані моделі та методи теорії надійності. Розглянемо основні види моделей оцінки надійності програмних продуктів із зазначенням їх переваг та недоліків.

Модель Джелінскі – Моранди заснована на припущеннях, що час до наступної відмови ПЗ (у разі збою або кібератаки) розподілено експоненційно, а інтенсивність відмов ПЗ пропорційна кількості помилок, що залишилися в програмі.

Перевагою моделі є простота розрахунків, а основним недоліком те, що при неточному визначенні величини початкової кількості помилок інтенсивність відмов програми може стати негативною.

Модель Шумана, передбачається, що дослідження безпеки та надійності ПЗ проводиться в кілька етапів, кожен з яких являє собою виконання ПЗ набору тестових даних. Виявлені протягом етапу тестування помилки реєструються, але не виправляються. Після завершення етапу виправляються всі виявлені помилки, коригуються тестові набори та проводиться новий етап тестування.

Основною перевагою моделі є визначення всіх невідомих параметрів, необхідних для розрахунків, і відсутність необхідності звернення до інших моделей. Недоліком є припущення моделі про те, що при коригуванні програмного забезпечення не вносяться нові помилки. Також для розрахунків потрібна велика кількість зареєстрованих даних.

В основі моделі Модель Шика – Волвертона лежить припущення, за яким частота помилок пропорційна як кількості помилок в ПЗ, а й часу тестування, тобто, ймовірність виявлення помилок з часом зростає.

Модель дозволяє досить точно розраховувати надійність і проста в застосуванні.

Недолік моделі: для розподілів, відмінних від експоненціального, коли інтенсивність помилки змінюється з часом, необхідно використовувати умовні розподіли. При цьому умовна ймовірність для певного інтервалу тестування повинна відповідати проміжку часу від початку тестування до початку розгляду інтервалу тестування. Інакше цей процес відновлення за умови усунення виявлених помилок призведе до завищення ймовірностей безпомилкового функціонування на всіх ділянках тестування.

Модель Коркорена передбачає наявність у ПЗ багатьох джерел програмних відмов, пов'язаних з різними типами помилок та вразливостей, та різну ймовірність їх появи.

Перевагою моделі є те, що вона використовує ймовірності відмов, що змінюються, для різних типів помилок і оцінюється ймовірність безвідмовного виконання ПЗ на даний момент часу. Недоліком є виконання оцінки з урахуванням апріорної інформації чи даних попереднього періоду функціонування однотипних зразків ПЗ.

Модель IBM. Під час експлуатації користувачем поточної версії програмного забезпечення розробники, як правило, займаються його супроводом – вносять деякі покращення або виправлення в цю версію, не чекаючи, поки користувач цього вимагатиме. Супровід може включати також додавання нових функцій. З деякого моменту, коли розробник вважає своє завдання виконаним, починається пасивний супровід, коли виправлення вносяться вже за запитом користувача. При супроводі в кожну нову версію програмного забезпечення вноситься значна кількість нових помилок, разом із доопрацюваннями, змінами та виправленнями, що потребує виправлень також і в наступній версії.

Зокрема, в компанії IBM спробували передбачити кількість подібних виправлень від версії до версії, ґрунтуючись на велику кількість експериментальних даних, зібраних у ході супроводу операційної системи.

Модель Шнайдера пов'язує число помилок у програмі з витратами, виміряними в «людино-місяцях», числом

підпрограм і загальним числом *тисяч* операторів у програмі.

Перевагою моделі є низька обчислювальна складність і простота реалізації, до недоліків можно віднести необґрунтованість емпірично обраного числа виправлень і те що, що вразливості нульового дня не враховуються, а функції ПЗ не досліджуються детально.

МОДЕЛЬ БЕЗПЕЧНОГО ФУНКЦІОНУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Запропонована модель безпечного функціонування ПЗ має усувати недоліки, властиві розглянутим моделям. За результатами проведених досліджень моделей, можна зробити наступні висновки про їх недоліки: деякі моделі при реалізації вимагають значного обсягу обчислювальних ресурсів, це пов'язано як із необхідністю накопичення та обробки великої кількості даних, так і з обчислювальними ресурсами компонентів системи, які здійснюють аналіз безпеки; ймовірнісні моделі, як правило, ґрунтуються на припущеннях про те, що інтенсивність відмов/атак чи кількість помилок у ПЗ заздалегідь відомі, відомий розподіл (стандартний, пуассонівський, біноміальний), що не завжди вірно для реальних процесів і систем; відмови ПЗ та кібератаки, викликані експлуатацією вразливостей у ПЗ, розглядаються як сукупність, внаслідок чого моделі не враховують важливої особливості роботи шкідливого програмного забезпечення – а саме характеру його звернень з пам'яттю; жодна з розглянутих моделей не забезпечує комплексного представлення про процес функціонування ПЗ, у тому числі, погляд з боку інформаційної безпеки відсутній.

Для усунення вищеописаних недоліків пропонується основою моделі покласти опис взаємодії сутностей, задіяних у процесі функціонування ПЗ. При цьому необхідно враховувати специфіку поведінки шкідливого програмного забезпечення на комп'ютері або іншому обчислювальному пристрої.

Дослідження показали, що з основних ознак зараження різного типу шкідливого програмного забезпечення наступні:

1. Уповільнення роботи комп'ютера або вузла мережі. Як правило, це викликано впровадженням шкідливого програмного забезпечення та виконанням своїх функцій, що вимагають обчислювальних ресурсів, внаслідок чого на виконання легітимних обчислювальних задач потрібно більше часу.
2. Виконання нових функцій, не властивих даному комп'ютеру: відправлення даних на сторонні веб-ресурси, запуск процесів, які раніше не виконувались, і т.д.
3. Проблеми з відкриттям файлів, які раніше успішно відкривалися на комп'ютері, а також зміни розширення файлів (зникнення розширень або, навпаки, поява подвійних розширень).
4. Раптове завершення роботи додатків або поява спливаючих вікон при роботі додатків, що потенційно сигналізує про шкідливі процеси, що виконуються фоном.

Для шкідливого програмного забезпечення характерне надмірне використання обчислювальних ресурсів, а також операції з пам'яттю, що значно відрізняються від операцій, що виконуються легітимним ПЗ.

Так, наприклад, для шкідливого програмного забезпечення, що використовує обчислювальні потужності середовища для майнінгу, характерним є такий прояв у системі, як уповільнення роботи обчислювальних компонентів системи, поява обчислювальних процесів, що використовують велику кількість ресурсів, а також відправлення мережевого трафіку на сторонні веб-ресурси.

Таким чином, ключову роль описі роботи програмногозабезпечення грають: функції, що виконуються програмним забезпеченням; зміни у використанні обчислювальних ресурсів; зміни, пов'язані з роботою програмного забезпечення з пам'яттю.

На рис. 2 наведено графічне представлення моделі як набору взаємодіючих сутностей. Опис взаємодії сутностей моделі (рис. 2), відбиває як

надійність функціонування програмного забезпечення, що з виконанням заявлених функцій, а й безпеку функціонування програмного забезпечення.

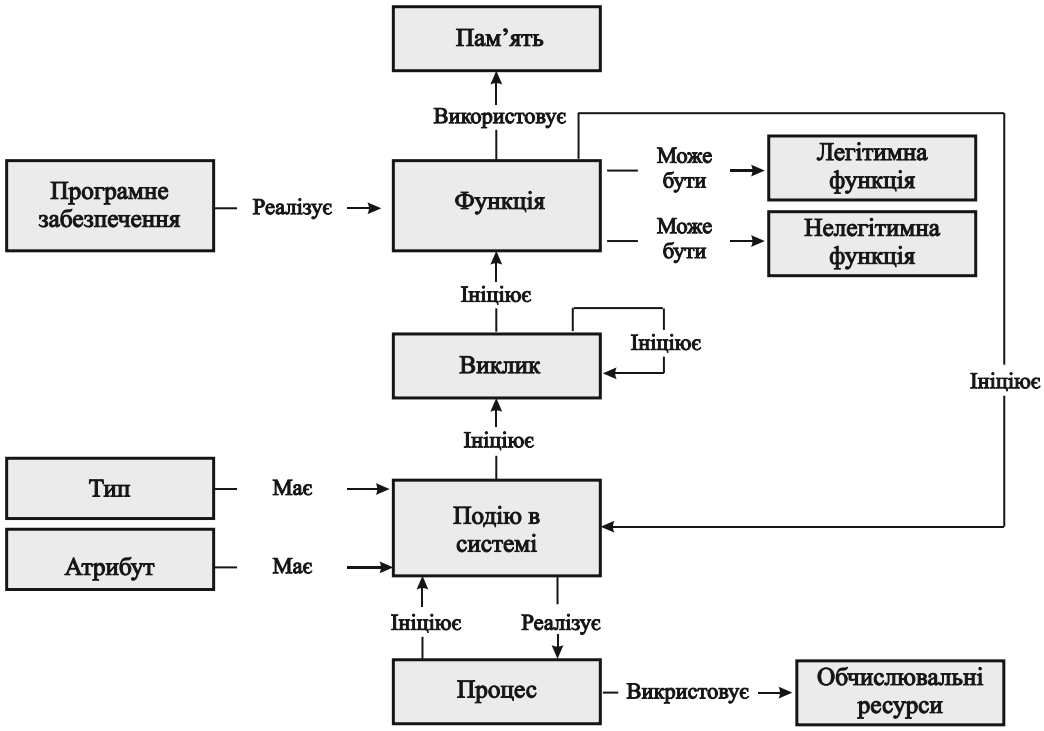


Рис. 2. Графічне представлення моделі безпечного функціонування програмного забезпечення

Fig. 2. Graphical representation of the model of secure software operation

Безпечним буде вважатися ПЗ, встановлення якого не веде до порушення легітимних відношень у системі, але також і те, що не має такого потенціалу внаслідок наявності у своєму складі певного шкідливого функціоналу. Для формалізації опису скористаємося математичним апаратом теорії множин. Опишемо модель безпечного функціонування ПЗ формально, представивши ПЗ ( ) як кортеж:

$$Software =< F, Sign, R, M, Memory > \qquad (1)$$

де:

1. $F = \{ f_1, f_2, ..., f_n \}$ - кінцева множина функцій, які виконує програмне забезпечення, що досліджується. Функції, що входять до множини F, поділяються на деструктивні F_{destr} та легітимні функції F_{leg}: $F = F_{leg} \bigcup F_{destr}$, $F_{leg} \bigcap F_{destr} = \varnothing$. При цьому, в будь якій із підмножин F_{leg} і F_{destr} можуть бути присутніми як документовані функції програмного забезпечення F_{doc}, так і не документовані F_{undoc}, $F_{doc} \bigcap F_{undoc} = \varnothing$.

2. $Sign$ – множина ознак, що характеризують виконання функцій програмного забезпечення. Маємо неоднозначне відображення множини F на множину $Sign : F \to Sign$, тобто кожному елементу множини F відповідає один або кілька елементів множини $Sign$. Використовуючи відповідні індекси, задамо відображення $F_{leg} \to Sign_{leg}$, $F_{destr} \to Sign_{destr}$, $F_{doc} \to Sign_{doc}$, $F_{undoc} \to Sign_{undoc}$.

3. R – обчислювальні ресурси, необхідні програмному забезпеченню для реалізації своїх функцій. Маємо відображення $F \to R$, внаслідок якого кожній f_i призначаються три чисельні значення: $\{ r_{i_{\min}}, r_i, r_{i_{\max}} \}$.

Очевидно, що оцінка необхідних обчислювальних ресурсів може змінюватись, проте в системі для кожного програмного продукту повинен бути виділений певний обсяг обчислювальних ресурсів, щоб система ефективно функціонувала.

4. M – множина методів дослідження, що застосовуються для виявлення шкідливого програмного забезпечення.

5. $Memory$ – множина операцій над пам'яттю, що виконуються при активації тієї чи іншої функції з множини F. Таким чином, маємо відображення $F \to Memory$, за якого кожній функції f_i зіставляється набір операцій над пам'яттю:

$$f_i \to \left\{ memory_{i_1}, memory_{i_2}, ..., memory_{i_k} \right\}.$$

Необхідно відзначити, що застосування методів M тільки в ідеальному випадку дозволить виявити множину ознак $Sign$, характерних для програмного продукту. На практиці слід говорити про визначення за допомогою методів M підмножини $Sign'$, $Sign' \in Sign$. На практиці вирішується завдання оцінки умовної ймовірності $P(A|B)$ події $A = \{Sign_{destr} = \varnothing \; and \; Sign_{undoc} = \varnothing\}$ при умові виконання події B реалізації деякого набору методів M', $M' \in M$ ′, що дозволяє виявити достатнє число ознак $Sign'_{destr}$ та $Sign'_{undoc}$,

$$B = \left\{ M' \to \left\{ Sign'_{destr}, Sign'_{undoc} \right\} \neq \varnothing \right\}.$$

Тоді умовна ймовірність $P(A|B)$ визначиться як (1):

Сформулюємо критерії безпечного функціонування програмного продукту в термінах моделі:

1. Програмний продукт не реалізує деструктивних та шкідливих функцій: $F_{destr} = \varnothing$ та $F_{undoc} = \varnothing$.

2. Об’єм, задіяних для аналізу безпеки програмного продукту (за допомогою множини методів M) обчислювальних ресурсів мінімізовано:

$$M : R \to R_{\min}, R_{\min} = \left\{ r_{1_{\min}}, r_{2_{\min}}, ..., r_{n_{\min}} \right\},$$

таким чином, кожна функція програмного продукту в ідеалі повинна використовувати для реалізації мінімально допустимого об’єму обчислювальних ресурсів;

3. Не повинно бути обчислювальних ресурсів, що витрачаються на будь - яку множину функцій F', що не перетинається з множиною $F : \exists F' \to R$, оскільки присутність шкідливого програмного забезпечення в системі часто призводить до активації нових, нетипових для роботи системи функцій.

4. Множина операцій над пам'яттю $Memory$ повинна мати відображення на множини ознак $Sign$, оскільки, важливою ознакою наявності шкідливого програмного забезпечення в системі є нетипові операції з пам'яттю: $Memory \to Sign \neq \varnothing$.

5. Множина методів дослідження, застосовуваних виявлення шкідливого програмного забезпечення, має використовувати результати відображення $Memory \to Sign$ означає, що для вирішення поставленого завдання доцільно застосовувати ті методи, які враховують операції програмного забезпечення з пам'яттю.

6. Множина використовуваних методів має бути такою, що умовна ймовірність $P(A|B)$ максимізована, тобто $M : P(A|B) \to 1 : (|) \to 1$.

$$P(A|B) = P(A) \cdot P(B) = P\{Sign_{destr} = \varnothing \; and \; Sign_{undoc} = \varnothing\} \cdot \cdot P\left\{ M' \to \left\{ Sign'_{destr}, Sign'_{undoc} \right\} \neq \varnothing \right\} \quad (1)$$

Досягнення максимальної ефективності використовуваних методів M малоймовірно, однак у такому разі буде досягнуто рівність (2):

$$Sign_{destr} = Sign'_{destr}, \; Sign = Sign'_{undoc} \quad (2)$$

Розроблена модель відрізняється від розглянутих моделей оцінки безпеки,

надійності та якості функціонування програмного продукту тим, що вона враховує як особливості функціонування шкідливого програмного забезпечення на компоненті системи, так і необхідність мінімізації обчислювальних ресурсів при аналізі. Особливу увагу слід приділити питанням практичної реалізації розробленої моделі, оскільки вона повинна поєднувати високу ефективність виявлення шкідливого програмного забезпечення та оптимальне використання обчислювальних ресурсів. Програмне забезпечення з потенційно небезпечними можливостями реалізує одну або кілька функцій, наслідки виконання яких можуть безпосередньо загрожувати порушенням безпеки даних, або призводити до виконання коду, що представляє цю загрозу. Види небезпечних можливостей представлені на рис. 3.

Значимість їх систематизації виходить з особливостях роботи з пам'яттю, виділені можливості, наступні: передача управління в область модифікованих даних; самодифікація або зміна коду іншого програмного забезпечення в оперативній пам'яті або на зовнішніх носіях; самодублювання, підміна собою іншого програмного забезпечення або перенесення своїх фрагментів в область оперативної або зовнішньої пам'яті, що не належить програмі; збереження інформації з областей оперативної пам'яті, які не належать програмному забезпеченню; спотворення, блокування чи заміну інформації, що є результатом роботи інших програм; приховування своєї присутності у програмному середовищі.

Виявлення функцій, що здійснюють роботу з пам'яттю таким чином, що виникають потенційно небезпечні можливості, особливо значимо, тому що це основна характеристика поведінки шкідливого програмного забезпечення. Тому при реалізації моделі на практиці слід використовувати механізм, який забезпечує ефективний моніторинг звернень до пам'яті та її використання.

Рис. 3. Програмне забезпечення з потенційно небезпечними можливостями

Fig. 3. Software with potentially dangerous capabilities

Процес у віртуальному операційному середовищі з повним контролем дій програмного забезпечення та відстеження алгоритму його роботи здійснюється більш ефективно. Таким чином реалізація розробленої моделі має базуватися на гіпервізорі, що дозволить віртуалізувати системні ресурси обчислювальних систем.

ВИСНОВКИ

В роботі виконано систематизацію моделей надійного та безпечного функціонування програмного забезпечення. В результаті проведеного дослідження виділено три типи моделей: аналітичні; статистичні; емпіричні.

Розглянуто ряд найчастіше застосовуваних моделей, виділено їх недоліки та переваги з погляду розв'язуваної задачі опису безпечного функціонування програмного продукту, та розпізнавання шкідливого програмного забезпечення. За результатами проведених досліджень розглянуті моделі мають переваги у плані простоти їх практичної реалізації, проте водночас виділено наступні недоліки: деякі з розглянутих моделей при реалізації вимагають великого обсягу обчислювальних ресурсів – для аналізу безпеки та накопичення архівних даних; використання статистичними та імовірнісними моделями припущень про те, що інтенсивність атак/відмов чи кількість помилок у програмному забезпеченню мають заздалегідь відомий розподіл (біноміальний, стандартний або пуасонівський), що не завжди вірно для реальних процесів і систем; немає поділу на відмови програмного забезпечення і вихід з ладу внаслідок кібератак, не враховуються також вразливості нульового дня; не аналізуються звернення досліджуваного програмного забезпечення до пам'яті, що могло б дати важливу інформацію про його легітимність чи наявність шкідливих функцій; жодна з розглянутих моделей не забезпечує комплексного представлення про процес функціонування програмного забезпечення, у тому числі, аналіз з боку інформаційної безпеки відсутній.

Завдання розпізнавання шкідливого програмного забезпечення з кожним роком стає дедалі актуальнішим і складнішим у зв'язку з цифровізацією галузей діяльності людини та використанням програмного забезпечення для виконання бізнес-логіки та технічних процесів у складних системах. Внаслідок цього, чим більший обсяг програмного забезпечення в системі, тим потенційно більше в ньому помилок, при цьому через підключення сучасних систем до мережі Інтернет, програмного забезпечення часто поширюється по мережі, що дозволяє зловмисникам створити нові вектори кібератак на системи.

Запропонована модель безпечного функціонування програмного продукту має усувати недоліки, властиві розглянутим моделям. Запропоновано модель усуває наведені недоліки за рахунок того, що вона враховує характерні особливості прояву шкідливого програмного забезпечення на пристроях, а саме вплив шкідливого програмного забезпечення на обчислювальні ресурси системи та роботу з пам'яттю. Це дозволяє розробленій моделі враховувати як надійність функціонування програмного забезпечення, так і безпеку.

У термінах моделі сформульовані критерії безпечного функціонування програмного забезпечення, зроблено висновок про те, що для найбільш ефективної реалізації такої моделі на практиці має бути використаний гіпервізор. Вибір конкретного гіпервізора залежить від потреб, призначення та специфіки системи.

REFERENCES

1. Doktryna informatsiinoi bezpeky Ukrainy, zatverdzhenoi Ukazom Prezydenta Ukrainy vid 25 liutoho 2017 roku № №47/2017, 15s.
2. Derzhavnyi standart Ukrainy Zakhyst informatsii. Tekhnichnyi zakhyst informatsii. Osnovni polozhennia. DSTU 3396.0-96 [Elektronnyi resurs]. – Rezhym dostupu: http://www.dsszzi.gov.ua/dsszzi/control/uk/publish/article?art_id=38883&cat_id =38836.
3. Zakon Ukrainy «Pro osnovni zasady zabezpechennia kiberbezpeky Ukrainy» zi zminamy. Vidomosti Verkhovnoi Rady (VVR), 2017, № 45, st.403, zi zminamy vid 28.07.2022 roku. Rezhym dostupu:

https://zakon.rada.gov.ua/laws/show/2163
4. **Tsybulnyk S. O., Barandych K. S.** (2022). Tekhnolohii rozroblennia prohramnoho zabezpechennia. Chastyna 1. Zhyttievyi tsykl prohramnoho zabezpechennia : bool. Kyiv, KPI im. Ihoria Sikorskoho, 270.
5. **Lysenko S. M., Shchuka R. V.** (2020). Analiz metodiv vyiavlennia shkidlyvoho prohramnoho zabezpechennia v kompiuternykh systemakh. *Visnyk Khmelnytskoho natsionalnoho universytetu*, 2(283). 101–107.
6. **Lenkov S. V., Dzhulii V. M., Bernaz A. M., Muliar I. V., Pampukha I.V.** (2023). Metod prohnozuvannia vrazlyvostei informatsiinoi bezpeky na osnovi analizu danykh tematychnykh internet-resursiv. *Zbirnyk naukovykh prats Viiskovoho instytutu Kyivskoho natsionalnoho universytetu imeni Tarasa Shevchenka*, Issue78, 123-134.
7. **Lenkov S. V., Dzhulii V. M., Solodieieva L. V**. (2022). Metod protydii poshyrenniu ta vyiavlennia shkidlyvoi informatsii v sotsialnykh merezhakh. *Zbirnyk naukovykh prats Viiskovoho instytutu Kyivskoho natsionalnoho universytetu imeni Tarasa Shevchenka*, Issue 77, 103-117.
8. **Lenkov S. V., Dzhulii V. M., Orlenko V. S., Sieliukov O. V., Atamaniuk A. V.** (2020). Model bezpeky poshyrennia zaboronenoi informatsii v informatsiino-telekomunikatsi-inykh merezhakh. *Zbirnyk naukovykh prats Viiskovoho instytutu Kyivskoho natsionalnoho universytetu imeni Tarasa Shevchenka*, Issue 68, 53-64.
9. **Krepych S. Ya., Spivak I. Ya.** (2020). Yakist prohramnoho zabezpechennia ta testuvannia : bazovyi kurs. Ternopil, FOP Palianytsia V. A. Publ., 478.
10. **Zolotukhina O. A., Nehodenko O. V., Reznyk S. Yu., Razina S. Ya.** (2020). Iakist ta testuvannia informatsiinykh system : Navchalnyi posibnyk dlia samostiinoi roboty studentiv vyshchykh navchalnykh zakladiv, Kyiv, NNIIT DUT, 128.
11. **Vyshnia V. B., Havrysh O. S., Ryzhkov E. V.** (2020). Osnovy informatsiinoi bezpeky: book. Dnipro, Dniprop. derzh. un-t vnutrish. sprav, 128.
12. Cotsialni merezhi – realni zahrozy virtualnoho svitu. [Elektronnyi resurs]. – Rezhym dostupu : http://ogo.ua/ articles/view/011–02–23/26490.htm.
13. **Ostapov S. E., Yevseiev S. P., Korol O. H.** (2016). Tekhnolohii zakhystu informatsii: book. Kharkiv, Vyd-vo KhNEU, 476.
14. **Lienkov S. V., Dzhulii V. M., Bernaz N. M., Bozhuk S. O.** (2017). Analiz isnuiuchykh metodiv ta alhorytmiv vyiavlennia atak v bezdrotovykh merezhakh peredachi danykh. *Zbirnyk naukovykh prats Viiskovoho instytutu Kyivskoho natsionalnoho universytetu imeni Tarasa Shevchenka*. Kiyv, VIKNU, Issue 56, 124-132.
15. **Buriachok V. L. Toliupa S. V., Semko V. V.** Informatsiinyi ta kiberprostory: problemy bezpeky, metody ta zasoby borotby : Book. Kyiv, DUT-KNU, 178.
16. **Rybalchenko L. V., Kosychenko O. O.** (2019). Problemy bezpeky personalnykh danykh v Ukraini. *Rehionalna ekonomika*. Zaporizhzhia. 2019, 57-62.
17. **Dzhulii V. M., Miroshnichenko O.V., Solodieieva L.V.** (2022). Metod klasyfikatsii dodatkiv trafika kompiuternykh merezh na osnovi mashynnoho navchannia v umovakh nevyznachenosti. *Zbirnyk naukovykh prats Viiskovoho instytutu Kyivskoho natsionalnoho universytetu imeni Tarasa Shevchenka*. Kyiv VIKNU, Issie 74, 73-82.
18. **Lavrov Ye. A., Perkhun L. P., Shendryk V. V.** (2017). Matematychni metody doslidzhennia operatsii: pidruchnyk. Sumy: Sumskyi derzhavnyi universytet, 212.
19. **Honchar S. F.** (2019). Otsiniuvannia ryzykiv kiberbezpeky informatsiinykh system obiektiv krytychnoi infrastruktury: monohrafiia. Kyiv, 175.
20. **Yemchuk L., Zhylinska O., Chornyi A., Dzhuliy V.** (2020). Organizational Network Analysis as a Tool for Leadership Assessment in Software Development Team. *Institute of Electrical and Electronics Engineers (30 September 2020)*; INSPEC Accession Number: 20008165; DOI: 10.1109/ACIT49673.2020.
21. Syhnatura ataky. Wikipedia [Elektronnyi resurs] – Rezhym dostupu do resursu: https://uk.wikipedia.org/wiki/Syhnatura_ataky
22. OPWNAI: Cybercriminals Starting to Use ChatGPT, January 6, 2023 [Elektronnyi resurs] – Rezhym dostupu do resursu: https://research.checkpoint.com/2023/opwnai-cybercriminals-starting-to-usechatgpt.

Information security model of functioning software

Serhii Lienkov, Volodymyr Dzhuliy, Oleksandr Yavorskyi, Kostyantyn Zatsepin

Abstract. The paper systematizes the models of reliable and safe functioning of the software. As a result of the research, three types of models were identified: analytical; statistical; empirical.

A number of the most frequently used models are considered, and their disadvantages and advantages are highlighted from the point of view of solving the problem of describing the safe functioning of a software product and recognizing malicious software. According to the results of the research, the considered models have advantages in terms of the simplicity of their practical implementation, but at the same time, the following disadvantages are highlighted: some of the considered models require a large amount of computing resources when implemented - for security analysis and accumulation of archival data; the use of statistical and probabilistic models of assumptions that the intensity of attacks/failures or the number of errors in software have a pre-known distribution (binomial, standard or Poisson), which is not always true for real processes and systems; there is no division into software failures and failures due to cyber attacks, zero-day vulnerabilities are also not taken into account; memory accesses of the investigated software are not analyzed, which could provide important information about its legitimacy or the presence of malicious functions; none of the considered models provides a comprehensive representation of the process of software functioning, including, there is no analysis from the information security side.

The task of recognizing malicious software is becoming more and more relevant and difficult every year in connection with the digitalization of human activities and the use of software for the execution of business logic and technical processes in complex systems. As a result, the larger the volume of software in the system, the more errors there are potentially, and due to the connection of modern systems to the Internet, the software is often distributed over the network, which allows attackers to create new vectors of cyber attacks on systems.

The proposed model of safe functioning of the software product should eliminate the shortcomings inherent in the considered models. The proposed model eliminates the mentioned shortcomings due to the fact that it takes into account the characteristic features of the manifestation of malicious software on devices, namely the impact of malicious software on the computing resources of the system and working with RAM. This allows the developed model to take into account both the reliability of software operation and security.

In terms of the model, the criteria for the safe functioning of the software are formulated, it is concluded that for the most effective implementation of such a model in practice, a hypervisor should be used.

Keywords: model, information security, vulnerabilities, attacks, confidential data, malware, secure operation.

к.т.н., доц. Муляр І.В. (ХмНУ)
Главюк В.С. (ХмНУ)
Зацепін К.О. (ХмНУ)
Чернов С.В. (ХмНУ)

ВИКОРИСТАННЯ МОДЕЛІ GPT ДЛЯ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ IoT-ПРИСТРОЇВ

З розвитком Інтернету речей (IoT) кількість підключених пристроїв стрімко зростає, що створює нові виклики у сфері кібербезпеки. IoT-пристрої часто мають обмежені обчислювальні ресурси, різноманітні операційні системи та використовують різні протоколи зв'язку, що ускладнює їх тестування на безпеку.

Модель GPT здатна аналізувати та генерувати різноманітні запити та команди, які можуть бути застосовані до IoT-пристроїв через різні інтерфейси, такі як HTTP, MQTT, CoAP та інші. Завдяки здатності розуміти контекст та особливості різних протоколів, GPT може генерувати специфічні тестові сценарії, спрямовані на виявлення вразливостей у прошивці, автентифікації, управлінні доступом та інших аспектах безпеки IoT-пристроїв.

Наприклад, модель GPT може бути використана для автоматизованого виявлення вразливостей типу «переповнення буфера», «недостатня перевірка вводу» або «жорстко закодовані облікові дані». Генеруючи різні варіанти вхідних даних та аналізуючи відповіді пристрою, система може ідентифікувати потенційні слабкі місця, які можуть бути використані зловмисниками. Це особливо актуально для пристроїв з обмеженими можливостями оновлення безпеки, де виявлення та усунення вразливостей на ранніх стадіях є критичним.

Крім того, GPT може допомогти у виявленні вразливостей, пов'язаних з мережевою взаємодією IoT-пристроїв. Модель може симулювати атаки типу «людина посередині» (MITM), підробку пакетів або перехоплення даних, генеруючи відповідні сценарії та аналізуючи реакцію пристроїв на такі дії. Це дозволяє оцінити стійкість IoT-систем до мережевих загроз та підвищити загальний рівень безпеки.

Використання моделі GPT у тестуванні IoT-пристроїв також сприяє автоматизації процесу та зменшенню необхідності ручної праці. Завдяки можливості моделі швидко адаптуватися до різних протоколів та стандартів, фахівці з безпеки можуть охопити ширший спектр пристроїв та сценаріїв використання. Це особливо важливо в умовах постійного зростання кількості

IoT-пристроїв та швидкого розвитку технологій, де традиційні методи тестування можуть бути недостатньо ефективними.

Загалом, інтеграція моделі GPT у процес тестування безпеки IoT-пристроїв має потенціал суттєво підвищити ефективність виявлення вразливостей та сприяти створенню більш безпечної екосистеми IoT. Це відкриває нові перспективи для досліджень та розробки інструментів, які допоможуть забезпечити надійність та захищеність сучасних мереж Інтернету речей.

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Зацепіна Костянтина Олександровича
ПІБ здобувача вищої освіти

Студента ФІТ, 2 курсу, групи КБЗІм-23-1

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 дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а) та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмнотехнічного засобу Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів в роботах.

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

5.12.2024
дата

підпис

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: Костянтин Зацепін

Співавтор:

Назва: Метод тестування безпеки програмного забезпечення для захисту інформації в корпоративній мережі

Науковий керівник: Ігор Муляр

Підрозділ: Кафедра кібербезпеки

Коефіцієнт подібності 1:1.3%

Коефіцієнт подібності 2:0%

Мікропробіли: 0

Заміна букв: 2

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2024-12-10 11:12:51.0

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 10.12.2024 експерт 

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 7%**

ID: 157200 Назва: Метод тестування безпеки програмного забезпечення для захисту інформації в корпоративній мережі Додано в БД: 2024-12-10 Автора: Зацепін Костянтин Керівники: Муляр І.В. Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	103356	792	817 (1%)	9 (1%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:
Назва: Метод тестування безпеки програмного забезпечення для захисту інформації в корпоративній мережі

Автор: Зацепін Костянтин Олександрович

Спеціальність: 125 – Кібербезпека та захист інформації

Освітня програма: Кібербезпека та захист інформації

Науковий керівник: Муляр Ігор Володимирович

Після аналізу звіту подібності зроблено такий висновок:

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

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою StrikePlagiarism складає 98,7%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism v-15.257 складає 99%.

Згідно з правилами чинного Положення «Про систему забезпечення академічної доброчесності у хмельницькому національному університеті» від 24.09.2024, авторська робота, обсяг оригінального тексту у відсотках до загального обсягу матеріалу в якій складає 90-100 %, визнається роботою з високою унікальністю тексту і допускається до захисту.

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

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

Керівник роботи _____ Ігор МУЛЯР

Гарант ОП _____ Віра ТІТОВА

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

Дата: 13.12.2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «магістр»

Магістр Зацепін К.О.
Тема Метод тестування безпеки програмного забезпечення для захисту інформації в корпоративній мережі

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека

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 Проаналізувано чинники, що впливають на забезпечення ефективного функціонування мережі, розглянути критерії та існуючі методи їх оцінки. Було створено математичну модель кібератаки, яка дає змогу дослідити ключові етапи проведення атаки та розробити практичні рекомендації щодо захисту мережі від подібних загроз. Удосконалено також математичну модель діагностики системи управління мережевими ресурсами, що сприяло підвищенню ефективності тестування безпеки програмного забезпечення корпоративної мережі. Результати моделювання продемонстрували можливість покращення ефективності тестування безпеки ПЗ на 4%.

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 обґрунтовується 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 програмного забезпечення, яке функціонує вкорпоративній мережі

4. Позитивні сторони роботи Кваліфікаційна робота містить ряд інноваційних рішень, зокрема запропонований метод забезпечує комплексний підхід до діагностики безпеки корпоративної мережі, поєднуючи можливості імітаційного моделювання, низькорівневого аналізу двійкового коду та математичної формалізації мережевих процесів

5. Негативні сторони роботи Розроблений метод не враховує динамічні зміни коду

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

7. Відгук про роботу в цілому В загальному кваліфікаційної роботи заслуговує позитивної оцінки. Весь матеріал структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи..

8. Інші зауваження

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

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

«13» 12 2024. _____ (підпис)

ХНУ. Зам. 54, тир. 6000, 2007.