Міністерство освіти і науки України
Харківський національний університет радіоелектриніки
Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Автоматизації проєктування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Автомат пошуку шкідливих компонентів у кіберпросторі
(тема)

Виконав: здобувач IV курсу, групи КІУКІ-21-9
Будаков Р.А.
(прізвище, ініціали)
Спеціальність *123 _ Комп'ютерна інженерія*
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник Хаханов В.І.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри АПОТ _____ Чумаченко С.В
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123_Комп'ютерна інженерія
(шифр і назва)

Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри АПОТ
Чумаченко С.В.
(підпис)
«_06_»___05_____ 2025 р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Будакову Рамазану Алазан-огли
(прізвище, ім'я, по батькові)

1. Тема роботи Автомат пошуку шкідливих компонентів у кіберпросторі

затверджена наказом університету від __21_____05_____2025 р. № _403Ст___

2. Термін подання студентом роботи до екзаменаційної комісії _20.06.2025 р.

3. Вихідні дані до роботи
Пошук шкідливих компонентів у кіберпросторі.
Середовище розробки PyCharm IDE 2024.1.3
Мова програмування Python 3.12.
Python у системах IoT

4. Перелік питань, що потрібно опрацювати в роботі
Постановка задачі. Аналіз та огляд існуючих альтернатив.
Розробка структурної схеми додатка.
Розробка алгоритму роботи додатка.
Аналіз розробленої системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

16 слайдів

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження і затвердження теми.	06.05.2025 – 06.05.2025	
2	Аналіз предметної області, постановка задачі, вибір стеку технологій.	07.05.2025 – 10.05.2025	
3	Розробка структури автомату пошуку шкідливих кодів		
4	Розробка алгоритму, загальної логіки та схеми роботи додатка.	11.05.2025 – 20.05.2025	
5	Розробка програмної частини додатка.	21.05.2025 – 31.05.2025	
6	Тестування та аналіз розробленого продукту, у порівнянні з конкурентами.	01.06.2025 – 03.06.2025	
7	Оформлення пояснювальної записки.	04.06.2025 – 10.06.2025	
8	Перевірка проєкту науковим керівником, допуск до захисту роботи.	11.06.2024 – 12.06.2025	

Дата видачі завдання 06.05.2025 р.

Здобувач _____ Будаков Р.А.
(підпис)

Керівник роботи _____ проф. Хаханов В.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка з кваліфікаційної містить: 47 с., 10 рисунків, 1 таблиця, 2 додатки, 40 джерел за переліком посилань.

ВІРУС, ТРОЯН, ЧЕРВ'ЯК, ШПИГУНСЬКЕ ПЗ, РЕКЛАМНЕ ПЗ, РУТКІТ, РАНСОМВАР, ФІШИНГ, БОТНЕТ, ЕКСПЛОЙТ, АРХІТЕКТУРА, СТРУКТУРА, КОМП’ЮТИНГ КІБЕРЗАХИСТУ, АВТОМАТ, КІБЕРПРОСТІР

Розробляється структура автомату пошуку шкідливих компонентів у кіберпросторі у межах архітектури комп’ютингу деструктивних кодів. Реалізується застосунок для аналізу веб-ресурсів, що виконує завдання виявлення потенційно шкідливого коду.

Мета роботи – підвищення якості розпізнавання та спрощення аналізу malware шляхом розробки автомату пошуку шкідливих компонентів у кіберпросторі у межах архітектури комп’ютингу деструктивних кодів.

Задачі: огляд сучасних тенденцій розвитку кіберпростору та літератури; аналіз інфраструктури – моделей, методів, архітектур, технологій – для ідентифікації шкідливих програм; розробка автомату пошуку шкідливих кодів у кіберпросторі; створення програмного застосунку.

REFERENCE

The explanatory note of the qualification work contains: 47 pages, 10 figures, 1 table, 2 applications, 40 references.

VIRUS, TROJAN, WORM, SPYWARE, ADWARE, ROOTKIT, RANSOMWARE, PHISHING, BOTNET, EXPLOIT, ARCHITECTURE, STRUCTURE, COMPUTING FOR CYBER DEFENSE, AUTOMAT, CYBERSPACE

The structure of the automaton for searching malware in cyberspace within the framework of the computing architecture of destructive codes is being developed. An application for analyzing web resources is being implemented that performs the task of detecting potentially malicious code.

The work aims to improve the quality of recognition and simplify malware analysis by developing a machine for searching for malicious components in cyberspace within the framework of the architecture of destructive code computing.

Tasks: review of current trends in the development of cyberspace and literature; analysis of the infrastructure – models, methods, technologies – for identifying malicious programs; development of a machine for searching for malicious codes in cyberspace; creation of a software application.

.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (інтерфейс прикладного програмування);

BGRU – Bidirectional Gated Recurrent Unit Neural Network (двонаправлена нейронна мережа зі стробованим рекурентним блоком);

BLSTM – Bidirectional Long Short-Term Memory (двонаправлена довготривала пам’ять);

cDOS – Citizen-Led Denial of Service (атака на відмову в обслуговуванні, керована громадянами);

CPS – Cyber Physical System (кіберфізична система);

ENISA – European Union Agency for Cybersecurity (Європейська спільна агенція з кібербезпеки);

GRU – Gated Recurrent Unit (керований рекурентний блок);

LFSR – Linear-Feedback Shift Register (регістр зсуву з лінійним зворотнім зв’язком);

LSTM – Long Short-Term Memory (довга которкочасна пам’ять);

CSS – Crystal Security (пісочниця);

DLP-системи – Data Loss Prevention (програмне забезпечення, призначене для захисту від утечки конфіденційної інформації);

DNN – Deep Neural Network (глибокі нейронні мережі);

IdM-система – Identity Management (система централізованого керування ідентифікаційними даними та правами доступу);

IT – інформаційні системи;

ITRC – Identity Theft Resource Center (Центр ресурсів з питань крадіжки особистих даних);

ML − машинне навчання;

NGFW – міжмережний екран нового покоління;

FML – Federated Machine Learning (федеративне машинне навчання);

PAM – Privileged Access Management;

PUA – потенційно небажані додатки;

SD-автомат – Similarity-Difference (підібність/відмінність) автомат;

SOC – власний чи комерційний центр моніторингу;

SSDLC – Secure Software Development Life Cycle (безпечний життєвий цикл розробки програмного забезпечення – методики забезпечення безпеки кожному етапі розробки);

VHDL – Very High Speed Integrated Circuits, Hardware Description Language;

VPN – Virtual Private Network (віртуальна приватна мережа);

WAF – Web Application Firewall (брандмауер веб-застосунків);

ІКТ – інфокоммунікаційні технології;

КП – коректна програма;

ПЗ – програмне забезпечення;

ШІ – штучний інтелект;

ШК – шкідливий код;

ШП – шкідлива програма.

ВСТУП

У сучасну цифрову епоху зі зростанням кількості кібератак та використанням шкідливого коду на веб-сайтах виникає потреба у створенні ефективних інструментів для швидкого виявлення потенційно небезпечного контенту.

Розробляється структура автомату пошуку шкідливих компонентів у кіберпросторі у межах архітектури комп'ютингу деструктивних кодів. Реалізується застосунок для аналізу веб-ресурсів, що виконує завдання виявлення потенційно шкідливого коду.

Мета роботи – підвищення якості розпізнавання та спрощення аналізу malware шляхом розробки автомату пошуку шкідливих компонентів у кіберпросторі у межах архітектури комп'ютингу деструктивних кодів.

Задачі: огляд сучасних тенденцій розвитку кіберпростору та літератури; аналіз інфраструктури – моделей, методів, архітектур, технологій – для ідентифікації шкідливих програм; розробка автомату пошуку шкідливих кодів у кіберпросторі; створення програмного застосунку.

1 СТАН ТЕХНОЛОГІЙ

Розглядаються сучасні тенденції та технології, що визначають актуальність роботи. Наводиться огляд літератури.

1.1 Актуальність тематики

За 2023 кількість атак, що призвели до витоку персональних даних, збільшилось на 78%. Про це відзначає у своєму звіті некомерційна компанія «Центр боротьби з розкраданнями персональних даних» (ITRC) [1]. Обсяг скомпрометованих даних зростає з кожним роком. Таке активне зростання злочинів стимулює розвиток нових методів захисту інформації. У зв'язку з цим спостерігаються нові тренди та тенденції у сфері кібербезпеки.

1. *Міжнародні тренди.* Індустрія інформаційної безпеки не стоїть на місці. Щоб перемогти в гонці з кіберзлочинцями, розробникам систем безпеки потрібно знати нові технології і якнайшвидше адаптувати їх до своєї роботи.

Компанія Gartner Inc. щорічно визначає глобальні технологічні тенденції [8], формуючи перспективні напрями на 5-10 років. Серед них – розвиток інтернету етичної поведінки, узагальнення досвіду в різних сферах та впровадження в керування людиною, підвищення конфіденційності обробки даних; розподілені хмарні сервіси, інваріантними до геолокації суб'єктів, кібербезпека процесів; засоби AI-engineering, гіперавтоматизація, управління, операції, моніторинг та захист від вірусів кіберпростору (рис. 1.1).

Gartner відокремлює 8 прогнозів щодо кібербезпеки на 2023-2025 роки [9]. Акцент робиться на законах про конфіденційність даних, атаках програм-вимагачів, кіберфізичних системах і аудитах на рівні правління визначаються пріоритети лідерів безпеки та ризиків.

Розповсюдження кіберфізичних систем, що включають системи, які поєднують кібернетичний і фізичний світи для таких технологій, як автономні автомобілі чи цифрові близнюки, створює ще один ризик для безпеки для

організацій, і те, як кібер-зловмисники можуть націлитися на ці системи, є одним із найважливіших прогнозів на найближчі роки.

Рисунок 1.1 – Схема глобальних технологічних тенденцій [9]

Безпека та управління ризиками стали питанням правління організацій. Порушення безпеки стають все більш поширеними та складнішими, що призводить до прийняття нових законів для захисту споживачів і компаній, які ставлять безпеку в центр своїх рішень.

На найближчі кілька років аналітики Gartner передбачають середовище, в якому більша децентралізація, посилене регулювання та наслідки для безпеки будуть більш серйозними до кінця 2025 року, а саме:

– 60% організацій використовуватимуть ризик кібербезпеки як основний визначальний фактор у проведенні операцій із третіми сторонами та ділових відносинах. Інвестори, особливо венчурні капіталісти, використовують ризик кібербезпеки як важливий фактор при оцінці можливостей. Організації все більше звертають увагу на ризик кібербезпеки під час ділових угод, включаючи злиття та поглинання та угоди з постачальниками. Як наслідок,

можуть надходити запити на додаткові дані про програму кібербезпеки партнера через опитування чи рейтинги безпеки;

– відсоток штатів, які приймають закони, що регулюють виплати програм-вимагачів, штрафи та переговори, зросте до 30% порівняно з менш ніж 1% у 2021 році;

– 40% рад директорів матимуть спеціальний комітет з кібербезпеки під керівництвом кваліфікованого члена правління;

– 70% генеральних директорів створять культуру корпоративної стійкості, щоб захистити себе від загроз кіберзлочинності, суворих погодних явищ, соціальних подій і політичної нестабільності;

– кібер-зловмисники зможуть використовувати робочі технологічні середовища як зброю достатньо успішно, щоб спричинити людські жертви.

2. *Застосування штучного інтелекту*. У 2023 році хакери активно використовували генеративний штучний інтелект для фішингових кампаній та зломів сайтів. Агентство з кібербезпеки Євросоюзу ENISA прогнозує, що зловживання штучним інтелектом залишиться в топ-10 [2] найбільших загроз кібербезпеці аж до 2030 року.

Агентство Європейського союзу з кібербезпеки (ENISA) публікує короткий виклад «Прогнозу загроз кібербезпеці на 2030 рік», в якому представлено огляд основних результатів у 10 найкращих рейтингах. Наступний список десяти найважливіших загроз включає переглянутий список нових загроз кібербезпеці, які матимуть вплив до 2030 року (рис. 1.2):

1) загроза ланцюжка постачання залежностей програмного забезпечення;

2) нестача навичок;

3) людські помилки та експлуатація застарілих систем у кіберфізичних екосистемах;

4) експлуатація невиправлених та застарілих систем у перевантаженій міжгалузевій технологічній екосистемі [нове в десятці];

5) зростання авторитаризму цифрового спостереження/втрата конфіденційності;

6) транскордонні постачальники послуг ІКТ як єдина точка відмови;

7) розширені кампанії з дезінформації/операції впливу;

8) зростання розширених гібридних загроз;

9) зловживання ШІ;

10) фізичний вплив природних/екологічних збоїв на критично важливу цифрову інфраструктуру [нове у десятці].

Рисунок 1.2 – Схема десяти найновіших загроз кібербезпеці до 2030 року [2]

Для злочинних схем зловмисники використовували як гучний ChatGPT від OpenAI, і власні мовні моделі. Поширення моделей з відкритим вихідним кодом опосередковано допомогло шахраям: вони брали за основу чужий інструмент та доопрацьовували код під свої завдання.

У існуючі комерційні ШІ вшиті обмеження, які не дають їм створювати шкідливого контенту, проте зловмисники навчилися обходити ці фільтри. А моделі, які шахраї розробляють самостійно, взагалі не мають етичних обмежень. Їх можна натренувати на вмісті хакерських форумів із даркнету.

Зловмисники також почали використовувати нейромережі у техніках соціальної інженерії. Наприклад, для підробки голосу та створення дипфейків у відеодзвінках [3].

Докладніше про те, як кіберзлочинці використовують мовні моделі, можна дізнатися з публікації поліцейської служби [4] Європейського Союзу Європол. Які заходи безпеки та існуючі рішення. Робота йде за двома основними напрямками: формування міжнародних етичних норм та застосування технологій штучного інтелекту в інформаційній безпеці.

З одного боку, великі міжнародні організації пропонують ухвалити єдиний кодекс етичного використання ШІ, щоб обмежити можливості кіберзлочинців. Наприклад, ЮНЕСКО ще у 2021 році представила свій збір відповідних рекомендацій [5].

З 2024-го Європарламент затвердив перший у світовій практиці закон про регулювання штучного інтелекту [6]. "Метою проекту є захист фундаментальних прав, демократії, верховенства закону та екологічної стійкості від високоризикованого штучного інтелекту, а також стимулювання інновацій та затвердження Європи як лідера в цій галузі. Регламент встановлює зобов'язання щодо ШІ, виходячи з його потенційних ризиків та рівня впливу", – йдеться на сайті парламенту.

З іншого боку, самі розробники рішень щодо інформаційної безпеки залучають ШІ у своїх продуктах, наприклад:

– використовують механізми машинного навчання, щоб виявляти нестандартні атаки;

– застосовують генеративний штучний інтелект на кшталт великих мовних моделей, щоб симулювати атаки, аналізувати вихідний код та виявляти ознаки шахрайства.

Інструменти протидії шахраям на базі таких технологій активно розвиваються і з кожним роком ставатимуть все швидше та ефективніше.

3. *Навчання співробітників основ кібербезпеки*. Агентство ENISA склало рейтинг найбільших загроз інформаційній безпеці [2]. Нестача навичок у

співробітників – на другому місці. Більше 60% атак хакерів починаються з електронних листів. Чекати на скорочення хакерських атак не доводиться. Найефективнішим способом протидії залишається навчання персоналу. Окрема проблема – брак навичок у працівників сфери кібербезпеки.

Дослідницька компанія Gartner вважає їх перекваліфікацію та перегляд [7] існуючих ролей одним із головних трендів 2024 року. Основи інформаційної безпеки повинні знати всі співробітники, які працюють з інформацією в інтернеті. Деякі великі компанії вже роблять це. Наприклад, вони проводять кібернавчання – розсилають фішингові листи, імітують внутрішні портали та веб-додатки, якими співробітники користуються щодня. Це допомагає команді розвивати уважність та критичне мислення, щоб не попастися на хитрощі шахраїв.

Крім навчання співробітників, важливо розвивати культуру зберігання даних. У кожного в компанії повинен бути доступ тільки до інформації, яка потрібна йому для роботи. Навіть якщо шахраї обдурять когось із співробітників, вони матимуть доступ лише до невеликої частини конфіденційної інформації, а не до всієї бази даних компанії. З цією метою впроваджують IdM- та DLP-системи.

Системи IdM розмежовують доступ до даних для різних співробітників, автоматично застосовуючи та підтримуючи попередньо налаштовану модель доступу на основі ролей:

– поміщають облікові записи працівників групи, які дають необхідний мінімум мережевих доступів при підключенні через корпоративний VPN;

– дають менеджерам доступ лише до частини клієнтської бази, з якою кожен із них працює.

Щоб конфіденційна інформація не проникла за межі компанії, можна налаштувати DLP-систему. Вона допоможе відстежувати переміщення даних і повідомить, якщо хтось із співробітників спробує надіслати інформацію або скопіювати на зовнішній носій.

4. *Розвиток екосистем.* Щоб ефективно збудувати систему безпеки в компанії, потрібно застосовувати різнопланові рішення. При цьому складно сформувати систему захисту із програм різних виробників: потрібно інтегрувати їх один з одним, постійно підтримувати та налаштовувати для протидії актуальним загрозам, вчасно встановлювати оновлення. Це забирає багато часу та вимагає від співробітників служби безпеки високої експертності.

Останнім часом набирають популярності екосистеми, які поєднують кілька продуктів:

– кожна програма виконує свої завдання, але при цьому вони обмінюються даними, доповнюють та посилюють один одного для комплексного захисту від кібератак;

– елементи екосистеми налаштовуються так, щоб співробітники могли централізовано керувати захистом на всіх рівнях, не переймаючись сумісністю програм;

– на обслуговування екосистем потрібно менше витрат, оскільки частина завдань переноситься на вендора.

Проте екосистемні рішення мають особливості, які потрібно врахувати на старті. Наприклад, ризик vendor lock-in, тобто прив'язки до постачальника. Вибираючи екосистемне програмне забезпечення, компанія потрапляє в залежність від цінової політики вендора. За більших обсягів впроваджень це може стати навантаженням для бюджету.

5. *Безперервний аналіз уразливостей та загроз*. В останні роки розвиваються сервіси та технології, які, з одного боку, прискорюють розробку, а з іншого – відкривають уразливості, якими користуються шахраї. Технологічний стек бізнес-сервісів, ПЗ та бібліотеки з відкритим вихідним кодом значно збільшують поверхню атаки.

Для посилення безпеки слід робити наступне:

– налаштувати процес контролю периметра, тобто вести облік системних та мережевих (у тому числі хмарних) сервісів компанії, які доступні

в Інтернеті. Не публікувати нові сервіси у відкритому доступі доти, доки служба безпеки не вживе заходів щодо захисту від кібератак – відповідно до типу сервісу;

– організувати процеси управління вразливістю та управління оновленнями у корпоративній інфраструктурі;

– використовувати засоби захисту, які виявляють та блокують спроби експлуатації вразливостей: антивірусні програми та міжмережеві екрани (NGFW, WAF).

Особливу увагу варто приділяти перевірці підрядників та використанню продуктів із відкритим вихідним кодом. Убезпечити компанію від недобросовісної поведінки підрядників допоможе обмеження та контроль доступу до інфраструктури, у тому числі із застосуванням засобів управління привілейованим доступом, та реалізація заходів, аналогічних застосованим до зовнішнього периметру.

Контролювати використання продуктів з відкритим вихідним кодом складніше: не всі вендори розкривають, які бібліотеки вони використовували при створенні продукту. Ще складніше відстежити транзитивні залежності, тобто бібліотеки, використані для написання інших бібліотек.

Під час розробки продуктів самотужки слід дотримуватися підходу SSDLC – методики забезпечення безпеки на кожному етапі розробки. Як підвищити рівень кібербезпеки в компанії:

– мінімізувати кількість мережевих сервісів компанії доступних в інтернеті. Перш ніж опублікувати сервіси у вільному доступі, важливо узгодити це зі службою безпеки, вжити захисних заходів та організувати контроль периметра компанії;

– сегментувати корпоративну мережу та обмежувати надмірний мережевий доступ як для користувачів, так і між сегментами;

– наслідувати принцип мінімальних привілеїв при видачі співробітникам доступів. Щоб знизити ризики нелегітимного використання привілейованих облікових записів, слід налаштувати керування привілейованим доступом, у

тому числі із використанням систем класу PAM та Privileged Access Management;

– вибудувати процес управління вразливістю та оновленнями в корпоративній інфраструктурі;

– регулярно робити аудити корпоративної інфраструктури, щоб перевірити, чи виконуються вимоги кібербезпеки;

– слідкувати за налаштуванням прав доступу у хмарних сервісах, якщо компанія їх використовує;

– регулярно освіжати знання співробітників про основи кібербезпеки;

– налаштувати процес виявлення атак за допомогою власного чи комерційного центру моніторингу – SOC. Для цього потрібно організувати збирання подій безпеки з компонентів корпоративної інфраструктури. Приклади таких подій: підключення користувачів, маніпуляції з їхніми обліковими записами та правами, доступ до критичних даних та виконання критичних операцій.

1.2 Огляд літератури

Федеративне машинне навчання (FML) описує структуру машинного навчання, яка забезпечує створення спільної моделі на основі даних, доступних у репозиторіях, що належать різним організаціям чи пристроям, при цьому дотримуючи вимоги до конфіденційності та безпеки. Цей стандарт пропонує архітектурні принципи і рекомендації для впровадження федеративного машинного навчання, охоплюючи аспекти, такі як опис, навчання, категоризація, сценарії використання, продуктивність і нормативні вимоги. У сучасних умовах робота з великими даними потребує об’єднання інформації з різних джерел для створення моделей машинного навчання, що часто суперечить чинному законодавству. Федеративне машинне навчання прагне запропонувати практичне вирішення цієї проблеми, дозволяючи ML-

додаткам використовувати розподілені локальні дані для процесу навчання без необхідності їх передавання між терміналами [10].

Моделі FML невдовзі можуть досягти продуктивності, яка буде еквівалентною або навіть перевершуватиме централізовані ML-обчислювачі, що збирають дані для навчання з усіх кінцевих точок. Цей підхід сприятиме співпраці між власниками джерел даних, забезпечуючи водночас захист конфіденційності та інформаційної безпеки користувачів. Стандартизація дозволить ефективно використовувати розподілені джерела даних без порушення етичних норм чи регуляцій, сприяючи розвитку і вдосконаленню загального інтелекту у середовищах cloud-edge computing.

Compute First Network [11] пропонує платформу федеративного безпечного навчання, спеціально розроблену для постачальників послуг зв'язку. Загальну архітектуру цієї платформи створено з урахуванням ключових вимог. Основуючись на TM Forum Business Process Framework, було розроблено конкретні функціональні можливості, а також впроваджено домен безпеки і криптографічну інфраструктуру. Це забезпечує довірені з'єднання між сторонами зв'язку, гарантуючи взаємну автентифікацію та захист даних.

Федеративне навчання [12, 13] спрямоване на обробку децентралізованих і розподілених даних із забезпеченням їх конфіденційності. Для локального навчання глобальної моделі використовуються конфіденційні дані зі смартфонів. Після цього модель об'єднується для формування оновленої глобальної версії, яка розповсюджується серед багатьох клієнтів.

Як механізм спільного машинного навчання, федеративне дозволяє кільком сторонам розробляти модель без використання даних навчання. Цей механізм виглядає перспективним, оскільки значно підвищує точність розпізнавання і розширює можливості співпраці в таких сферах, як медицина та банківська справа. Це особливо актуально в умовах, коли обмін даними ускладнюється через проблеми безпеки, юридичні, технічні чи етичні аспекти. Проте, традиційне централізоване федеративне навчання має недолік у вигляді центрального сервера, який представляє собою єдину точку відмови. У

відповідь на цю проблему пропонується децентралізований підхід до федеративного навчання під назвою Chain FL. Цей метод використовує блокчейн для делегування відповідальності за зберігання моделі між вузлами мережі, усуваючи залежність від централізованого сервера.

Федеративне навчання виступає як основа глибокого навчання для децентралізованих даних [14]. Сучасні мобільні пристрої мають доступ до конфіденційної інформації, але їх обчислювальні можливості залишаються обмеженими. Edge федеративне навчання (EdgeFed) забезпечує розподілення процесу оновлення локальних моделей, який виконується автономно та одночасно на мобільних пристроях. Дані, отримані від цих пристроїв, інтегруються на Edge-сервері з метою підвищення ефективності навчання та зменшення навантаження на канали зв'язку.

Федеративне навчання з використанням нейронних мереж [15] є темою значного інтересу завдяки здатності оновлювати параметри моделей без необхідності передачі необроблених даних користувачів. Проте, незважаючи на цю перевагу, воно залишається вразливим до різних загроз безпеки, оскільки зловмисники можуть отримувати доступ до конфіденційної інформації учасників через спільні репозиторії. У процесі навчання глибоких нейронних мереж постають два ключових питання: як забезпечити конфіденційність даних користувачів під час навчання і як гарантувати достовірність та цілісність агрегованих результатів на сервері. Для вирішення цих проблем запропоновано VerifyNet – першу перевірену структуру федеративного навчання, яка одночасно зберігає конфіденційність. Зокрема, використовується протокол подвійного маскування, що забезпечує захист локальних градієнтів користувачів у процесі федеративного навчання.

За даними інституту AV-TEST [16], щодня реєструється понад 350 000 нових зразків шкідливих програм та потенційно небажаних додатків (PUA). Для захисту конфіденційності використовуються методи федеративного навчання, що дозволяють проводити аналіз, не порушуючи приватності даних. Виявлення характеристик здійснюється на основі ретельного аналізу

шкідливих програм. Запропонований підхід демонструє надзвичайно високу точність 0,92 при тестуванні на наборі даних, наданому сервісом VirusTotal.

У середовищі хмарних обчислень [17], яке включає віртуальні машини, контейнери та інші технології, епідемія шкідливих програм може істотно зашкодити бізнес-процесам. Запропоновано ієрархічний підхід для виявлення та аналізу шкідливого програмного забезпечення, використовуючи сучасні досягнення у сфері машинного навчання з застосуванням графів, гіперграфів та моделей роботи з природною мовою. Аналіз здійснюється на основі моделей навчання із залученням індуктивного графа та гіперграфа. Розглядається сценарій із взаємодією кількох хмар із різним рівнем вимог до конфіденційності, які співпрацюють для протидії поширенню шкідливих програм. У цьому контексті пропонується використання федеративного навчання для забезпечення логічних висновків із дотриманням принципів конфіденційності. Обговорюються також ключові питання, пов'язані зі створенням надійних екосистем.

У [18] описано систему федеративного навчання зі збереженням конфіденційності (PPFL), призначену для виявлення шкідливих програм на платформі Android. Ця система дає змогу мобільним пристроям співпрацювати під час навчання класифікатора, не розкриваючи при цьому конфіденційну інформацію про API чи конфігурації дозволів. Реалізація базується на методі федеративного навчання з використанням машин опорних векторів (SVM) та методології безпечних багатосторонніх обчислень. Для оцінювання системи застосовано набір даних про шкідливі програми для Android, створений Національним інститутом інформаційних та комунікаційних технологій (NICT) у Японії. Результати експериментів демонструють переваги запропонованої системи PPFL у порівнянні з централізованими моделями навчання, зокрема вищу продуктивність класифікатора та забезпечення конфіденційності даних додатків у локальних моделях.

В роботі [19] подано ефективну архітектуру на базі федеративного навчання під назвою Fed-IIoT, призначену для виявлення шкідливих програм Android у середовищі IIoT. Fed-IIoT складається з двох основних компонентів: на стороні учасника дані обробляються з використанням двох атак, побудованих на генеративних змагальних мережах (GAN) і федеративних генеративних змагальних мережах (FedGAN). На стороні сервера здійснюється управління глобальною моделлю спільного навчання із її інтеграцією до системи A3GAN (агрегація до мережі GAN). Ключова перевага Fed-IIoT полягає в тому, що пристрої можуть безпечно брати участь у IIoT і взаємодіяти ефективно, не порушуючи конфіденційності. Отримані результати демонструють високу точність запропонованих алгоритмів, а також підтверджують, що захисна стратегія A3GAN забезпечує збереження конфіденційності даних мобільних користувачів Android і перевищує на 8% точність існуючих рішень.

Шкідливе програмне забезпечення активно поширюється на комп'ютери по всьому світу, і багато антивірусних постачальників застосовують сигнатурні методи для його виявлення. Проте швидкість оновлення баз даних сигнатур вірусів завжди відставатиме від темпів появи нових варіантів шкідливого ПЗ. Crystal Security Sandbox (CSS) стежить за виконанням виконуваних файлів Windows і генерує очищений проміжний результат, що використовується для класифікації шкідливого ПЗ. Це новий підхід у сфері виявлення шкідливих програм. Очищений проміжний результат може описати поведінку шкідливого ПЗ, але є надто об'ємним і надлишковим для ефективного вирішення задачі. Тому пропонується його стиснення та оптимізація.

Виконуваний аналіз шкідливого коду в роботі [20] дозволяє створювати метрики для оцінки поведінки зловмисного програмного забезпечення: як саме воно функціонує в операційній системі, які техніки заплутування виконуються, які потоки виконання провокують основну заплановану активність, застосування мережевих операцій, завантаження файлів, збір

системної інформації та доступ до даних. Не менш важливо визначити методи ідентифікації нових зразків шкідливого ПЗ із подібними характеристиками та розробити способи захисту коректних функцій системи. Ручний аналіз для створення сигнатур стає малоефективним через значні витрати часу порівняно з темпами появи та поширення нових зловмисних програм.

У системі [21] реалізовано метод виявлення шкідливих програм, що базується на використанні глибокого навчання API. За допомогою пісочниці Cuckoo та моделі BLSTM отримуються послідовності викликів API шкідливих програм. Методом фільтрації та оптимізації видаляються надлишкові виклики API, залишаючи лише релевантні послідовності. Експерименти проводилися на великому наборі даних, який включав більше 20 тисяч зразків. Результати засвідчили, що модель BLSTM перевершує GRU, BGRU, LSTM і SimpleRNN за показниками продуктивності виявлення шкідливих програм, демонструючи точність на рівні 97,85%.

Стандартні методи виявлення [22] здатні ідентифікувати лише відомі шкідливі програми на основі раніше збережених сигнатур їхнього коду, що робить їх неефективними проти нових загроз на початковому етапі поширення. У випадках, коли дані про нові шкідливі зразки не надходять і бібліотека вірусних кодів не оновлюється, користувачі залишаються без відповідного захисту. Як альтернативу запропоновано новий підхід до виявлення загроз шляхом аналізу журналу подій у реальному часі. Для моделювання дій користувача та моніторингу реакцій додатків використовується пісочниця, яка активує процес деактивації нових шкідливих програм. Ефективність цього методу була перевірена під час експериментів, де використовувалися як нові шкідливі, так і нешкідливі програми, для оцінки показників продуктивності та точності їхньої ідентифікації.

У роботах [23–26] представлено моделі та методи виявлення шкідливих кодів у контексті федеративного машинного навчання, а також виконано аналітичний огляд сучасних технологій для пошуку вірусів.

1.3 Висновки та постановка завдання

Таким чином, тенденції з кіберзахисту на найближчі кілька років, що визначені аналітиками компанії Gartner передбачають середовище, в якому більша децентралізація, посилене регулювання та наслідки для безпеки будуть більш серйозними вже до кінця 2025 року. Отже, розробка програмно-апаратних рішень, пов’язаних з комп’ютингом кіберзахисту, залишається актуальною задачею для комп’ютерних інженерів.

Мета роботи – підвищення якості розпізнавання та спрощення аналізу malware шляхом розробки автомату пошуку шкідливих компонентів у кіберпросторі у межах архітектури комп’ютингу деструктивних кодів.

Задачі: огляд сучасних тенденцій розвитку кіберпростору та літератури; аналіз інфраструктури – моделей, методів, архітектур, технологій – для ідентифікації шкідливих програм; розробка автомату пошуку шкідливих кодів у кіберпросторі; створення програмного застосунку.

2 ІНФРАСТРУКТУРА ДЛЯ ВИЯВЛЕННЯ ШКІДЛИВОГО КОДУ

Аналізуються моделі, методи, архітектури, технології для ідентифікації шкідливих програм, а також засобів програмування для подальшої розробки автомату виявлення шкідливих кодів.

2.1 Комп’ютинг шкідливого коду

Пісочниця являє собою ізольоване модельне середовище для перевірки та контролю виконання підозрілого коду. Вона створена для аналізу та ідентифікації шкідливої активності програм з метою запобігання їх запуску у зовнішньому середовищі. Виходячи з цього визначення, далі розглядаються структури пісочниць, орієнтовані на використання кіберпростору як інтегрованого елемента систем захисту.

Наступна архітектура комп'ютингу (рис. 2.1) [27] є однією з найактуальніших для впровадження, згідно з проведеним аналізом сучасних технологій захисту кіберпростору від шкідливого програмного забезпечення. Концепція FL-комп'ютингу, запропонована компанією Google, полягає у використанні спільної масової моделі машинного навчання для покращення точності та продуктивності, при цьому зберігаючи конфіденційність даних кожного користувача.

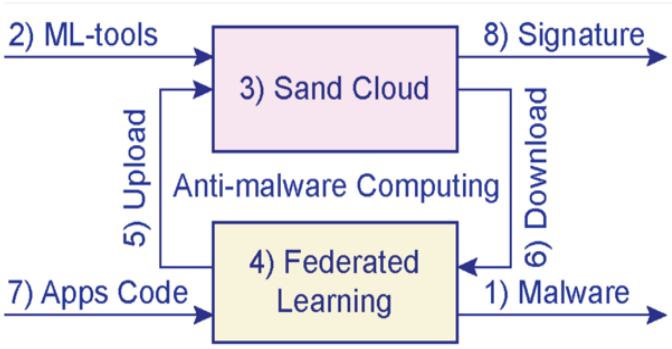


Рисунок 2.1 – Федеративний комп’ютинг машинного навчання на периферії хмари (Cloud-edge federated ML-computing) [27]

Подана архітектура cloud-edge обчислень (рис. 2.2) для алгоритму федеративного навчання, який функціонує за допомогою ітераційного процесу, що складається з чотирьох етапів: Training – локальне навчання; Upload – завантаження параметрів до хмарної пісочниці; Aggregating – агрегування цих параметрів у хмарі; Download – завантаження їх на термінали.

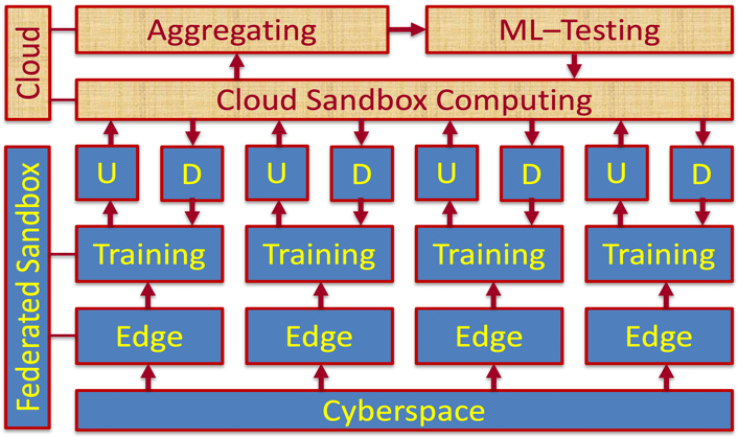


Рисунок 2.2 – Обчислення для виявлення шкідливого програмного забезпечення в хмарній пісочниці [27]

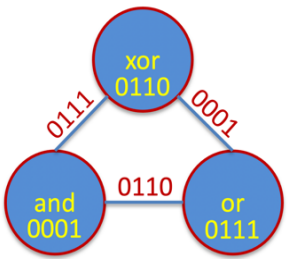
2.2 Способи подання та виміри кіберфізичних процесів

У комп’ютингу застосовуються три основні способи опису процесів і явищ: табличний, аналітичний і графовий [27-30]. При цьому таблиця та вектор є двома формами опису моделей, що можуть легко переходити одна в одну. Вектор виступає компактним варіантом таблиці істинності, представлений як впорядкована послідовність станів виходу за умови, що вхідні компоненти адреси впорядковані за зростанням. У випадках необхідності матриця може бути перетворена в одновимірний вектор, що робить її більш придатною для паралельної обробки даних у регістровій пам'яті. Відновлення таблиці чи матриці з векторної форми відбувається доволі просто. Вектор розглядається як компактний та технологічний спосіб опису об'єктів, функцій та структур у контексті Security computing.

Метрика вимірювання процесів і явищ у дискретному (двійковому) просторі базується на трьох аксіомах: рефлексивності, симетричності та транзитивності. Ці аксіоми визначають циклічну або замкнуту взаємодію між трьома компонентами: 1, 2 та 3. Ключову роль відіграє аксіома конволюції простору процесів і явищ. Вона формулюється так: "Xor-відношення відстаней між кінцевою кількістю компонентів, замкнутих у циклі, завжди дорівнює нулю" [28]:

$$\oplus_{i=1}^{n} d_i = 0. \tag{2.1}$$

Цілком природно, що XOR-сума відстаней між замкнутими функціональними об'єктами, які подані у векторному вигляді, також дорівнює нулю. Водночас, це ж справедливо і для XOR-суми кубітних векторів функцій, які представлені як об'єкти (and = 0001, or = 0111, xor = 0110), що показано на рис. 2.3, також дорівнює нулю [28].



xor⊕and⊕or=
=0000
xor=and⊕or
and=xor⊕or
or=xor⊕and

X_1	X_2	xor	and	or
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	1

Рисунок 2.3 – Схема конволюції логічних функцій

Отже, можна стверджувати, що xor-відношення між двома зазначеними функціями співпадає з третьою функцією. Разом усі три функції (Xor, And, Or), об'єднані через операцію xor-відношення, дають результат, еквівалентний нулю:

$$X \oplus A \oplus O = 0. \tag{2.2}$$

У лемі конволюції трьох функцій від двох змінних наголошується на існуванні лише чотирьох пар логічних функцій, пов’язаних відношенням XOR, які у підсумку формують XOR-функцію. Це твердження описує чіткий зв’язок між логічними операціями, демонструючи їхню властивість зберігати операцію XOR у певних умовах:

1	2	3	4
$\oplus$=0110	$\oplus$=0110	$\oplus$=0110	$\oplus$=0110
$\wedge$=0001	$\overline{\vee}$=1000	X_1=0011	$\overline{X}_1$=1100
$\vee$=0111	$\overline{\wedge}$=1110	X_2=0101	$\overline{X}_2$=1010

(2.3)

Перші два варіанти у (2.3) можна розглядати як пари логічних функцій, взаємодія яких формує xor-функцію. Решта належать до вироджених функцій, які залежать лише від однієї змінної і здатні створювати xor-операцію. З огляду на те, що другий варіант є інверсією першого, можна дійти висновку: лише дві логічні функції, а саме кон'юнкція (and) та диз'юнкція (or), мають здатність породжувати xor-вектор у процесі xor-взаємодії.

Практична цінність леми про конволюцію функцій полягає в наступному: 1) оператор xor виступає як універсальний і унікальний інструмент для оцінки схожості та відмінностей між будь-якими кіберфізичними процесами та явищами; 2) лише дві логічні операції (and і or) у своїй взаємній відмінності утворюють операцію xor; 3) дві базові функції від однієї змінної (інверсія та повторення) також забезпечують реалізацію xor-взаємодії; 4) тріада логічних функцій (and, or, xor) формує векторну метрику, яка використовується для аналізу та обчислення структурних характеристик:

$$S(a,b) = a_i \underset{i=1,n}{\wedge} b_i, \quad D(a,b) = a_i \underset{i=1,n}{\oplus} b_i \tag{2.4}$$

і нормованих оцінок подібності-відмінності між процесами та явищами

$$S_n(a,b) = \frac{\Sigma_{i=1}^{n}\left(a_i \underset{i=1,n}{\wedge} b_i\right)}{\Sigma_{i=1}^{n}\left(a_i \underset{i=1,n}{\vee} b_i\right)}, \quad D_n(a,b) = \frac{\Sigma_{i=1}^{n}\left(a_i \underset{i=1,n}{\oplus} b_i\right)}{\Sigma_{i=1}^{n}\left(a_i \underset{i=1,n}{\vee} b_i\right)}; \tag{2.5}$$

5) операція диз'юнкції (OR) формує загальну метрику одиничних значень координат, яка використовується для вимірювання норми подібності чи відмінності між двома процесами або явищами; 6) векторна двійкова модель у вигляді сигнатури застосовується для опису шкідливого коду та коректних функціональностей, базуючись на асерційному механізмі. Цей механізм дає змогу здійснювати моніторинг стану програмного продукту в рамках простору та часу. Сигнатура виступає у ролі двійкового ідентифікатора процесу чи явища. Вона генерується шляхом згортки довільно довгої двійкової послідовності в 16-розрядний код із використанням LFSR-регістра зсуву зі зворотними зв'язками. У цьому процесі враховуються розряди 7, 9, 12 і 16, які сумуються за допомогою XOR-елемента, після чого результат передається на вхід регістра [31-34].

2.3 Виявлення шкідливих програм на основі федеративного машинного навчання

У межах технологічної форми опису коректних програм (КП) і деструктивних процесів для алгоритмів і архітектур машинного навчання розглядаються структури даних, зокрема таблиці істинності на термінальній стороні FML.

Для вирішення специфічної задачі пошуку та розпізнавання шкідливих програм (ШП) у синтаксично та семантично незв'язаних кодах слід забезпечити виконання кількох ключових умов. По-перше, необхідно уникати врахування морфологічних взаємозв'язків. По-друге, усі функціональні елементи та шкідливі програми слід розглядати як незалежні компоненти, які разом формують універсум примітивів у межах програмного додатка. Третім критерієм є максимально ефективне використання синтаксичних форматів популярних типів ШП для їх ідентифікації, базуючись на метриках визначення подібності чи відмінностей, зокрема за алгоритмами Левенштейна або Хеммінга. Четвертий пункт передбачає активне застосування відкритих

бібліотек онтологій та правильних кодів як базових еталонів для ідентифікації malware у програмних фрагментах [33]. Додатково, необхідно враховувати взаємозв'язки між компонентами ШП та коректними програмами (КП), впроваджуючи фільтри для посилення або ослаблення кореляцій між ними. Це здійснюється шляхом використання квадратичної матриці подібності-відмінності. Нарешті, слід зважати на специфіку роботи ML-систем для sandbox-аналізу, наприклад, відповідно до представленого варіанту на рис. 2.4.

Federated Machine Learning Data Structures

Splitting	All Data Set = {code – malware}					
	Training Data 80%					Test Data 20%
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Learning: Finding Parameters
Split 1	**Fold 1**	Fold 2	Fold 3	Fold 4	Fold 5	
Split 2	Fold 1	**Fold 2**	Fold 3	Fold 4	Fold 5	
Split 3	Fold 1	Fold 2	**Fold 3**	Fold 4	Fold 5	
Split 4	Fold 1	Fold 2	Fold 3	**Fold 4**	Fold 5	
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	**Fold 5**	
					Final evaluation:	Testing Data

Рисунок 2.4 – ML-система для обробки пісочниці [33]

Такий трудомісткий процес, як процедура split-fold, можна не брати до уваги, враховуючи наявність відкритих бібліотек і семантичних стандартів для точної ідентифікації шкідливих програм (ШП). Отже, оптимальне поєднання наявних технологічних рішень із розробленими алгоритмами дозволяє сформувати ефективну архітектуру обчислювальної системи типу cloud-sandbox для виявлення шкідливого програмного забезпечення за допомогою методів машинного навчання.

Комп'ютинг, спрямований на виявлення та усунення шкідливих кодів, базується на використанні основного характеристичного рівняння тестування:

$$F \oplus T \oplus L = 0. \quad (2.6)$$

Це дозволяє визначати будь-який із компонентів на основі двох інших відомих. У цьому контексті F позначає функціональний код, T – код із

можливими порушеннями, а L – шкідливе програмне забезпечення, яке потребує усунення.

Архітектура комп'ютингу для моніторингу в режимі онлайн та усунення ШП подана на рис. 2.5.

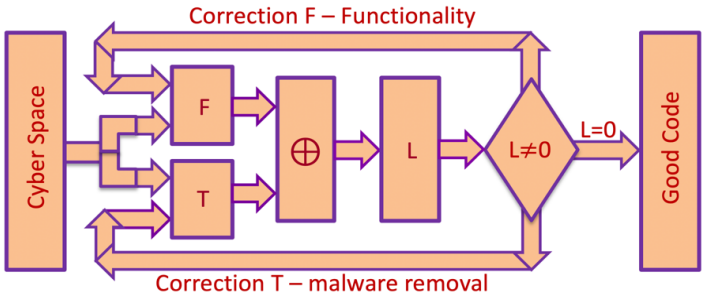


Рисунок 2.5 – Схема комп'ютингу пошуку та усунення шкідливих компонентів

2.4 Виявлення шкідливих програм на основі карти тестування та механізмів векторно-логічного комп’ютингу

Існує повна аналогія процесів modeling та simulation на основі векторно-логічних та інтелектуальних моделей.

Smart model – це розумні структури даних для моделювання будь-якого вхідного набору сигналів для визначення стану виходів без алгоритму.

AI-modeling – це синтез моделі за відомими станами входів-виходів.

AI (forward) simulation – це процес визначення стану виходів за відомими станами входів. Це також робочий режим для формування inference.

AI backsimulation (backpropagation) – це процес визначення стану входів за відомими станами виходів, що застосовується при пошуку помилок та оптимізації смарт-моделі.

Testing map – це карта тестування функціональності, що вирішує всі питання: моделювання та діагностування несправностей, синтезу мінімального тесту та оцінки його якості.

Всі зазначені процеси оперують з трьома компонентами (тест-модель-помилки) рівняння (2.6), серед яких два мають бути відомі для пошуку третього.

Таблиця істинності та її компактна сигнатура – логічний вектор, які є універсальним механізмом на вирішення всіх завдань тестування функцій, структур, шин даних. У будь-якому комп'ютингу це повний набір компонентів, які потребують тестування та верифікації. Логічний вектор – це універсальна модель комп'ютингу для аналізу будь-яких процесів або явищ з метою управління, які цікаві для IT-ринку. На основі логічного вектора будується карта тестування для будь-якої функціональності, в тому числі, і для виявлення деструктивних компонентів.

2.5 Кібератаки та типи загроз

Кібератаки є однією з найсерйозніших загроз сучасного цифрового світу, спрямованих на порушення безпеки, крадіжку даних або завдання шкоди інформаційним системам. Вони охоплюють широкий спектр методів, від простих фішингових атак до складних цілеспрямованих атак на інфраструктуру організацій. Фішинг передбачає надсилання шахрайських листів або повідомлень, які обманом змушують користувачів розкривати конфіденційні дані, такі як паролі чи банківські реквізити. Віруси та трояни, вбудовані в шкідливе програмне забезпечення, можуть заражати системи, викрадати інформацію або блокувати доступ до даних. Програми-вимагачі (ransomware) шифрують файли жертви, вимагаючи викуп за їх розблокування. Атаки типу DDoS (Distributed Denial of Service) перевантажують сервери, роблячи веб-ресурси недоступними для користувачів. У веб-просторі поширені атаки на основі вразливостей, таких як SQL-ін’єкції або міжсайтовий скриптинг (XSS), які дозволяють зловмисникам отримувати доступ до баз даних або виконувати шкідливий код. Обфускований JavaScript, як у функціях eval() або atob(), часто використовується для приховування шкідливого коду на веб-сторінках.

Кібератаки можуть бути спрямовані як на окремих користувачів, так і на великі корпорації чи державні установи, що призводить до значних фінансових і репутаційних втрат. Зростання кількості підключених пристроїв

в Інтернеті речей (IoT) створює нові вразливості, які зловмисники використовують для створення ботнетів.

Соціальна інженерія залишається популярним методом, оскільки вона експлуатує людський фактор, а не лише технічні вразливості. Актуальність кібератак посилюється через швидкий розвиток технологій, що вимагає постійного оновлення методів захисту. Антивірусні програми та веб-сканери, подібні до розробленого в дипломній роботі, відіграють ключову роль у виявленні та нейтралізації таких загроз. Проте зловмисники постійно вдосконалюють свої методи, що робить кібербезпеку динамічною та складною галуззю. Для ефективного протистояння кібератакам необхідна комбінація технічних рішень, освіченості користувачів і своєчасного оновлення захисних систем.

2.6 Архітектура антивірусних додатків

Архітектура антивірусних додатків є складною системою, яка поєднує кілька компонентів для забезпечення ефективного виявлення та нейтралізації загроз. Основним елементом такої архітектури є скануючий модуль, який відповідає за аналіз файлів, веб-ресурсів або мережевого трафіку на наявність шкідливого коду. Цей модуль зазвичай використовує сигнатурний аналіз, порівнюючи вміст із базою даних відомих загроз.

Другий важливий компонент – евристичний аналізатор, який виявляє невідомі загрози на основі поведінкових патернів або аномалій у коді. Наприклад, антивірус може ідентифікувати підозрілу активність, таку як несанкціонований доступ до системних ресурсів. Ще одним ключовим елементом є модуль оновлення, який забезпечує актуальність бази сигнатур і алгоритмів аналізу шляхом регулярного завантаження оновлень із серверів розробника.

Антивірусні додатки часто включають модуль моніторингу в реальному часі, який відстежує активність системи та блокує загрози до їх активації. Для веб-сканування, як у розглянутому застосунку, архітектура може включати

модуль асинхронних HTTP-запитів для швидкої обробки великих обсягів даних. Важливою частиною є також інтерфейс користувача, який забезпечує зручне введення даних і відображення результатів аналізу. Сучасні антивірусні програми часто інтегрують хмарні технології для аналізу великих даних і швидшого реагування на нові загрози. Архітектура може бути модульною, що дозволяє додавати нові функції, наприклад, підтримку аналізу динамічного контенту (рис. 2.6).

Коли користувач вводить URL-адресу в рядок браузера, формується запит до відповідного веб-ресурсу. Сервер обробляє цей запит і надсилає браузеру необхідні файли. Браузер, у свою чергу, інтерпретує ці файли, щоб відобразити запитувану сторінку, дозволяючи користувачу взаємодіяти з нею.

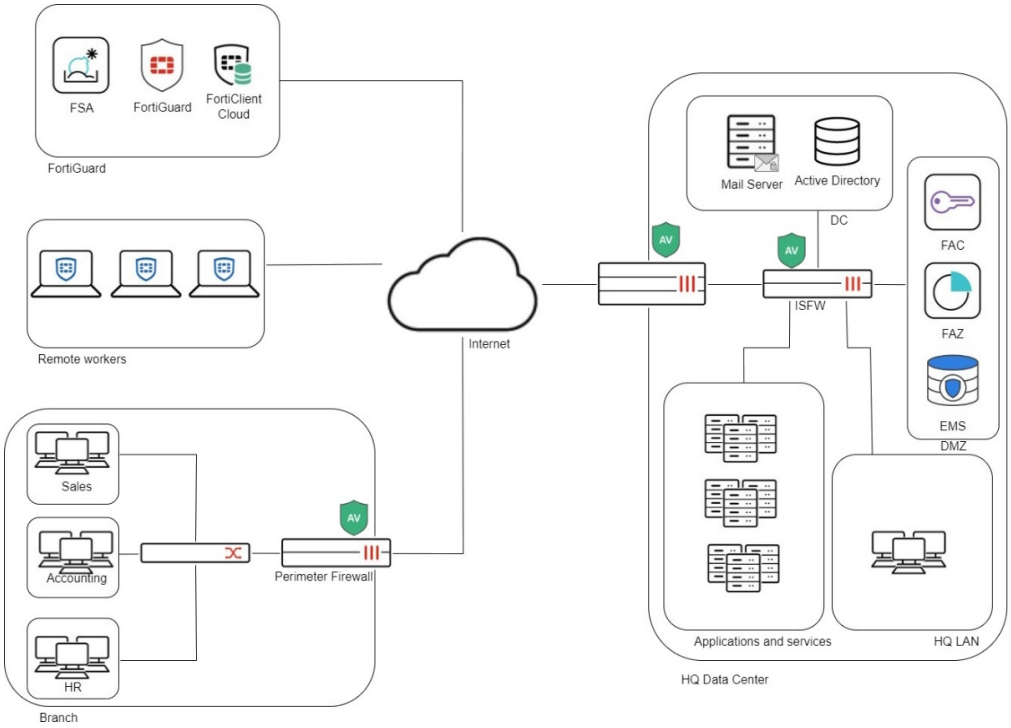


Рисунок 2.6 – Архітектура захисту від кіберзагроз

Крім того, антивірусні додатки враховують обмеження продуктивності, оптимізуючи алгоритми для мінімального впливу на ресурси системи. Безпека самої програми також є важливим аспектом, адже антивірус може стати мішенню атак. Нарешті, архітектура має бути гнучкою, щоб адаптуватися до нових типів загроз, таких як шифрувальники або фішингові атаки.

2.7 Функціональність архітектури антивірусних додатків

Функціональність архітектури антивірусних додатків визначає їхню здатність ефективно виявляти, аналізувати та нейтралізувати загрози в різних середовищах. Основною функцією є сканування вмісту, яке включає аналіз файлів, веб-сторінок або мережевих даних на наявність відомих і невідомих загроз. Сигнатурний аналіз дозволяє швидко ідентифікувати шкідливий код, порівнюючи його з базою даних сигнатур, що регулярно оновлюється. Евристичний аналіз доповнює сигнатурний, виявляючи підозрілу поведінку, наприклад, спроби виконання функцій eval() або document.write() у веб-контенті. Моніторинг у реальному часі забезпечує постійне спостереження за системою, дозволяючи блокувати загрози до їх активації. Для веб-сканерів, таких як розглянутий застосунок, ключовою функцією є асинхронна обробка HTTP-запитів, що забезпечує швидке сканування множини URL-адресів. Антивірусні додатки також надають звіти про результати сканування, які можуть бути представлені у зрозумілому форматі з використанням кольорового виведення, як у бібліотеці colorama. Деякі програми включають функцію карантину, яка ізолює підозрілі файли чи скрипти для подальшого аналізу. Важливою є також можливість інтеграції з хмарними сервісами для аналізу великих обсягів даних або використання машинного навчання для покращення точності. Функція оновлення забезпечує актуальність бази даних і алгоритмів, що критично для реагування на нові загрози. Користувацький інтерфейс дозволяє налаштовувати параметри сканування, наприклад, вибір типів патернів для аналізу. Додаткові функції можуть включати аналіз динамічного контенту, наприклад, JavaScript, який завантажується асинхронно. Антивірусні додатки також можуть надавати статистику про виявлені загрози, що допомагає користувачам оцінити рівень безпеки. Нарешті, функціональність передбачає оптимізацію продуктивності, щоб мінімізувати навантаження на систему під час сканування.

2.8 Аналіз доступних засобів програмування

Порівняння C#, Python та C++ як засобів для написання антивірусів. Мова програмування C# є популярним вибором для розробки антивірусного програмного забезпечення завдяки своїм можливостям і зручності використання. C# працює на платформі .NET, що забезпечує високу продуктивність і доступ до потужних бібліотек для роботи з файловою системою, мережею та криптографією. Ця мова має строгу типізацію, що зменшує ймовірність помилок, пов'язаних із неправильним керуванням пам'яттю, що є критичним для антивірусів, які обробляють великі обсяги даних. C# також підтримує асинхронне програмування, що дозволяє ефективно сканувати файли та реагувати на загрози в реальному часі. Однак C# залежить від .NET Framework або .NET Core, що може обмежувати її використання на не-Windows платформах, хоча сучасні версії .NET підтримують кросплатформність. Крім того, антивіруси, написані на C#, можуть бути дещо повільнішими через накладні витрати на виконання в керованому середовищі.

Python, на відміну від C#, є інтерпретованою мовою, що робить її простішою для швидкої розробки та прототипування антивірусних рішень. Завдяки великій кількості бібліотек, таких як PyCryptodome для криптографії чи psutil для моніторингу системи, Python дозволяє створювати антивіруси з мінімальними зусиллями. Його синтаксис є простим і зрозумілим, що полегшує підтримку коду. Однак Python має суттєві недоліки для антивірусного програмування: через свою інтерпретовану природу він значно повільніший, ніж компільовані мови, що критично для сканування великих обсягів даних у реальному часі. Крім того, Python менш ефективний у керуванні пам'яттю, що може призводити до підвищеного споживання ресурсів при обробці складних загроз, таких як поліморфні віруси.

C++ є однією з найпотужніших мов для створення антивірусного програмного забезпечення завдяки своїй швидкості та низькорівневому доступу до апаратних ресурсів. Ця мова дозволяє оптимізувати

продуктивність, що є ключовим для антивірусів, які потребують швидкого сканування файлів, аналізу пам’яті та роботи з драйверами. C++ використовується в багатьох комерційних антивірусах, таких як Kaspersky чи ESET, через можливість створення високоефективного коду. Проте розробка на C++ є складнішою через необхідність самостійного керування пам’яттю, що може призводити до помилок, таких як витоки пам’яті чи неправильна обробка вказівників. Крім того, розробка на C++ займає більше часу через складніший синтаксис і відсутність вбудованих засобів для швидкого прототипування.

Порівнюючи ці мови, C# пропонує баланс між продуктивністю та зручністю розробки, що робить її хорошим вибором для антивірусів, які працюють у керованому середовищі Windows. Python найкраще підходить для прототипування або створення допоміжних інструментів, таких як аналізатори логів чи скрипти для автоматизації, але не для основного ядра антивірусу. C++ залишається найкращим вибором для створення високопродуктивних антивірусних двигунів, але вимагає значних зусиль і досвіду. Вибір мови залежить від конкретних вимог до антивірусу: продуктивність (C++), швидкість розробки (Python) чи універсальність (C#).

Порівняння архітектурних патернів проектування додатків для захисту від зовнішніх атак. Архітектурний патерн «Шари» (Layered Architecture) є одним із найпоширеніших підходів до проектування додатків із захистом від зовнішніх атак. У цій архітектурі система поділяється на шари, такі як презентаційний, бізнес-логіки та доступу до даних. Кожен шар виконує чітко визначену функцію, а взаємодія між шарами суворо регламентована. Це забезпечує ізоляцію компонентів, що ускладнює зловмисникам доступ до критичних частин системи, наприклад, бази даних. Шар безпеки, який включає автентифікацію, авторизацію та фільтрацію вхідних даних, часто розміщується між презентаційним шаром і зовнішнім середовищем. Перевагою цього патерну є простота реалізації та масштабування, але він може бути вразливим до атак, якщо міжшарова ізоляція недостатньо строга, наприклад, при неправильній валідації даних.

Мікросервісна архітектура набуває популярності в системах, що потребують захисту від зовнішніх атак, завдяки своїй децентралізованій природі. У цій архітектурі додаток розбивається на невеликі незалежні сервіси, кожен із яких відповідає за окрему функцію. Наприклад, один мікросервіс може відповідати за автентифікацію, інший – за обробку запитів користувача. Це дозволяє ізолювати вразливі компоненти та обмежити наслідки атаки: компрометація одного сервісу не обов’язково призведе до компрометації всієї системи. Мікросервіси також полегшують використання контейнерів (наприклад, Docker) і оркестраторів (Kubernetes), що додають додаткові шари безпеки, такі як мережева сегментація. Однак мікросервіси ускладнюють управління системою через велику кількість компонентів, а також потребують ретельного налаштування міжсервісної взаємодії, щоб уникнути вразливостей, таких як незахищені API.

Патерн «Захищений конвеєр» (Secure Pipeline) фокусується на безпечній обробці даних від моменту їх надходження до системи до їх збереження чи передачі. У цьому підході кожен етап обробки даних (вхід, обробка, зберігання, вивід) захищається окремими механізмами безпеки, такими як шифрування, токенізація чи перевірка цілісності. Наприклад, вхідні дані проходять через фільтр валідації для захисту від ін’єкцій, а вихідні дані шифруються перед передачею. Цей патерн особливо ефективний для захисту від атак типу SQL-ін’єкцій, XSS чи CSRF. Його перевагою є модульність і можливість інтеграції нових механізмів безпеки без зміни основної архітектури. Однак реалізація захищеного конвеєра може бути складною через необхідність ретельного налаштування кожного етапу обробки, а також може знижувати продуктивність системи через додаткові перевірки.

Порівнюючи ці патерни, «Шари» є найпростішим для реалізації та підходить для монолітних додатків, але може бути менш гнучким у складних системах. Мікросервісна архітектура забезпечує високу ізоляцію та стійкість до атак, але вимагає значних ресурсів для управління. «Захищений конвеєр» ідеально підходить для систем, де пріоритетом є безпека даних на всіх етапах

обробки, але його складність може бути виправданою лише для висококритичних додатків. Вибір патерну залежить від масштабу системи, вимог до безпеки та доступних ресурсів для розробки та підтримки.

2.9 Висновки до розділу 2

Виконано аналіз сучасних підходів, інфраструктури – моделей, методів, архітектур, технологій – для ідентифікації шкідливих програм, технології cloud-edge computing, а також федеративні алгоритми та архітектури машинного навчання, механізми векторно-логічного комп’ютингу, що ґрунтуються на реакціях функціональності та malware на тестові набори даних. Проведено аналіз доступних засобів програмування.

3 АВТОМАТ ВИЯВЛЕННЯ ШКІДЛИВИХ КОДІВ

Розробляється структура автомату пошуку шкідливих компонентів у кіберпросторі у межах архітектури комп’ютингу деструктивних кодів. Реалізується застосунок для аналізу веб-ресурсів для виявлення потенційно шкідливого коду.

3.1 Архітектура комп’ютингу деструктивних кодів

Діагностика спрямована на те, щоб знайти і визначити, які саме шкідливі програми або їх частини заважають комп'ютеру працювати належним чином.

Моделювання для симуляції передбачає створення спеціального середовища (sandbox) з даними, необхідними для проведення діагностичного експерименту над моделлю шкідливого ПЗ та "чистим" пристроєм.

Sandbox (рис. 3.1) – це інструмент для організації тестового діагностичного експерименту. Він включає в себе алгоритми тестування, моделі шкідливого ПЗ у вигляді вхідних даних для тесту, а також реакцію тестованого пристрою (UUT) на ці вхідні дані.

Результатом тестування є сигнатура, представлена у вигляді логічного вектора, отриманого шляхом XOR-порівняння вихідних станів "чистого" комп'ютера та комп'ютера, що перевіряється на наявність шкідливого ПЗ. Цей вектор (вектор сигнатури) вказує на відмінності між станами. Ненульова сигнатура (ідентифікатор шкідливого ПЗ) свідчить про виявлення деструктивних змін у системі.

Процес діагностики полягає у розшифруванні отриманої сигнатури (вектора відгуку) для ідентифікації конкретного місця розташування та типу шкідливого ПЗ. Це здійснюється шляхом зіставлення сигнатури з картою тестування або бібліотекою відомих шкідливих програм.

Карта тестування являє собою спрощену модель виявлення шкідливого ПЗ у функціональних областях комп'ютера, що описується логічним вектором.

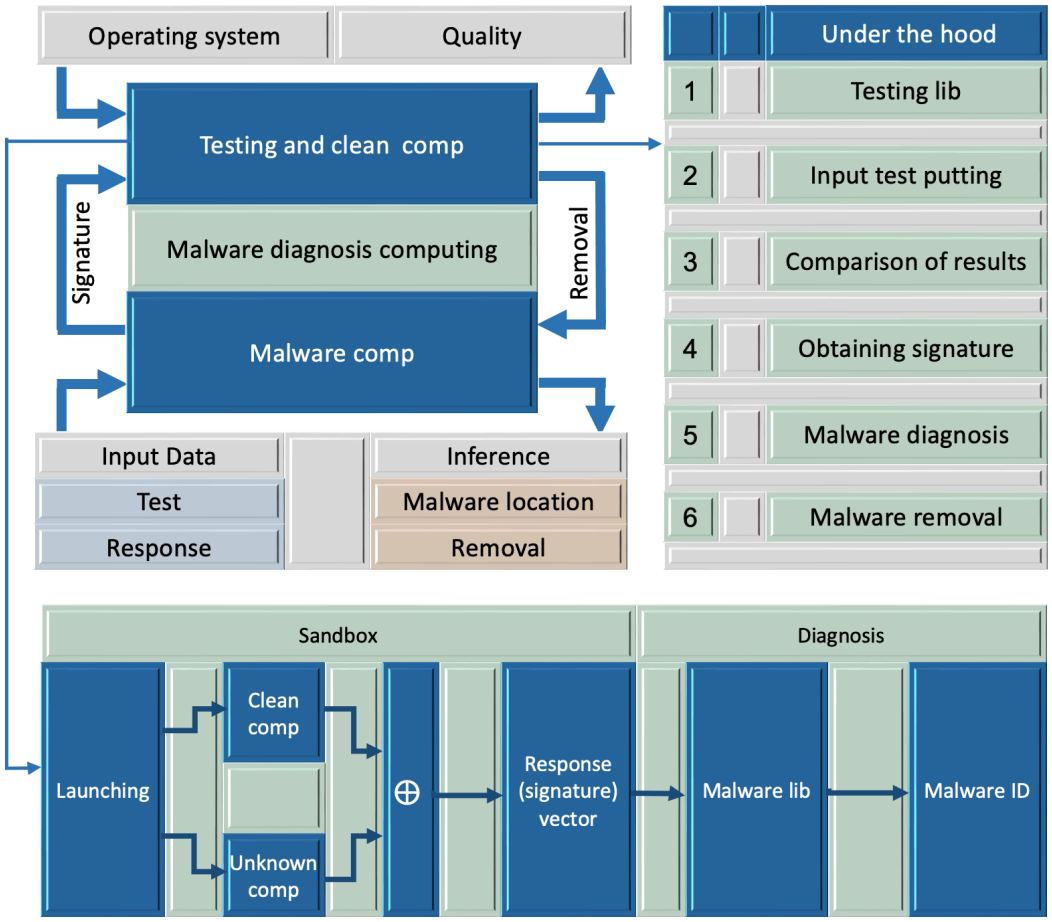


Рисунок 3.1 – Схема комп'ютингу деструктивних кодів

Процедура пошуку деструктивних програм полягає в тестуванні системного програмного продукту тестами верифікації, побудованими розробниками такої програми. Після подачі тесту на заражену та валідну програму виконується порівняння двох об'єктів за критичними точками на основі xor-операції. Якщо результат порівняння дорівнює нулю за всіма координатами векторів, це означає відсутність деструктивних компонентів. В іншому випадку формується вектор порівняння, який є ключем до пошуку таких же деструктивних компонентів у бібліотеці. Якщо такого збігу не сталося, це новий деструктивний компонент і він поповнює бібліотеку деструктивних кодів (рис. 3.2). Для мінімізації обсягів інформації можна використовувати сигнатуру, яка являє собою відбиток скільки завгодно довгої двійкової послідовності. Сигнатурний метод використовується всіма компаніями, які займаються діагностуванням апаратних чи програмних

деструктивних кодів. Останній рядок таблиці – результат порівняння двох програмних об'єктів, мають одиничні значення (табл. 3.1).

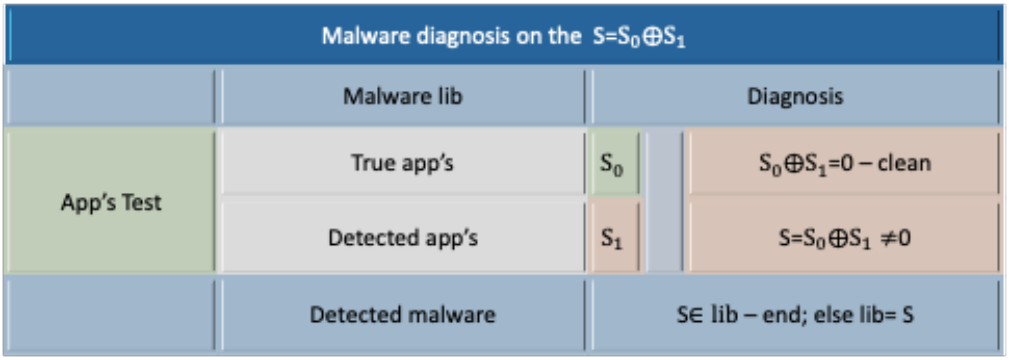


Рисунок 3.2 – Схема визначення та діагностики malware

Таблиця 3.1 – Формування бібліотеки malware

	Malware lib														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1			1				0				0		1		0
2				1		0		0		1		0	1	0		
3		1		0			0	1		0	0	1	1	0		
4				0			0	1		0	0	1				
5				1				0				0		1	1	1
6				1		0		0		1		0	1	0		
7		1		0			0	1		0	0	1	1	0		
8		1		0			0	1		0	0	1			1	
9				1				0				0		1		1
10	1			1		0		0		1		0	1	0		
11			1	0			0	1		0	0	1	1	0		
12			1	0			0	1		0	0	1				
13								0				0		1	1	0
14					1			0				0	1	0		
15		0	0				1	0		1		0	1	0		
16		0	0				1	0				0				
17				1				0				0		1	1	1
18		1		1		0		0		1		0	1	0		
19				0			0	1		0	0	1	1	0	1	
20		1		0			0	1		0	0	1				
21				1				0				0		1		0
22				1		0		0		1		0	1	0		
23		1		0			0	1		0	0	1	1	0	1	
24		1		0			0	1		0	0	1				
25				1	1			0	0		1	0		1		1
26	0	0	1		0	1	0		1	0	0	1	1	0	1	
27			1	0			0	1		0	0	1	1	0		
28			1				0			0	0	1			1	
29								0	0		1	0		1	0	0
30			0		1		1	0	0		1	0	1	0		
31		0	0				1	0	0		1	0	1	0	1	
new		0	0		1		1	0	0	1	1	0		1	0	1

Цей вектор (тут порожні координати – невизначені координати) не має подібності до жодного вектора, який знаходиться в бібліотеці. Тому він формує нову деструкцію та підлягає занесенню до бібліотеки. Слід зазначити, що наявність невизначених координат не дозволяє отримати сигнатуру деструктивного коду, тому тут використовується вектор, який дає змогу встановити місце та тип деструктивного коду у програмі. Як правило, реперні точки, за якими відбувається порівняння, це виходи схеми або програмного продукту, які формують Inference.

3.2 Розробка додатку

Розробляється програмний додаток для аналізу веб-ресурсів на наявність потенційно шкідливого коду, зокрема обфускованого JavaScript або підозрілих патернів, таких як eval() чи document.write(). Додаток призначений для автоматизації сканування веб-сторінок, що дозволяє користувачам, включаючи веб-розробників, адміністраторів і дослідників безпеки, швидко оцінювати безпечність сайтів. Основне завдання полягає у створенні ефективного інструменту, який використовує асинхронні HTTP-запити для обробки великої кількості URL-адресів за короткий час. Додаток аналізує текстовий вміст сторінок за допомогою регулярних виразів, виявляючи підозрілі елементи, такі як base64-кодування або IP-адреси, які можуть вказувати на шкідливий код. Консольний інтерфейс забезпечує зручне введення списку URL-адресів і виведення результатів у форматі з кольоровим маркуванням для легкого сприйняття. Використання бібліотеки aiohttp дозволяє асинхронно обробляти запити, що підвищує продуктивність порівняно з синхронними методами. Додаток розроблено з урахуванням модульності, що полегшує додавання нових функцій, таких як аналіз динамічного контенту чи інтеграція з хмарними сервісами. Програма оптимізована для мінімального використання системних ресурсів, із тайм-аутом запитів у 10 секунд для запобігання зависанням. Обмеження включають

неможливість аналізу нетекстового контенту та залежність від стабільності мережі. У перспективі планується створення графічного інтерфейсу та інтеграція з інструментами для аналізу JavaScript, що виконується динамічно. Проєкт супроводжується документацією, яка описує принципи роботи, обмеження та можливі напрями вдосконалення. Очікується, що додаток стане корисним інструментом для початкового аналізу безпеки веб-ресурсів. Проєкт поєднує сучасні технології програмування та принципи кібербезпеки для створення ефективного рішення.

3.3 Архітектура системи

Архітектура додатка побудована за модульним принципом, що забезпечує гнучкість, масштабованість і зручність у підтримці. Система складається з трьох основних компонентів: модуля введення даних, ядра обробки та модуля виведення результатів. Модуль введення даних дозволяє користувачу вводити список URL-адресів через консоль, завершуючи введення ключовим словом «старт». Ядро обробки, центральний компонент, використовує бібліотеку aiohttp для асинхронного завантаження вмісту веб-сторінок. Аналіз вмісту виконується за допомогою бібліотеки re, яка шукає підозрілі патерни, визначені у словнику, такі як eval(), base64 чи IP-адреси. Модуль виведення результатів форматує дані сканування та відображає їх у консолі з використанням бібліотеки colorama для кольорового маркування: зелене для чистих сторінок, червоне для підозрілих і пурпурове для помилок. Асинхронна обробка через asyncio дозволяє паралельно обробляти кілька URL, що значно скорочує час виконання. Кожен запит обмежено тайм-аутом у 10 секунд, щоб уникнути затримок через недоступні сайти. Архітектура підтримує чіткі інтерфейси між компонентами, що полегшує додавання нових модулів, наприклад, для збереження результатів у файл. Система враховує обмеження продуктивності, оптимізуючи алгоритми для мінімального споживання ресурсів. Обмеженням є відсутність аналізу динамічного контенту, такого як JavaScript, що завантажується асинхронно. У майбутньому

планується інтеграція з хмарними сервісами для обробки великих обсягів даних. Архітектура дозволяє легко оновлювати словник патернів для аналізу нових загроз. Система придатна для використання на стандартних ПК завдяки низьким вимогам до ресурсів. Загалом, архітектура забезпечує ефективну основу для сканування веб-ресурсів.

3.4 Реалізація додатка

Реалізація додатка для аналізу веб-ресурсів на наявність потенційно шкідливого коду передбачає створення ефективного інструменту, який автоматизує сканування веб-сторінок для виявлення підозрілих елементів, таких як обфускований JavaScript чи небезпечні функції. Додаток розроблено на Python завдяки його гнучкості та підтримці потужних бібліотек для роботи з мережею та текстом. Програма використовує бібліотеку aiohttp для асинхронних HTTP-запитів, що дозволяє обробляти велику кількість URL-адресів одночасно. Асинхронна обробка реалізована через бібліотеку asyncio, яка забезпечує паралельне виконання задач, значно скорочуючи час сканування. Для аналізу вмісту сторінок застосовується бібліотека re, яка шукає підозрілі патерни, такі як eval(), document.write() або base64-кодування, визначені у словнику suspicious_patterns. Бібліотека colorama використовується для кольорового форматування результатів у консолі, що полегшує їх сприйняття користувачем. Користувач вводить список URL-адресів через консольний інтерфейс, завершуючи введення словом «старт». Кожен запит обмежено тайм-аутом у 10 секунд, щоб запобігти зависанню при недоступності сайтів. Програма перевіряє тип вмісту сторінки, пропускаючи нетекстові ресурси, такі як зображення чи відео. Аналіз тексту сторінки виконується шляхом пошуку регулярними виразами, а результати виводяться у консоль із зазначенням виявлених патернів або повідомленням про їх відсутність. Додаток оптимізований для низького споживання ресурсів, що дозволяє використовувати його на стандартних ПК. Модульна структура

програми полегшує додавання нових функцій, таких як збереження результатів у файл. Обмеженням є неможливість аналізу динамічного контенту, наприклад, JavaScript, що завантажується асинхронно. У майбутньому планується інтеграція з інструментами, такими як Selenium, для аналізу такого контенту. Розглядається також можливість створення графічного інтерфейсу на основі Tkinter або PyQt для підвищення зручності. Поточна консольна версія є кросплатформною і підходить для технічних користувачів, таких як адміністратори чи розробники. Програма обробляє помилки, наприклад, недоступність сайтів, і відображає їх у зрозумілому форматі. Тестування додатка проведено на наборі URL, що підтвердило його ефективність у виявленні підозрілих патернів. У перспективі планується інтеграція з хмарними сервісами для обробки великих обсягів даних. Загалом, реалізація додатка забезпечує швидке, надійне та модульне рішення для аналізу безпеки веб-ресурсів.

3.5 Висновки до розділу 3

Розроблено структуру автомату пошуку шкідливих компонентів у кіберпросторі у межах архітектури комп’ютингу деструктивних кодів. Реалізовано застосунок для аналізу веб-ресурсів, що виконує завдання виявлення потенційно шкідливого коду, такого як обфускований JavaScript, за допомогою асинхронного сканування.

ВИСНОВКИ

1. Проаналізовано тенденції з кіберзахисту на найближчі кілька років, що визначені аналітиками компанії Gartner передбачають середовище, в якому більша децентралізація, посилене регулювання та наслідки для безпеки будуть більш серйозними вже до кінця 2025 року. Отже, розробка програмно-апаратних рішень, пов’язаних з комп’ютингом кіберзахисту, залишається актуальною задачею для комп’ютерних інженерів.

2. Наведено аналіз сучасних підходів, інфраструктури – моделей, методів, архітектур, технологій – для ідентифікації шкідливих програм, технології cloud-edge computing, а також федеративні алгоритми та архітектури машинного навчання, механізми векторно-логічного комп’ютингу, що ґрунтуються на реакціях функціональності та malware на тестові набори даних.

3. Розроблено автомат пошуку шкідливих компонентів у кіберпросторі у межах архітектури комп’ютингу деструктивних кодів.

4. Реалізовано застосунок для аналізу веб-ресурсів, що виконує завдання виявлення потенційно шкідливого коду, такого як обфускований JavaScript, за допомогою асинхронного сканування. Використання бібліотек aiohttp і asyncio забезпечує високу продуктивність завдяки паралельній обробці множини URL-адресів. Аналіз вмісту сторінок за допомогою регулярних виразів дозволяє ефективно виявляти підозрілі патерни, такі як eval() чи base64. Консольний інтерфейс із кольоровим виведенням, реалізованим через colorama, є зручним для технічних користувачів. Модульна архітектура додатка полегшує його подальше вдосконалення та додавання нових функцій. Обмеженням є неможливість аналізу динамічного контенту та залежність від стабільності мережі. У перспективі планується створення графічного інтерфейсу та інтеграція з інструментами для аналізу JavaScript. Додаток є ефективним інструментом для початкового аналізу безпеки веб-ресурсів. Тестування підтвердило його надійність і точність у виявленні загроз. Проєкт має потенціал для розвитку, зокрема, шляхом інтеграції з хмарними сервісами та розширення словника патернів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Identity Theft Resource Center 2023 Annual Data Breach Report Reveals Record Number of Compromises [Електронний ресурс]. – Режим доступу: https://www.idtheftcenter.org/post/2023-annual-data-breach-report-reveals-record-number-of-compromises-72-percent-increase-over-previous-high.

2. Skills shortage and unpatched systems soar to high-ranking 2030 cyber threats [Електронний ресурс] // ENISA. – Press release від 27.03.2024. – Режим доступу: https://www.enisa.europa.eu/news/skills-shortage-and-unpatched-systems-soar-to-high-ranking-2030-cyber-threats.

3. Kong H. ‘Everyone looked real’: multinational firm’s Hong Kong office loses HK$200 million after scammers stage deepfake video meeting [Електронний ресурс]. – 04.02.2024. – Режим доступу: https://sc.mp/41ucj.

4. ChatGPT – impact of Large Language Models on Law Enforcement [Електронний ресурс]. – 11.06.2024. – Режим доступу: https://www.europol.europa.eu/publications-events/publications/chatgpt-impact-of-large-language-models-law-enforcement.

5. Recommendation on the Ethics of Artificial Intelligence : ЮНЕСКО [Текст]. – SHS/BIO/REC-AIETHICS/2021. – 2021. – 21 p. – Режим доступу: https://unesdoc.unesco.org/ark:/48223/pf0000380455.

6. Artificial Intelligence Act: MEPs adopt landmark law [Електронний ресурс]. – Березень 2024. – Режим доступу: https://www.europarl.europa.eu/news/en/press-room/20240308IPR19015/artificial-intelligence-act-meps-adopt-landmark-law.

7. Cybersecurity Trends: Resilience Through Transformation [Електронний ресурс]. – Режим доступу: https://www.gartner.com/en/cybersecurity/topics/cybersecurity-trends.

8. Gartner Top 10 Strategic Technology Trends 2023 [Електронний ресурс]. – Режим доступу: https://www.gartner.com/en/articles/gartner-top-10-strategic-technology-trends-for-2023.

9. The Top 8 Cybersecurity Predictions for 2021–2022 [Електронний ресурс]. – Режим доступу: https://www.gartner.com/en/articles/the-top-8-cybersecurity-predictions-for-2021-2022.

10. IEEE Guide for Architectural Framework and Application of Federated Machine Learning : IEEE Std 3652.1-2020. – [Effective from 2021]. – New York : IEEE, 2021. – 69 p. – DOI: 10.1109/IEEESTD.2022.93.

11. Operation and Security Consideration of Federated Learning Platform Based on Compute First Network [Text] / [L. Zhao, X. Tang, Z. You et al.] // 2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops) : proceedings. – IEEE, 2020. – P. 117–121. – DOI: 10.1109/ICCCWorkshops49972.2020.9209951.

12. Heralding the Future of Federated Learning Framework: Architecture, Tools and Future Directions [Text] / S. K. Das, S. Bebortta // 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence) : proceedings. – IEEE, 2021. – P. 698–703. – DOI: 10.1109/Confluence51648.2021.9377066.

13. Chain FL: Decentralized Federated Machine Learning via Blockchain [Text] / [C. Korkmaz, H. E. Kocas, A. Uysal et al.] // 2020 Second International Conference on Blockchain Computing and Applications (BCCA) : proceedings. – IEEE, 2020. – P. 140–146. – DOI: 10.1109/BCCA50787.2020.9274451.

14. Federated Learning System [Text] / [Y. Ye, S. Li, F. Liu et al.] // IEEE Access. – 2020. – Vol. 8. – P. 209191–209198. – 10.1109/ACCESS.2020.3038287.

15. VerifyNet: Secure and Verifiable Federated Learning [Text] / G. Xu, H. Li, S. Liu, K. Yang, X. Lin // IEEE Transactions on Information Forensics and Security. – 2020. – Vol. 15. – P. 911–926. – DOI: 10.1109/TIFS.2019.2929409.

16. Using Federated Learning on Malware Classification [Text] / K.-Y. Lin, W.-R. Huang // 2020 22nd International Conference on Advanced Communication Technology (ICACT), Phoenix Park, Korea (South) : proceedings. – IEEE, 2020. – P. 585–589. – DOI: 10.23919/ICACT48636.2020.9061261.

17. Towards Deep Federated Defenses of Malware in Cloud Ecosystems [Text] / J. Payne, A. Kundu // 2019 1st IEEE International Conference on Trust,

Privacy and Security in Intelligent Systems and Applications (TPS-ISA) : proceedings. – IEEE, 2019. – P. 92–100. – 10.1109/TPS-ISA48467.2019.00020.

18. A Privacy-Preserving Federated Learning System for Android Malware Detection Based on Edge Computing [Text] / R.-H. Hsu et al. // 2020 15th Asia Joint Conference on Information Security (AsiaJCIS) : proceedings. – IEEE, 2020. – P. 128–136. – DOI: 10.1109/AsiaJCIS50894.2020.00031.

19. FED-IIoT: A Robust Federated Malware Detection Architecture in Industrial IoT [Text] / [R. Taheri, M. Shojafar, M. Alazab et al.] // IEEE Transactions on Industrial Informatics. – 2020. – DOI: 10.1109/TII.2020.3043458.

20. AOS: An Optimized Sandbox Method Used in Behavior-Based Malware Detection [Text] / [H. J. Li, C.-W. Tien, C.-H. Lin et al.] // 2011 International Conference on Machine Learning and Cybernetics, Guilin, China: proceedings. – IEEE, 2011. – P. 404–409. – DOI: 10.1109/ICMLC.2011.6016683.

21. Malware Automatic Analysis [Text] / CAB d. Andrade, CG d. Mello, JC Duarte // 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence: proceedings. – IEEE, 2013. – P. 681–686. – DOI: 10.1109/BRICS-CCI-CBIC.2013.119.

22. A Robust Malware Detection System Using Deep Learning on API Calls [Text] / Y. Liu, Y. Wang // 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) : proceedings. – IEEE, 2019. – P. 1456–1460. – DOI: 10.1109/ITNEC.2019.8728992.

23. Моделі і методи пошуку шкідливих кодів на архітектурі федеративного машинного навчання [Текст] / В. І. Хаханов, А. С. Саприкін // Міжнародний науковий журнал “Modern Scientific Researches”. – 2021. – Вип. 16. – 18 с.

24. Сучасні технології пошуку вірусів, федеративні алгоритми та архітектури (аналітичний огляд) [Текст] / О. С. Саприкін // Радіоелектроніка та інформатика. – 2020. – № 4. – 16 с.

25. Federated Machine Learning Architecture for Searching Malware [Text] / V. I. Hahanov, A. S. Saprykin // East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 10–13 September 2021 : proceedings. – IEEE, 2021. – P. 1–4. – DOI: 10.1109/ewdts52692.2021.9581000.

26. Malware Searching Methods at FML-Architecture [Text] / V. I. Hahanov, A. S. Saprykin // East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 10–13 September 2021 : proceedings. – IEEE, 2021. – P. 1–5. – DOI: 10.1109/ewdts52692.2021.9581024.

27. Mobile Malware Detection in Sandbox with Live Event Feeding and Log Pattern Analysis [Text] / W. Lin, J. Pan // 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS) : proceedings. – IEEE, 2016. – P. 1–6. – DOI: 10.1109/APNOMS.2016.7737204.

28. Hahanov V. Cyber Physical Computing for IoT-driven Services [Text] / V. Hahanov. – New York : Springer, 2018. – 279 p.

29. Qubit Fault Detection in SoC Logic [Text] / [M. Karavay, V. Hahanov, E. Litvinova et al.] // 2019 IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia : proceedings. – IEEE, 2019. – P. 1–7.

30. Qubit Technology for Analysis and Diagnosis of Digital Devices [Text] / V. I. Hahanov, Bani Amer Tamer, S. V. Chumachenko, E. I. Litvinova // Electronic Modeling. – 2015. – Vol. 37, No. 3. – P. 17–40.

31. Big Data Driven Cyber Analytic System [Text] / [V. Hahanov, E. Litvinova, W. Gharibi et al.] // IEEE International Congress on Big Data, New York City : proceedings. – IEEE, 2015. – P. 615–622.

32. Comparison of Binary and LFSR Counters and Efficient LFSR Decoding Algorithm [Text] / [A. Ajane, P. M. Furth, E. E. Johnson et al.] // 2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS) : proceedings. – IEEE, 2011. – P. 1–4. – DOI: 10.1109/MWSCAS.2011.6026392.

33. Study and Analysis of LFSR Architectures [Text] / [R. Oommen, M. K. George, S. Joseph] // 2018 International Conference on Systems in Digital Enterprise Technology (ICCSDET) : proceedings. – IEEE, 2018. – P. 1–6. – DOI: 10.1109/ICCSDET.2018.8821227.

34. Evaluation of Feature and Signature Based Training Approaches for Malware Classification Using Autoencoders [Text] / [S. S. Tirumala, M. R. Valluri, D. Nanadigam] // 2020 International Conference on Communication Systems & NETworkS (COMSNETS) : proceedings. – IEEE, 2020. – P. 1–5. – DOI: 10.1109/COMSNETS48256.2020.9027373.

35. Aiohttp Documentation [Електронний ресурс]. – Режим доступу: https://docs.aiohttp.org/en/stable/.

36. Colorama Documentation [Електронний ресурс]. – Режим доступу: https://pypi.org/project/colorama/.

37. Python Regular Expressions (re) Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/re.html.

38. OWASP Web Security Testing Guide [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-web-security-testing-guide/.

39. Design Patterns: Elements of Reusable Object-Oriented Software [Text] / J. Gamble, R. Helm, R. Johnson, J. Vlissides. – Boston : Addison-Wesley Professional, 2021.

40. Asyncio Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/asyncio.