Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: «Автоматизований аналізатор шкідливого ПЗ на основі sandbox-технології»

Виконавець: студент IV курсу, групи КБ-41

_____ **Григорій ПОНОМАРЕНКО**
(підпис) (ім'я, прізвище)

	Підпис	**Ім'я, прізвище**
Керівник		Олександр ТОРОШАНКО

Нормоконтроль		Юрій БАБЕНКО

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студенту **КБ-41** **Пономаренко Григорію Григоровичу**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Автоматизований аналізатор шкідливого ПЗ на основі sandbox-технології

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Cuckoo Sandbox, Sysmon, YARA, Elasticsearch, поведінковий аналіз

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитись та проаналізувати принципи роботи sandbox-техноло-
гій та існуючі рішення динамічного аналізу. Сформулювати основні недоліки базових аналізаторів та обгрунтувати напрямки удосконалень. Розробити та реалізувати розширену інфраструктуру на базі Cuckoo Sandbox з інтеграцією

YARA, Sysmon, Elasticsearch та провести тестування системи після покращень.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Вдосконалене середовище динамічного аналізу на основі Sandbox-технології

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видав _____ Олександр ТОРОШАНКО
(підпис) (ім'я, прізвище)

Завдання прийняв до виконання _____ Григорій ПОНОМАРЕНКО
(підпис) (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 27.01.2025	*виконано*
2	Аналіз літератури	28.01.2025 – 18.02.2025	*виконано*
3	Обґрунтування вибору рішення	19.02.2025 – 26.02.2025	*виконано*
4	Теоретичні основи аналізу шкідливого ПЗ за допомогою sandbox-технологій	27.02.2025 – 19.03.2025	*виконано*
5	Аналіз проблеми вдосконалення sandbox-аналізатора	20.03.2025 – 14.04.2025	*виконано*
6	Покращення модулів аналізатору	15.04.2025 – 08.05.2025	*виконано*
7	Тестування та оцінка роботи аналізатора, порівняння результа-тів до та після покращення	09.04.2025 – 26.05.2025	*виконано*
8	Оформлення пояснювальної записки	27.05.2020 – 03.06.2025	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	04.06.2025 – 13.06.2025	

Завдання видав _____ Олександр ТОРОШАНКО
(підпис) (ім'я, прізвище)

Завдання прийняв до виконання _____ Григорій ПОНОМАРЕНКО
(підпис) (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13 червня 2025 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, має 56 сторінок основного тексту, 2 таблиці, 15 рисунків. Список використаних джерел містить 18 найменувань та займає 2 сторінки.

Методи дослідження кваліфікаційної роботи:

- аналіз науково-технічної літератури;
- моделювання інфраструктури аналізу;
- експериментальне тестування;
- порівняння результатів.

Об’єктом дослідження є процес виявлення та аналізу шкідливого ПЗ.

Предметом дослідження є технологічні інструменти динамічного аналізу шкідливого ПЗ із використанням sandbox-середовищ, зокрема Cuckoo Sandbox та супутніх рішень.

У кваліфікаційній роботі досліджено принципи роботи систем динамічного аналізу, здійснено огляд сучасних інструментів (як відкритих, так і комерційних), визначено ключові обмеження існуючих sandbox-рішень. Практична частина роботи присвячена процесу покращення системи на основі Cuckoo Sandbox: встановлено та налаштовано середовище, реалізовано інтеграцію Sysmon, YARA-Rules, Elasticsearch та Kibana для розширеного аналізу та візуалізації результатів.

Запропонована інфраструктура дозволяє автоматично обробляти зразки шкідливого ПЗ, здійснювати сигнатурний і поведінковий аналіз, а також наочно представити отримані дані. Проведено тестування роботи системи до та після вдосконалень, що підтвердило її ефективність.

Результати роботи можуть бути використані для навчання, проведення досліджень у сфері кібербезпеки, а також як основа для впровадження аналітичних рішень у корпоративному середовищі.

Ключові слова: шкідливе ПЗ, динамічний аналіз, sandbox, Cuckoo Sandbox, YARA, Sysmon, Elasticsearch, Kibana, візуалізація.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПЗ	–	Програмне забезпечення
IP	–	Internet Protocol
P2P	–	Peer-to-peer
RAT	–	Remote Access Trojan
ОС	–	Операційна система
BSOD	–	Blue Screen of Death
C&C	–	Command and Control Server
DDoS	–	Distributed Denial-of-Service
API	–	Application Programming Interface
APT	–	Advanced Persistent Threat
C2	–	Command and Control
SOC	–	Security Operations Center
YARA	–	Yet Another Recursive Acronym
IOC	–	Indicators of Compromise
ACPI	–	Advanced Configuration and Power Interface
CPUID	–	Central Processing Unit Identification
DLL	–	Dynamic-Link Library
SIEM	–	Security Information and Event Management
CLI	–	Command Line Interface

ЗМІСТ

ВСТУП

Актуальність даної роботи визначається тим, що в умовах стрімкої цифровізації суспільства кіберзагрози стають дедалі більш витонченими та небезпечними. Сучасні користувачі, компанії та державні установи щоденно стикаються з ризиком зараження комп'ютерних систем шкідливим програмним забезпеченням, яке здатне викрадати дані, порушувати цілісність інформаційних ресурсів або знищувати їх. Класичні засоби захисту, як, наприклад, антивірусні рішення на основі сигнатур, не завжди в змозі своєчасно виявляти нові, ще невідомі типи загроз. У даному контексті особливу актуальність набуває використання sandbox-технологій для динамічного аналізу поведінки підозрілих об'єктів в ізольованому середовищі.

З огляду на зростання кількості атак з використанням складного шкідливого ПЗ, важливо розробляти інструменти, які дозволяють автоматизувати процес виявлення таких загроз та мінімізувати вплив людського фактору. Sandbox-технології створюють умови для безпечного запуску потенційно шкідливих файлів і аналізу їхньої активності без ризику для основної системи.

Аналіз останніх досліджень та літератури. До науковців, що досліджують методи аналізу шкідливого ПЗ та застосування sandbox-технологій, належать Джоел Янг, Карсон Цу, Олівер Келлер. Щодо українських дослідників, то можна виокремити Кузьменка О.О., Барабаша І.І., Плотніков В.І – вони розглядали питання поведінкового аналізу програмного забезпечення та побудови систем автоматичного виявлення загроз.

Метою роботи є покращення і підвищення якості та результатів детектування шкідливого ПЗ з використанням sandbox-технології на основі їх поведінкового аналізу.

Для досягнення поставленої мети необхідно розглянути такі питання:

- Розглянути класифікацію та властивості сучасного шкідливого програмного забезпечення;
- Ознайомитись з принципами роботи sandbox-технологій;
- Проаналізувати наявні рішення у сфері динамічного аналізу ПЗ;
- Проаналізувати структуру та функціональні модулі автоматизованого аналізатора задля подальшого впровадження покращень;
- Проведення тестування роботи аналізатора.

Об'єктом дослідження є процес виявлення та аналізу шкідливого програмного забезпечення в ізольованому середовищі.

Предметом дослідження є методи автоматизованого динамічного аналізу ПЗ за допомогою sandbox-підходів.

Методи дослідження кваліфікаційної роботи бакалавра:

- аналіз літератури;
- аналіз документів;
- порівняльний аналіз існуючих інструментів;
- перевірка ефективності запропонованого підходу.

РОЗДІЛ 1

Теоретичні основи аналізу шкідливого ПЗ за допомогою sandbox-технологій

1.1 Класифікація та характеристика сучасного шкідливого програмного забезпечення

Шкідливе програмне забезпечення, або «малвар», як його ще називають, є одним з ключових викликів сучасної кібербезпеки і, по своїй суті, являє будь-яке програмне забезпечення, навмисно зроблене для спричинення збоїв у роботі комп'ютера, сервера, клієнта або комп'ютерної мережі, витоку конфіденційної інформації, отримання несанкціонованого доступу до інформації чи системи, позбавлення доступу до інформації або такого, що несвідомо втручається в безпеку та конфіденційність комп'ютера користувача.

Два найбільш поширених способи «підхопити» шкідливе ПЗ в нашій системі – це Інтернет та електронна пошта. Тож, по суті, щоразу, коли ми підключені до Інтернету, ми вразливі. Задля подальшого захисту нашої системи також важливо знати як відбувається її зараження. Автори шкідливого ПЗ використовують різноманітні фізичні та віртуальні способи для його поширення. Можна виділити наступні найбільш поширені способи «підхоплення» малвару:

- Знімні накопичувачі: програми шкідники доставляються в систему за допомогою USB-накопичувачів або зовнішнього жорсткого диску. Наприклад, малвар може бути автоматично встановлений, коли заражений знімний накопичувач підключається до ПК.
- Зараженні веб-сайти: шкідливе програмне забезпечення знаходить свій шлях до пристрою через популярні інструменти для співпраці та, наприклад, оновлення, які автоматично завантажують програми зі шкідливих веб-сайтів у системи без схвалення чи відома користувача.

- Фішингові атаки: атаки даного типу базуються на використані фішингових електронних листів, замаскованих під легітимні повідомлення, що містять шкідливі посилання або вкладення, для доставки виконуваного файлу шкідливого ПЗ нічого не підозрюючим користувачам. Складні атаки малвару часто виконуються з використанням серверу командного управління, який дозволяє зловмисникам взаємодіяти із зараженими системами, викрадати конфіденційні дані та навіть дистанційно керувати скомпрометованим пристроєм або сервером.
- Обфускація: нові штами шкідливого ПЗ включають нові методи ухилення та обфускації, розроблені для обману користувачів, адміністраторів безпеки та антивірусних продуктів. Деякі з цих методів ухилення спираються на просту тактику, як використання веб-проксі для приховування шкідливого трафіку або отримання IP-адрес. До складніших кіберзагроз належать поліморфне шкідливе ПЗ, яке може неодноразово змінювати свій базовий код, щоб уникати інструменти виявлення на основі сигнатур; методи боротьби з пісочницею, які дозволяють малвару виявляти, коли його аналізують, і, відповідно, затримувати своє виконання, доки воно не покине пісочницю; наостанок без файлове шкідливе ПЗ – знаходиться лише в оперативній пам’яті системи, щоб уникнути виявлення.
- ПЗ зі сторонніх веб-сайтів/ресурсів: існують випадки, коли шкідливе програмне забезпечення може бути завантажено та встановлено в систему одночасно з іншими програмами або додатками. Зазвичай софт зі сторонніх веб-сайтів або файли, що проходять через P2P мережу, потрапляють до цієї категорію. Наприклад, комп’ютери на основі ОС Microsoft можуть несвідомо встановити ПЗ, яке є потенційно небажаним, навіть не зважаючи на зауваження самої компанії Microsoft.

Щодо самого шкідливого ПЗ, то кожен його вид має свої унікальні риси та характеристики. Загалом вирізняють наступні 11 видів малвару:

- Вірус: найпоширеніший тип шкідника, який може запускатись та поширюватись, заражаючи інші програми та файли.

- Черв’як: може самостійно розмножуватись без програми-носія та зазвичай поширюється без будь-якої взаємодії з авторами шкідливого програмного забезпечення.
- Троянський кінь: розроблений так, щоб виглядати як легітимна програма для отримання доступу до системи. Одразу після активації та встановлення можуть починати виконувати свої шкідливі функції.
- Шпигунське програмне забезпечення: збирає інформацію та дані про пристрій та користувача, а також спостерігає за діяльністю користувача без його відома.
- Програма вимагач: заражає систему користувача та шифрує її дані та файли. Після кіберзлочинці вимагають викуп від жертви в обмін на розшифрування даних системи.
- Руткіт: шкідник отримує доступ адміністратора до системи жертви. Одразу після встановлення програма надає зловмисникам root-доступ або привілейований доступ до системи.
- Вірус-бекдор: вірус-бекдор або ж троян віддаленого доступу (RAT) таємно створює бекдор у заражену комп’ютерну систему, що дозволяє зловмисникам віддалено отримувати до неї доступ, не попереджаючи про це самого користувача або програми безпеки системи.
- Рекламне ПЗ: відстежує історію браузера користувача з метою відображення спливаючих вікон або банерної реклами, яка спонукає користувача до здійснення покупки. Наприклад, рекламодавець може використовувати файли cookie для відстеження веб-сторінок, які відвідує користувач, для кращого таргетування реклами.
- Кейлогери: також називаються системними моніторами. Головна їх задача це відстежування майже всього, що користувач робить на своєму комп’ютері. Це включає написання електронних листів, відкриття веб-сторінок, доступ до програмного забезпечення та натискання клавіш.
- Логічні бомби: тип малвару призначений для заподіяння шкоди та зазвичай вводиться в систему після виконання певних умов. Логічні бомби

залишаються в сплячому стані та спрацьовують, коли виконується певна подія, як, наприклад, виконання користувачем певної дії в певну дату та час.

- Експлойти: даний тип шкідника використовує існуючі вразливості, недоліки або слабкі місця в апаратному чи програмному забезпеченні системи. Замість того, щоб покладатись на тактики соціальної інженерії, вони використовують технічні вразливості для отримання несанкціонованого доступу та виконання інших шкідливих дій, таких як виконання довільного коду всередині системи.

1.2 Методи виявлення шкідливого ПЗ

Виявлення шкідливого ПЗ – це ідентифікація та вивчення програмного забезпечення, що може завдати шкоди комп’ютеру, мережі або іншому забезпеченню. Постійно проводити даний аналіз є доволі затратною задачею як мінімум по часу, тому необхідно вміти бачити прояви аномальної поведінки, які в свою чергу є ознакою, що в нашій системі є малвар. До таких ознак можна віднести наступні:

- Сповільнення швидкості роботи комп’ютера: одним із побічних ефектів шкідливого програмного забезпечення є зниження швидкості ОС. Незалежно від того, чи виходимо ми в Інтернет, чи просто використовуємо локальні програми, використання ресурсів нашої системи виглядає аномально високим. Ми навіть можемо помітити, що вентилятор вашого комп'ютера обертається на повній швидкості – це гарний показник того, що щось займає системні ресурси у фоновому режимі. Це зазвичай трапляється, коли комп'ютер потрапив у ботнет, тобто мережу поневолених комп'ютерів, які використовуються для здійснення DDoS-атак, розсилки спаму або майнінгу криптовалюти.
- Постійно спливаюча реклама: несподівана спливаюча реклама є типовою ознакою зараження шкідливим програмним забезпеченням. Вона особливо пов'язана з формою шкідливого програмного забезпечення, відомою

як рекламне ПЗ. Більше того, спливаючі вікна зазвичай постачаються з іншими прихованими загрозами шкідливого програмного забезпечення. Тож, якщо спливає щось схоже на «Вітаємо, ви виграли безкоштовний телефон!» у спливаючому вікні, натискати на дане повідомлення не можна.

- Аварійне завершення роботи ОС: може бути зависання або BSOD. Оста-

нній виникає в системах Windows після виникнення фатальної помилки.

- Незрозуміла втрата місця на диску: може бути пов'язано з роздутим шкідливим програмним забезпеченням, що ховається на нашому жорсткому диску, також відомим як пакетне програмне забезпечення.

- Незрозуміле збільшення інтернет активності нашої системи: найкраще дану ознаку розглядати на прикладі троянів. Як тільки троян потрапляє на цільовий комп'ютер, наступне, що він робить, це звертається до C&C зловмисника, щоб завантажити вторинний малвар, часто програму-вимагач. Сплеск інтернет-активності дуже часто пояснюється даними діями. Те саме стосується ботнетів, шпигунських програм та будь-якого інших шкідливого ПЗ, яке потребує зв'язку з C&C.

Це лише деякі з багатьох ознак наявності шкідливого ПЗ в нашій системі. Варто відзначити, шо навіть якщо здається, що у нашій системі все працює добре, не варто розслаблятися, адже відсутність новин не обов’язково означає гарні новини. Потужне шкідливе програмне забезпечення може ховатися глибоко у нашій системі, уникаючи виявлення та займаючись своїми брудними справами, не викликаючи жодних тривожних сигналів.

У питанні виявлення малвару важливу роль відіграють і методи виявлення шкідливого ПЗ. Виділяють три основні категорії: на основі сигнатур, на основі аналізу поведінки та на основі евристичного аналізу. Розглянемо детальніше кожну з цих категорій.

1. Аналіз на основі сигнатур

На даний момент цей метод є найбільш традиційним і широко використовуваним. Його суть полягає у зіставлені вмісту файлу або об’єктів з

відомими шаблонами (сигнатурами) шкідливого ПЗ. Сигнатура – це унікальний фрагмент коду, який зустрічається тільки в певному вірусі або шкідливому файлі. Сигнатури створюються на основі унікальних фрагментів коду, структур даних або послідовних байтів, характерних для конкретної програми-шкідника. Дані фрагменти коду містять характеристики, які відрізняють вірус від інших файлів, а також деяку інформацію про поведінку вірусу в системі. Нижче наведе-

но загальний процес виявлення малвару на основі сигнатур.

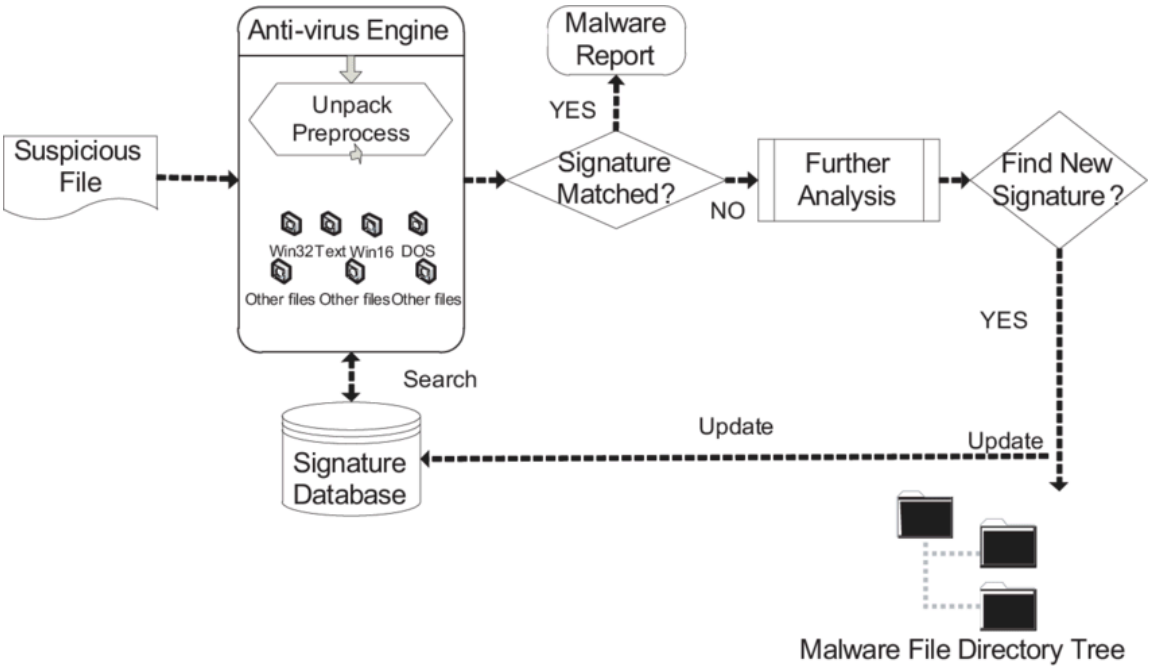


Рисунок 1.2 – Загальний процес виявлення малвару на основі сигнатур

Процес починається з того, що підозрілий файл надходить до антивірусного механізму. Усередині механізму файл розпаковується, попередньо оброблюється та класифікується за типом (Win32, DOS, txt file тощо). Після цього антивірус здійснює пошук збігів між обробленим файлом та наявними запасами у базі даних сигнатур.

Якщо сигнатура виявляється, система формує звіт про виявлення шкідливого ПЗ. У протилежному випадку, якщо відповідна сигнатура не знайдена, то файл йде на подальший, більш глибокий аналіз, у процесі якого є

шанс виявлення нової сигнатури. Якщо нова сигнатура додається до бази даних, то також оновлюється директорія шкідливих файлів, що в майбутньому сприяє ефективнішому виявленню малвару.

Перевагами даного аналізу є висока точність виявлення відомих загроз, низький рівень хибно-позитивних результатів аналізу та ефективність у роботі з великою кількістю файлів. Щодо недоліків, то аналіз на основі сигнатур не є ефективним проти невідомих або модифікованих загроз. Також дуже важливе постійне оновлення баз сигнатур. І останнє – є можливість обходу цього типу аналізу обфускацією та/або шифруванням.

2. Аналіз на основі аналізу поведінки

Методи поведінкового аналізу зосереджені на моніторингу дій та взаємодії шкідливого програмного забезпечення в системі, щоб зрозуміти його поведінку, характеристики та потенційний вплив. На відміну від статичного та динамічного аналізу, які досліджують код та дії шкідливого програмного забезпечення під час виконання, поведінковий аналіз спостерігає, як шкідливе програмне забезпечення взаємодіє зі своїм середовищем, включаючи системні ресурси, файли, процеси та мережевий зв'язок. Щодо поширених методів поведінкового аналізу, то варто відмітити наступні:

- Аналіз трасування виконання: включає фіксацію послідовності дій, що виконуються шкідливим програмним забезпеченням під час його виконання. Це включає створення процесів, взаємодію з файловою системою, зміни реєстру, мережевий зв'язок та інші дії на системному рівні. Аналізуючи трасування виконання, аналітики можуть виявити закономірності, аномалії та шкідливу поведінку, яку демонструє шкідливе програмне забезпечення.
- Моніторинг викликів API: відстежує виклики API, що здійснюються шкідливим програмним забезпеченням для взаємодії з операційною системою та зовнішніми ресурсами. Це включає виклики системних функцій, бібліотек та зовнішніх служб. Відстежуючи виклики API, аналітики можуть виявити підозрілу поведінку, таку як спроби доступу до конфіденційних ресурсів, зміни системних налаштувань або встановлення мережевих з'єднань.

- Моніторинг реєстру та файлової системи: шкідливе ПЗ часто вносить зміни до системного реєстру та файлової системи, щоб зберігати їх після перезавантаження, встановлювати механізми збереження та приховувати свою присутність. Методи поведінкового аналізу включають моніторинг змін у ключах реєстру, значеннях, файлах, каталогах та інших системних ресурсах, внесених шкідливим програмним забезпеченням. Це дозволяє аналітикам виявляти шкідливі модифікації, визначати механізми збереження та відстежувати активність шкідливого програмного забезпечення з часом.
- Метод приманки малвару: даний метод включає розгортання файлів-приманок або пасток для заманювання та спостереження за взаємодією шкідливого програмного забезпечення. Це включає створення приманок або імітованих середовищ, призначених для залучення та захоплення зловмисників. Відстежуючи поведінку шкідливого програмного забезпечення в середовищах-пастках, аналітики можуть збирати розвідувальні дані, виявляти методи атаки та розробляти контрзаходи для зменшення загрози.

Перевагами даного методу є можливість виявлення zero-day загроз та складних атак, аналіз реальної поведінки програми в середовищі виконання та, що дуже важливо, даний метод є стійким до обфускації та поліморфізму. Щодо недоліків, то вони присутні: високе навантаження на систему, висока складність реалізації та обслуговування і можливість виявлення sandbox-середовища з боку шкідливого ПЗ, що призведе до уникнення аналізу.

3. Аналіз на основі евристичного аналізу

Два попередньо розглянуті методи аналізу мають свої недоліки і, щоб їх подолати було впроваджено евристичний метод аналізу. Евристичний метод аналізу – це метод, що базується на перевірці коду на підозрілі ознаки використання правил прийняття рішень. Це спроба виявити шкідливе ПЗ на основі аналізу його структури, логіки або інструкцій без прив’язки до конкретної сигнатури. Наприклад, програма, що намагається переписати завантажувальний сектор, навіть без відповідної сигнатури, може бути класифікована як потенційно небезпечна. Технологія цього аналізу базується на

ідеї, що нові віруси подібні до попередніх. Сам процес виявлення включає три етапи: сканування файлів, аналіз по багатьом критеріям та динамічний аналіз.

Даний метод дозволяє виявити раніше невідоме шкідливе ПЗ, але, як і аналіз на основі аналізу поведінки, все ще має доволі високий відсоток хибних результатів. Також даний метод може потребувати доволі значних обчислювальних ресурсів. Евристичний аналіз погано працює з новим малваром,
написаним, так би мовити, з нуля.

Перевагами даного методу аналізу, як зазначалось раніше, є відсутність потреби в базі сигнатур. Додатково є можливість виявлення модифікованих або обфускованих загроз. Також дуже добре підходить для попереднього сканування підозрілих файлів. Щодо недоліків, то можна виділити наступні: потенційно високі вимоги до обчислювальних ресурсів, складність налаштування евристичних правил для складних середовищ та трохи вищий, ніж у інших методів, рівень хибно-позитивних результатів.

1.3 Принципи роботи динамічного аналізу на основі sand-box системи

У сучасній кібербезпеці загрози шкідливого програмного забезпечення еволюціонували, обходячи традиційні засоби захисту, тому аналіз шкідливого програмного забезпечення є необхідною навичкою для аналітиків безпеки та команд реагування на інциденти. Динамічний аналіз шкідливого програмного забезпечення – це зручний метод викриття складного шкідливого програмного забезпечення шляхом його виконання в ізольованому середовищі для спостереження за його фактичною поведінкою в режимі реального часу.

На відміну від статичного аналізу шкідливого програмного забезпечення, який перевіряє файл без виконання, динамічний аналіз ефективно виявляє загрози нульового дня, APT, поліморфне шкідливе програмне забезпечення,

програми-вимагачі та трояни, які уникають виявлення на основі сигнатур та традиційних антивірусних рішень.

Динамічний аналіз шкідливого ПЗ – це метод виявлення шкідливої активності шляхом запуску шкідливого програмного забезпечення в sandbox середовищі. За допомогою цього методу аналітики можуть бачити, як екземпляр малвару реагує на систему, включаючи модифікації файлів, зміни в реєстрі, мережевий зв'язок та виконання команд.

Sandbox або пісочниця – це безпечне середовище, в якому можна відкривати та запускати файл або URL-посилання. По суті, це спеціально розроблена віртуальна машина, яка запускається перед відкриттям файлу або URL-адреси, а потім вимикається через певний проміжок часу.

Усі дії в пісочниці, такі як файли, що відкриваються або створюються, а також встановлені мережеві підключення, записуються та доступні через логи. Логи можуть допомогти нам зрозуміти, чи був файл або URL-адреса шкідливим. Вони також можуть допомогти нам пов’язати шкідливе програмне забезпечення з раніше поміченою активністю, наприклад, на основі певних мережевих підключень або створених файлів.

Як же відбувається динамічний аналіз на основі sandbox? В цілому можна виділити чотири етапи аналізу. Розглянемо їх детальніше:

1. Збір зразків шкідливого ПЗ

Першим кроком у динамічному аналізі шкідливого програмного забезпечення є збір підозрілих виконуваних файлів або шкідливих файлів з різних джерел, таких як, наприклад, заражені вкладення електронної пошти, шкідливі URL-адреси з функцією автоматичного завантаження, заражені комп'ютери, на яких активно працює шкідливе програмне забезпечення тощо.

Фахівці з безпеки використовують виявлення сигнатур, евристичні методи та аналіз на основі поведінки для виявлення потенційного шкідливого ПЗ, перш ніж воно буде поміщено в карантин в ізольованому контрольованому середовищі.

2. Запуск в пісочниці

Після збору даних шкідливе програмне забезпечення запускається в безпечному, ізольованому середовищі, такому як віртуальна машина або система sandbox. Це робиться для того, щоб запобігти зараженню виробничих систем шкідливим програмним забезпеченням.

Використовувані пісочниці:

- Cuckoo Sandbox
- Any.Run
- Hybrid Analysis
- Falcon Sandbox

Ці пісочниці дозволяють відстежувати поведінку шкідливого ПЗ, взаємодію з системою та маневри ухилення без загрози. Більш детально розглянемо ці використовувані пісочниці у наступному підрозділі.

3. Моніторинг поведінки

Під час роботи кожна активність шкідливого програмного забезпечення в мережі, реєстрі та операційній системі уважно відстежується, а саме наступні процеси:

- Створення та виконання процесів: визначення того, чи завантажується малвар в запущені системні процеси чи створює нові;
- Аналіз мережевої активності: виявлення зв'язку систем управління та контролю (C2), витік даних або спроби завантаження додаткових корисних навантажень; використання таких інструментів, як Wireshark та Netstat для аналізу мережевого трафіку шкідливого ПЗ.
- Модифікації системи: моніторинг змін файлової системи, змін реєстру та запланованих завдань, що призводять до збереження шкідливого ПЗ; використання таких інструментів, як Regshot та Autoruns для керування змінами системи.

4. Звітування

Під час збору даних про поведінку збираються індикатори компрометації для аналізу загроз. Аналітики надають звіти, які включають сигнатури шкідливого програмного забезпечення (наприклад, хеш-значення: MD5,

SHA256); IP-адреси та домени, що використовуються для зв'язку C2; зміни файлової системи (наприклад, ключі реєстру, видалені файли); спроби впровадження процесів та дії з ескалації привілеїв.

Ці звіти допомагають командам кібербезпеки, аналітикам SOC та командам реагування на інциденти розробляти стратегії зменшення наслідків шкідливого програмного забезпечення та покращувати виявлення загроз у режимі реального часу.

З першого погляду може здатись, що у динамічного sandbox аналізу одні переваги, проте необхідно також розуміти й обмеження використання «пісочниці» для шкідливих програм:

- Складне шкідливе програмне забезпечення часто розроблене для виявлення його виконання в пісочниці та зміни його поведінки, щоб уникнути виявлення. Поширені тактики ухилення включають затримку виконання, перевірку артефактів віртуальних машин або вимогу взаємодії з користувачем.
- Пісочниці для шкідливих програм потребують значних обчислювальних ресурсів для створення та підтримки ізольованих середовищ для аналізу. Це включає потребу у віртуальних машинах або контейнерах, пам'яті, сховищі та обчислювальній потужності. Високий попит на ці ресурси може призвести до вузьких місць у продуктивності.
- Шкідливе ПЗ може поводитися по-різному залежно від конкретних конфігурацій системи, версій програмного забезпечення або налаштувань мережі. Якщо пісочниця не точно відтворює цільове середовище, певні моделі поведінки або експлойти можуть не спрацювати, що призведе до неповного аналізу.

1.4 Огляд сучасних інструментів аналізу шкідливого ПЗ з використанням sandbox

У сучасній практиці кібербезпеки існує доволі широке коло інструментів, які реалізують sandbox-підхід для динамічного аналізу шкідливих об’єктів.

Умовно можна класифікувати їх такими ознаками як тип (відкриті або комерційні рішення), рівень автоматизації, типи підтримуваних ОС, глибина аналізу тощо. Розглянемо декілька інструментів більш детально.

1. Cuckoo Sandbox

- Тип: програмне забезпечення з відкритим вихідним кодом;
- ОС: Windows, Linux, MacOS;
- Характеристика: підтримка власних шаблонів, інтеграція з YARA-правилами, Suricata, можливість гнучкого налаштування;
- Переваги: безкоштовний софт, активна спільнота користувачів;
- Недоліки: складність налаштування, застарілі шаблони ОС, обмежена підтримка сучасних версій Windows.

2. Joe Sandbox

- Тип: комерційне програмне забезпечення;
- ОС: Windows, Linux, MacOS, Android;
- Характеристика: глибокий статичний та динамічний аналіз трасування API-викликів, підтримка багатьох форматів файлів;
- Переваги: багатофункціональність, автоматичне генерування звітів, висока точність;
- Недоліки: висока вартість ліцензії.

3. Any.Run

- Тип: інтерактивна хмарна sandbox-платформа;
- ОС: Windows;
- Характеристика: ручне керування віртуальним середовищем під час аналізу, візуалізація мережевих подій, інтеграція з VirusTotal;
- Переваги: зручність для швидкого аналізу «живих» загроз;
- Недоліки: обмежена функціональність безкоштовної версії.

4. FireEye AX

- Тип: комерційне корпоративне рішення;
- ОС: Windows;

- Характеристика: створення IOC, автоматичне виявлення складних загроз, підтримка інтеграції з SOC;
- Переваги: висока надійність, глибока аналітика, підтримка корпоративної інфраструктури;
- Недоліки: складність впровадження, значна вартість.

5. Hybrid Analysis

- Тип: безкоштовна версія з обмеженнями;
- ОС: Windows, Android;
- Характеристика: статичний і динамічний аналіз, автоматичне формува-ння поведінкових профілів, доступ до API;
- Переваги: простота використання, публічний доступ до прикладів аналізу;
- Недоліки: обмеження щодо кастомізації середовища.

Висновки за розділом 1

Аналіз шкідливого ПЗ – фундаментальний процес в сучасній кібербезпеці, який допомагає класифікувати, виявляти та нейтралізувати загрози. Виходячи зі складності та різноманіття малвару, традиційні методи їх виявлення, як сигнатурний аналіз, не завжди є ефективними і не завжди можуть забезпечити своєчасне і, що важливо, точне виявлення нових та/або замаскованих атак. Виходячи з цього, поведінковий та евристичний методи аналізу є дуже важливими, оскільки дозволяють розпізнавати шкідливе ПЗ у випадках, коли відсутні відомі сигнатури.

Поведінковий аналіз базується на запуску підозрілих файлів та/або програм у спеціальних ізольованих середовищах, так званих пісочницях, і є дуже ефективним у процесі виявлення програм-шкідників. У свою чергу запуск підозрілого ПЗ у sandbox дозволяє отримати детальну інформацію про його реальну поведінку та походження. Додатково можна отримати інформації про

створення процесів, змінах в системних файлах, спроби отримання привілеїв вищого рангу тощо.

Незважаючи на свої переваги в поведінкового аналізу є і свої недоліки. Сучасний малвар доволі часто має вбудовану функцію розпізнання його виконання в ізольованому середовищі, що дозволяє змінювати свою поведінку або ж навіть повністю припиняти свою роботу. До того ж даний вид аналізу потребує значних обчислювальних ресурсів і, в додаток, імітація цільового середовища не завжди ідеальна.

В наш час існує багато інструментів для проведення sandbox-аналізу, які можуть задовольнити багато потреб. До цих інструментів відносяться Cuckoo Sandbox, Joe Sandbox, FireEye AX та Any.Run. Відкриті рішення є більш гнучкими до налаштувань, але часто мають обмежену підтримку нових ОС і їх складніше інтегрувати корпоративну інфраструктуру. Комерційні продукти в свою чергу мають більш глибокий функціонал та можливість більш глибокого аналізу. Проте і їх вартість відповідна.

Таким чином можна сказати, що успішна реалізація аналізу шкідливого ПЗ вимагає комплексного підходу, що в свою чергу включає підготовку та збір зразків шкідливого ПЗ, організацію контролю поведінки в пісочниці, аналіз зібраних даних та генерацію звітів, які в подальшому будуть у нагоді фахівцям з безпеки.

РОЗДІЛ 2

Аналіз проблеми вдосконалення sandbox-аналізатора

2.1 Оцінка ефективності існуючих sandbox-рішень

Як вже відомо з першого розділу, sandbox-рішення є дуже важливими та своєрідною основою для глибокого аналізу шкідливого ПЗ. Вони дозволяють досліджувати поведінку підозрілих файлів та програм у спеціально створеному контрольованому середовищі, що, в свою чергу, мінімізує ризики поширення загроз на працюючі систему продуктиву. Проте є одне важливе питання – ефективність sandbox-рішень. Важливо її грамотно оцінити відповідно до необхідних технічних характеристик, налаштувань, реалізації та типу атак проти яких їх будуть застосовувати.

До основних важливих критеріїв ефективності sandbox-системи відносяться наступні 5:

1. Глибина поведінкового аналізу

Суть sandbox – це максимально наблизити створене штучно середовище до реального. Відповідно з цього випливає й умова ефективності – можливість забезпечення повного набору дій виконуваного шкідливого ПЗ, а саме:

- Зміни в файловій системі та реєстрах;
- Мережева активність;
- Взаємодія з API;
- Наявність системних викликів, драйверів та ін’єкцій в інші процеси;
- Поведінкові тригери, як, наприклад, автозавантаження та запуск відкладених програм

2. Надійність в умовах спроб ухилення

Дуже багато сучасного шкідливого ПЗ має можливість розпізнання віртуального середовища і, виходить, що ефективність пісочниці значною

мірою визначається її здатністю залишатись невидимою для малвару. В даному випадку основними є анти-віртуалізація (розпізнання середовища, як віртуальної маши-
ни), анти-емуляція та анти-інструментування (тобто виявлення моніторингових бібліотек, як, наприклад hooking API).

Цікавий факт – згідно дослідження MITRE та AV-Test понад 30 відсотків сучасних шкідливих програм використовують принаймні один анти-sandbox механізм.

3. Гнучкість та кастомізація середовища

Очевидно, що чим більше пісочниця схожа на реальну систему кінцевого користувача тим вище шанс того, що шкідливе ПЗ буде вести себе природно і «нічого не підозрювати». В даному критерії є важливими наступні пункти:

- Можливість зміни мови інтерфейсу, версії ОС тощо; простіше кажучи, присутність повної кастомізації;
- Наявність встановлених популярних програм (офісний пакет, браузери тощо);
- Імітація активності користувача шляхом взаємодії з інтерфейсом або імітацію друку та натискань клавіш;
- Індивідуальне налаштування мережі.

4. Продуктивність та масштабованість

Оскільки в пісочницях щодня може відбуватись аналіз сотень файлів та об’єктів, то важливо, щоб були наявні такі властивості, як швидкий аналіз (до 5 хвилин на один файл/об’єкт), паралельний аналіз, інтеграція з іншими рішеннями (антивіруси, SIEM, Threat Intelligence платформи тощо) та достатня кількість ресурсів аби система могла витримати все покладене на неї навантаження.

5. Якість звітності

Останній, і один з найважливіших критеріїв ефективності – це звітність та фіксування активності. В даному випадку ключовими особистостями будуть генерація чітких та зрозумілих для аналітика звітів та логів, візуалізація

мережевих з’єднань та залежностей, можливість експорту логів у різних форматах та інтеграція з зовнішніми ресурсами як VirusTotal та Suricata, наприклад.

Нижче наведено порівняльний аналіз популярних sandbox-рішень, які ми розглядали у попередньому розділі, у вигляді таблиці:

Таблиця 2.1

Порівняльний аналіз sandbox-рішень

Назва	Тип	ОС	Автоматизація	Глибина аналізу	Недоліки
Cuckoo Sandbox	Відкритий код	Win, Linux	Висока	Середня	Вразливий до анти-evasion, складне налаштування
Joe Sandbox	Комерцій-ний	Win, Linux, Android	Висока	Висока	Дорогий
Any.Run	Хмарний	Win	Середня	Висока	Лімітований функціонал безкоштовної версії
FireEye AX	Корпора-тивний	Win	Висока	Висока	Висока вартість, складне розгортання
Hybrid Analysis	Хмарний	Win, Android	Висока	Середня	Обмежена кастомізація

2.2 Виявленні недоліки у функціонуванні типових аналізаторів

У пункті 2.1 даного розділу ми розглядали критерії ефективності sandbox-рішень. Спираючись на них було проведено аналіз типових систем та аналізаторів. Можна сказати наступне: попри значну кількість переваг, сучасні sandbox-системи мають свої обмеження та недоліки, які в свою чергу пливають

на якість виявлення та дослідження шкідливого ПЗ. Ці недоліки як технічні, так й архітектурні. Далі розглянемо найважливіші з них, спираючись на конкретні приклади ресурсів та платформ, що дасть нам необхідну інформацію та обізнаність для подальших покращень.

Однією з ключових і важливих проблем більшості sandbox-аналізаторів є недостатній рівень симуляції реального користувацького середовища. Багато систем, як, наприклад, Cuckoo Sandbox або базова конфігурація Hybrid Analysis створюють віртуальні машини, які є буквально надто стерильними для ефективного аналізу шкідливого ПЗ. Як згадувалось раніше, середовище не має базового пакету програм і, відповідно, легко виявляється програмами-шкідниками. Наприклад, зразки з родини Ursnif/Gozi перевіряють наявність поштових клієнтів, документів у папці “Documents” або активних мережевих сесій перш ніж почати свою активність. Зразки з родини Emotet, ще відоме як Heodo, також демонструють свою активність лише за умови наявності в системі слідів реальної користувацької діяльності.

Ще однією критичною вразливістю та недоліком є численні технічні характеристики, за якими шкідливе ПЗ може розпізнати, що воно знаходиться у тестовому середовищі. До даних характеристик в першу чергу відносяться ознаки віртуалізації. VMWare, наприклад, часто ідентифікується MAC-адресами з префіксами 00:0C:29 або 00:50:56. VirtualBox же через специфічні записи в ACPI таблицях та процеси VirtualBox.exe. Перевірки на наявність подібного типу ознак дуже легко проходять, якщо не використовуються спеціальні засоби маскування і тоді шкідливе ПЗ або одразу припиняє свої дії, або ж взагалі не виконує нічого. Наприклад зразки QakBot або TrickBot у ранніх стадіях проводять перевірки на віртуальне середовище шляхом звернення до інструкцій CPUID. Якщо результат позитивний, то вони імітують нормальну поведінку. Щодо самих тестових середовищ, то у випадку з Any.Run реалізоване базове маскування, проте більш продвинуті зразки, як раніше згадуваний Emotet, виявляє середовище та завершує свою роботу.

Наступний недолік, який так чи інакше можна прив’язати до попередньої вразливості, це обмеженість моделювання поведінки реального користувача. Як зазначалось раніше, деяке шкідливе ПЗ починає виконуватись лише після певних дій користувача – відкриття певного файлу, натискання на кнопку тощо. Такий малвар, як FormBook та Zloader очікують дій користувача та не починають свою роботу без його взаємодії з інтерфейсом. На даному етапі більшість аналізаторів реалізують лише базові сценарії – рух миші та натискання клавіш. Проте часто ці дії є дуже неприродніми і досвідченні зловмисники додають у шкідливе ПЗ функціонал виявлення цих фальшивих дій, який, в свою чергу, блокує будь-які зловмисні дії програми-шкідника.

Не менш важливою проблемою є обмеження тривалості сесії аналізу. Типова довжина сеансу приблизно від 5 до 15 хвилин, що в цілому є прийнятним для швидкої оцінки. Проте даного часу абсолютно недостатньо, коли шкідливе ПЗ використовує механізми відкладеного виконання. DarkComet, наприклад, реалізує запуск функціоналу лише через пів години. В даному випадку об’єкт банально не демонструє свою активність в проміжку виділених 15-ти хвилин на аналіз. Підтвердженням цьому є дані на відомому публічному сервісі MalwareBazaar, де задокументовано чимало випадків, коли Hybrid Analysis чи Cuckoo Sandbox не виявляли жодних підозрілих ознак того чи іншого шкідливого ПЗ – а все через скорочену тривалість аналізу.

В контексті недоліків варто згадати і проблему масштабованості. Навіть популярні та надійні рішення, як Joe Sandbox та корпоративні, більш продвинуті версії Cuckoo Sandbox, при обробці сотень зразків підозрілих файлів через деякий час втрачають стабільність і починають фальшивити. Це може проявлятись у якості звітів, глибини трасування чи просто у зростанні часу обробки черг. Особливо це критично для SOC підрозділів, де аналітику очікують буквально real-time результатів. Навіть у звітах VirusTotal за 2023 рік вказувалось, що за пікових навантажень обмежуються певні функції – не запускається автоматичний аналіз DLL-модулів, наприклад.

Ще одним доволі значущим недоліком є обмежена та слабка інтеграція з іншими компонентами інформаційної безпеки. Як зазначалось в першому розділі, багато open-source sandbox-рішень не підтримують інтеграцію з платформами як MISP та MITRE ATTACK. Cuckoo Sandbox, наприклад, в базовій версії не підтримує обробку результатів через сторонні SIEM-системи, а Any.Run обмежує доступ до API та звітів, що робить буквально неможливою автоматичну обробку великої кількості підозрілих файлів. У корпоративних же рішеннях така можливість присутня, але потребує ліцензування та доволі складного налаштування.

На останок варто згадати проблему яка займає або першу або другу позицію по важливості – це логування та детальність звітів. Як згадувалось раніше, аналітики SOC підрозділів очікують буквально real-time результатів, які потім аналізуються. В цьому безумовно допомагають і логи. Хоча звіти більшості типових аналізаторів і мають доволі детальні логи, проте ці дані і близько не завжди структуровані належним чином. Наприклад у звітах Any.Run відсутня кореляція подій із MITRE ATTACK, не формуються автоматичні рекомендації, а деякі дії взагалі класифікуються помилково як невідомі. Joe Sandbox в свою чергу створює великі, але не дуже деталізовані звіти, в яких складно виділити критичні події.

Виходячи з даного аналізу типових систем та аналізаторів можна зробити певні висновки. Сучасні sandbox-аналізатори демонструють доволі великі можливості по виявленню програм-шкідників, проте в той же час вони мають і свої недоліки, що значно знижують їх ефективність. Адаптивність до нових можливостей ухилення, масштабованість, звітність – це лише деякі з аспектів, які було розглянуто під час аналізу і у всіх було виявлено ті чи інші недоліки. Даний аналіз та недоліки наштовхують лише на думку, що потрібні вдосконалення, та формують основу для визначення напрямку їх покращення.

2.3 Постановка задачі покращення та обґрунтування вибору до вдосконалення

Аналіз сучасного стану технологій динамічного аналізу шкідливого ПЗ дозволяє зробити висновок, що прогрес у сфері sandbox-технологіях безперечно є і не малий. Проте існує чимало недоліків, які суттєво обмежують ефективність виявлення шкідливих об’єктів у реальних умовах. Особливо це стосується систем з відкритим кодом, таких як Cuckoo Sandbox, які, не дивлячись на їх гнучкість та безкоштовність, дуже часто поступаються комерційним рішенням за глибиною спостереження за поведінкою підозрілих програм та повнотою аналітики.

Основною проблемою базових установок Cuckoo Sandbox є обмежена деталізація поведінкових подій, що фіксуються. Стандартні можливості даного програмного рішення не завжди відстежують такі активності як:

- Приховане створення/видалення файлів;
- Маніпуляції з реєстрами;
- Зміни політик безпеки в системі;
- Спроби підвищення привілеїв.

Ще одна проблема – не зовсім гнучкі звіти. По своїй суті звіти Cuckoo Sandbox технічно насищенні, проте читати їх важко, що в свою чергу дуже сповільнює аналіз, а це вже знижує швидкість реагування на загрозу.

На даному етапі гарною ідеєю є завдання покращення системи Cuckoo Sandbox шляхом інтеграції додаткових інструментів збору поведінкових даних та модулів сигнатурного виявлення, що дозволить підвищити глибину аналізу без необхідності повної перебудови системи чи витрат на комерційне ПЗ.

Після аналізу доступних засобів був обраний наступний варіант:

1. Інтеграція Microsoft Sysmon (System Monitor) – потужний інструмент аудиту подій, який дозволяє розширити можливості фіксації характеристик шкідливого ПЗ. Якщо порівнювати дане рішення зі звичайними логами Windows, то перше надає більш деталізовану інформацію (запуск

процесів, змін до реєстрів, створення файлів та інше). Інтеграція Microsoft Sysmon у віртуальне середовище Cuckoo Sandbox дозволить значно покращити якість аналізу без додаткових витрат, так як є безкоштовним інструментом з відкритим доступом.

2. Підключення системного сигнатурного аналізу YARA – дозволяє автоматично виявляти шкідливі файли та поведінкові шаблони за заданими правилами. Завдяки відкритим джерелам та відкритій базі YARA-Rules можна швидко розширити можливості системи ідентифікувати нові загрози.

3. Ще одне покращення, яке можна реалізувати – вдосконалення читаємості звітів шляхом візуалізація та форматування результатів. Навіть базові графіки часової шкали подій та взаємодії процесів дозволять суттєво краще сприймати отримані результати аналізу. Для цього можна використати вбудовані функції Cuckoo Web Interface або ж встановити Kibana чи Elasticsearch.

Дані покращення дозволять досягти наступних цілей:

- Підвищення якості поведінкового аналізу завдяки більш детальним логам;
- Розширення виявлення шкідливого ПЗ шляхом впровадження YARA-rules;
- Краща інтерпретованість результатів, що є дуже важливим в умовах швидкого прийняття рішень;
- Нова система так і залишиться «безкоштовною» і буде повноцінною альтернативою комерційним продуктам.

У наступному, третьому розділі, буде реалізовано впровадження зазначених змін у систему Cuckoo Sandbox, проведено тестування та задокументовано результати.

Висновки за розділом 2

У другому розділі було проведено комплексне дослідження сучасних підходів до динамічного аналізу шкідливого ПЗ із використанням sandbox-технологій. Було здійснено детальний огляд загальних принципів роботи динамічного аналізу, сучасних методик виявлення шкідливої підозрілої активності та особливості застосування sandbox-рішень у реальному середовищі.

Окрему увагу біло приділено порівнянню відкритих та комерційних рішень. З даного порівняння зроблено висновки: платформи з відкритим кодом, як Cuckoo Sandbox, є гнучкими, масштабованими та безкоштовними, але вони поступаються комерційним рішень, що в цілому і не дивно. Зокрема це стосується глибини аналізу, підтримки схем аналізу (евристичний та сигнатурний), а також в зручності звітів та якості самого аналізу.

Також в ході другого розділі було проведено аналіз типових проблем аналізаторів. До цих проблем відносяться наступні:

- Недостатня симуляція дій користувача;
- Коротка сесія аналізу шкідливого ПЗ;
- Явні ознаки, які дають шкідливому ПЗ зрозуміти, що воно знаходиться у віртуальному середовищі;
- Нечитабельні звіти.

Додатково, на прикладі Cuckoo Sandbox, було спочатку проведено аналіз його функціональних обмежень, а після поставлено задачу покращення. Якщо коротко, то було виявлено неповне та не глибоке логування дій, що відбуваються в середовищі та не реалізоване механізми віртуалізації результатів. З мето підвищення даної платформи було запропоновано наступні напрямки покращення:

- Інтеграція Microsoft Sysmon задля розширення обсягу задокументованих дій, які фіксуються в процесі аналізу.
- Підключення YARA-rules задля можливості автоматизовано ідентифікувати як відомі, так і нові зразки шкідливого ПЗ.

- Покращення віртуалізації звітів для більш легкої подальшої роботи аналітиків.

Оцінюючи результати другого розділу, можна впевнено сказати, що sandbox-технології безумовно важливі, але не досконалі. Їх ефективність залежить в певній мірі залежить від доповнення сторонніми засобами. В наступному, третьому розділі, запропоновані покращення будуть практично впровадженні, відбудеться тестування «до» та «після», а також буде проведена оцінка ефективності нової вдосконаленої системи.

2.4

РОЗДІЛ 3

Розробка та програмна реалізація покращення sandbox-аналізатора шкідливого ПЗ

3.1 Архітектура запропонованого рішення

У даному підрозділі розглядається архітектура удосконаленої, покращеної системи динамічного аналізу шкідливого ПЗ, що базується на відкритому рішенні Cuckoo Sandbox з інтеграцією додаткових інструментів, а саме Microsoft Sysmon, YARA-rules та засобів візуалізації. Основна мета запропонованої архітектури – розширити функціональні можливості системи без істотного ускладнення її інфраструктури, втрати сумісності з базовим функціоналом Cuckoo Sandbox, та, що дуже важливо, з максимально мінімальним втручанням в ядро системи.

Базова архітектура Cuckoo Sandbox передбачає наявність кількох ключових компонентів: host-машини, віртуалізованого середовища аналізу (guest) та модуля звітності. Завантаження шкідливого зразка відбувається на guest-машині, де здійснюється його виконання у контрольованому середовищі, після чого зібрана інформація передається до центрального вузла для обробки та аналізу. У рамках запропонованого покращення ці рівні будуть доповненні раніше зазначеними інструментами та службами:

1. Компонент аудиту подій – Microsoft Sysmon

У середовищі guest, де відбувається запуск потенційно шкідливого ПЗ, буде встановлено Sysmon – системний агент, що входить до пакету Sysinternals Suite. Його головна функція – збір детальних даних про події, пов’язані з поведінкою процесів, створенням/видаленням файлів, змінами в реєстрі, мережевою активністю, спробами підвищення привілеїв тощо.

2. Модуль сигнатурного аналізу – YARA

Для виявлення відомих загроз у процесі динамічного аналізу до системи буде додано підтримку YARA-rules, що дозволить виявляти малвар на основі сигнатур. Cuckoo підтримує інтеграцію з YARA, однак у нашому випадку функцію буде розширено шляхом підключення додаткових баз сигнатур з відкритих репозиторіїв, як, наприклад, GitHub YARA-rules, що в свою чергу дасть нам можливість виявляти більшу кількість сучасного малвару.

3. Модулі візуалізації результатів – Elasticsearch та Kibana

Для покращення читаємості результатів буде впроваджено вивантаження логів з Sysmon та звітів Cuckoo у Elasticsearch. Далі ці дані візуалізуються завдяки Kibana – це інструменти побудови дашбордів, що в свою чергу дає можливість створювати таймлайни, графіки взаємодії процесів, фільтрацію тощо. Це однозначно зменшить час, необхідний для ручного аналізу та дозволить більш оперативно приймати рішення щодо ПЗ.

Важливо також зазначити загальний зв’язок між компонентами для кращого розуміння:

- Збір логів Sysmon здійснюється за допомогою спеціальних агентів, які пересилають дані в Elasticsearch;
- YARA інтегрується з Cuckoo через спеціальні плагіни, які сканують файли до та після їх виконання;
- Користувач взаємодіє з системою через інтерфейс, доповнений Kibana для зручнішої аналітики.

Трохи нижче, на рисунку 3.1, наведено схему архітектури покращеного аналізатора зі зв’язками компонентів.

Дана покращена система дозволяє не тільки виявляти шкідливу активність на основі динамічного аналізу, а також на основі сигнатур та інтерпретувати результати у більш зручному вигляді. Важливо зазначити, що всі використані компоненти є відкритими та безкоштовними, що робить систему придатною для використання в умовах обмеженого бюджету.

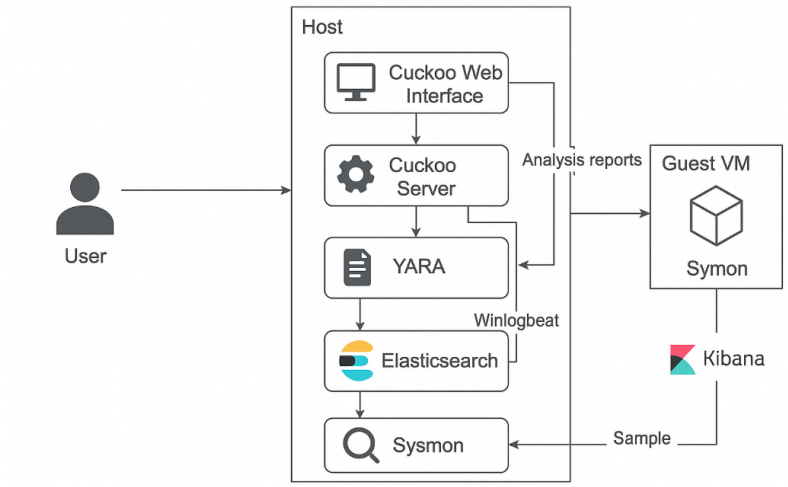


Рисунок 3.1 – Архітектура покращеного аналізатора на базі Cuckoo Sandbox з інтеграцією Sysmon, YARA та Kibana

3.2 Покращення модулів

З метою підвищення ефективності аналізу шкідливого ПЗ в межах середовища Cuckoo Sandbox було проведено ряд технічних вдосконалень, які під собою мають встановлення, налаштування та розширення базового функціоналу модулів систему. У даному підпункті детально описано процес інтеграції інструментів моніторингу, аналізу поведінки, сигнатурного сканування та, що немало важливо, зручної візуалізації результатів. Розглянемо кожен з процесів більш детально.

1. Встановлення та налаштування Cuckoo Sandbox

Першим, і, мабуть, найважливішим кроком було створення спеціалізованого віртуального середовища для Cuckoo на базі операційної системи Ubuntu 20.04. Для забезпечення сумісності з Cuckoo 2.0.7 було обрано Python 2.7, а також, в подальшому, було створено ізольоване середовище, venv. Встановлення відбувалось з офіційного репозиторію. Після цього було встановлено наступні системні бібліотеки та пакети:

- PostgreSQL – основна реляційна БД ля збереження аналітичної інформації;

- MongoDB – допоміжна NoSQL-база для збереження структурованих об’єктів;
- tcpdump та tshark – інструменти для захоплення мережевого трафіку;
- libmagic, libyaml, swig, libssl, libffi – бібліотеки, які забезпечують функціонування окремих обробників;
- git, curl, python-dev, pip – забезпечують взаємодію з репозиторіями та надають можливість керування залежностями.

Після цього були створенні директорії:

- /cuckoo/storage/binaries – збереження файлів, що аналізуються;
- /.cuckoo/conf/ – для конфігурацій;
- /.cuckoo/yara/ - для сигнатур YARA.

Наступний етап – налаштування ключових конфігураційних файлів на яких, по суті, все й тримається:

- cuckoo.conf – основний файл, в якому вказано тип системи віртуалізації;
- processing.conf – обробка результатів аналізу, зокрема активація модуля YARA;
- reporting.conf – модулі збереження результатів, зокрема Elasticsearch.

Усі конфігураційні файли налаштовувались вручну з урахуванням особливостей операційної системи. Результатом даних налаштувань є підняте на localhost наше особисте середовище Cuckoo Sandbox. Рисунок наведено нижче:

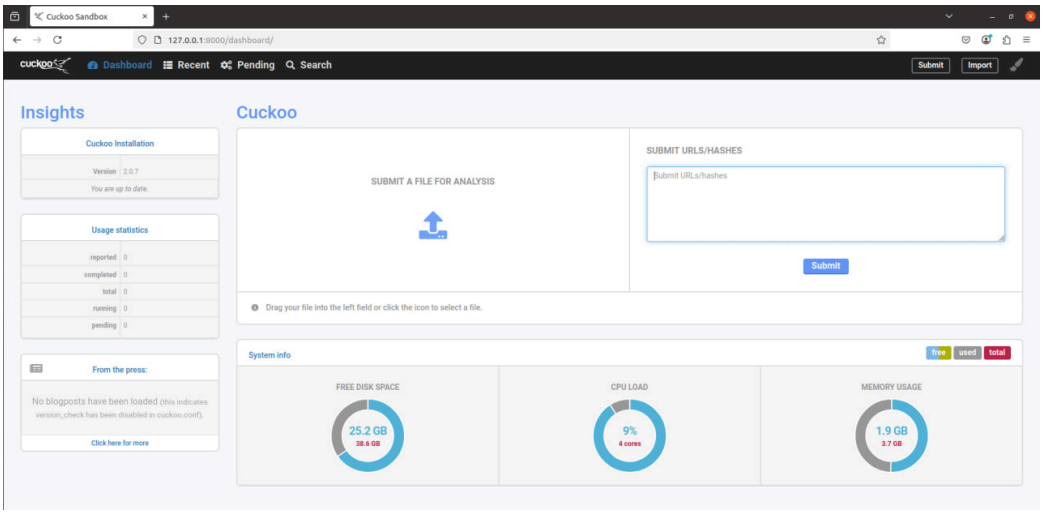


Рисунок 3.2 – середовище Cuckoo Sandbox на localhost

2. Інтеграція YARA правил

Одним з ключових напрямків розширення функціональності стала підтримка YARA – потужного інструменту для сигнатурного виявлення шкідливого ПЗ.

Було створено структуру директорій у /.cuckoo/yara та завантажено актуальні правила з репозиторію YARA-Rules, що на GitHub. Використовувалась наступна команда: git clone https://github.com/Yara-Rules/rules. Для активації даного функціоналу у конфігураційному файлі processing.conf було вручну додано секцію [yarascan]:

```
[yarascan]
enabled = yes
rules_path = /home/greg/.cuckoo/yara
```

Рисунок 3.2 – секція [yarascan] в конфігураційному файлі

Було перевірено наявність .yar та .yara файлів у відповідних каталогах – важливі файли, що і містять YARA-Rules. Додатково було додано можливість створення та інтеграція власних сигнатур. Користувач може власноруч створювати правила, які будуть виявляти шкідливе ПЗ за вказаними властивостями поведінки тощо.

3. Встановлення та налаштування Sysmon

З метою підвищення точності поведінкового аналізу, було розгорнуто утиліту Sysmon на гостьовій Windows-10 машині. Дана утиліта дозволяє фіксувати системні події на рівні ядра – це може бути створення процесів, зміни в реєстрах, відкриття мережевих з’єднань тощо. Для розгортання Sysmon було використано офіційний конфігураційний xml-файл, який можна завантажити з репозиторію SwiftOnSecurity або MITRE. Для встановлення використовувалась наступна команда (попередньо скачали .exe файл Sysmon, поклали конфігураційний xml-файл в папку до .exe файлу та відкрили командний рядок у папці з цими двома файлами): Sysmon64.exe –accepteula –I sysmonconfig-export.xml.

Після цього Sysmon запускався разом з системою, створюючи журнал подій. Ці події зберігаються в стандартний журнал подій Windows: Applications and Services Logs/Microsoft/Windows/Sysmon/Operational. В подальшому цей журнал подій може оброблятись в Cuckoo або виводитись в Elasticsearch для більш детального аналізу. Нижче наведено рисунок зі журналу подій Windows, який зберігає події. На рисунку видно, що джерелом є саме Sysmon.

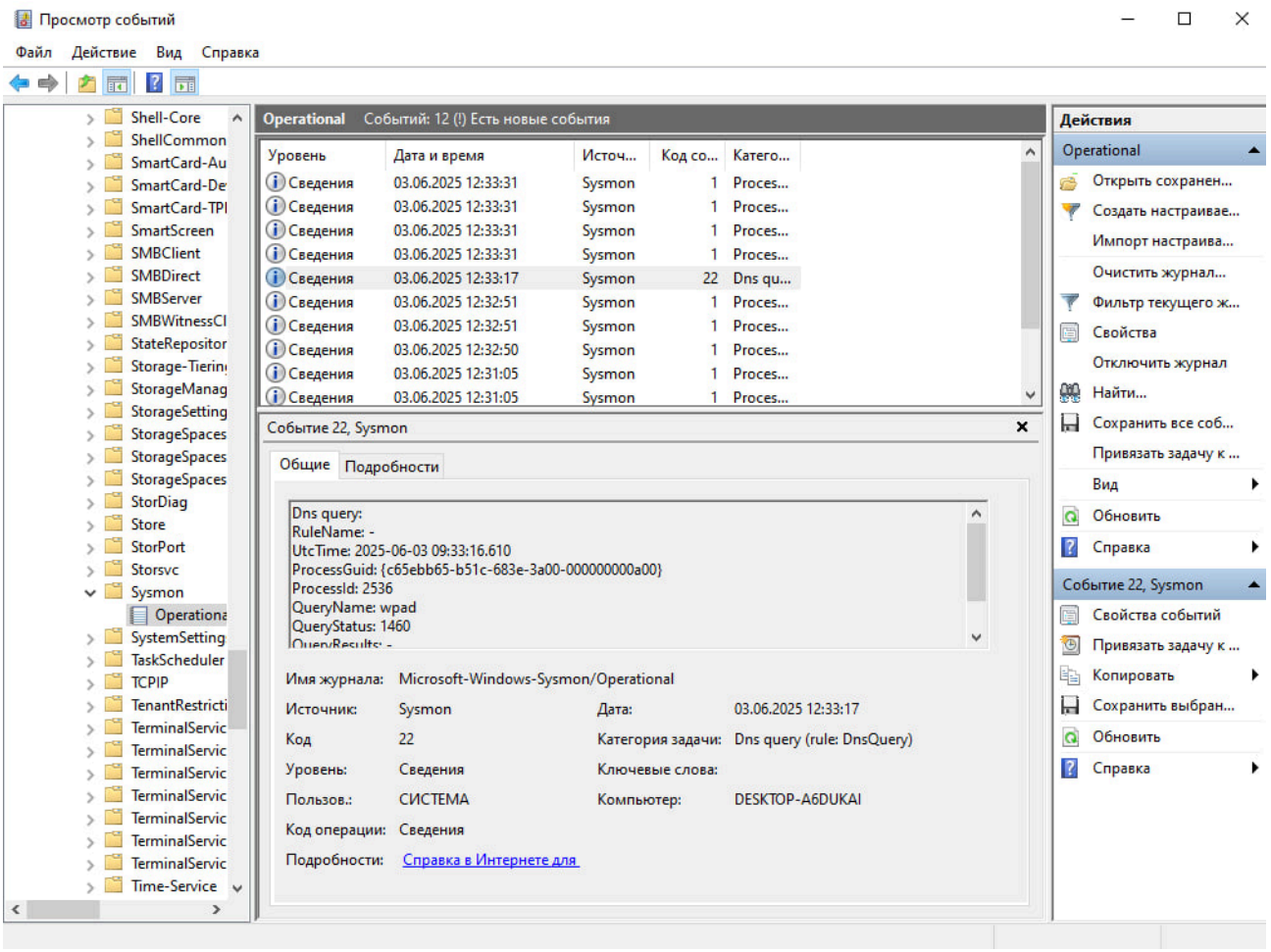


Рисунок 3.4 – журнал подій Windows з інтегрованим Sysmon

4. Підключення Elasticsearch та Kibana

Значним кроком у напрямку візуалізації було підключення та встановлення Elasticsearch та Kibana. Elasticsearch приймає та зберігає структуровані дані з аналізу Cuckoo, а Kibana, в свою чергу, дозволяє будувати дашборди та переглядати результати через веб-інтерфейс виступаючи у ролі візуалізатора.

Було встановлено Elasticsearch та Kibana версії 7.17.28 через офіційні .deb пакети. Після відповідних налаштувань та додавання можливості автоматичних надсилань результатів Cuckoo (відредаговано раніше згадуваний файл reporting.conf) було запущено обидва ресурси. Сам веб-інтерфейс доступний на localhost:5601. Важливо відмітити наступне: Kibana приймає не json, а ndjson файли. Тобто треба автоматизувати конвертування json файлів у ndjson, або інші

формати. Це реалізовано за допомогою простого Python скрипту, який отримує за початкове значення json файл і перетворює його у ndjson. На рисунку нижче наведено web-інтерфейс працюючого Elasticsearch та Kibana на localhost:

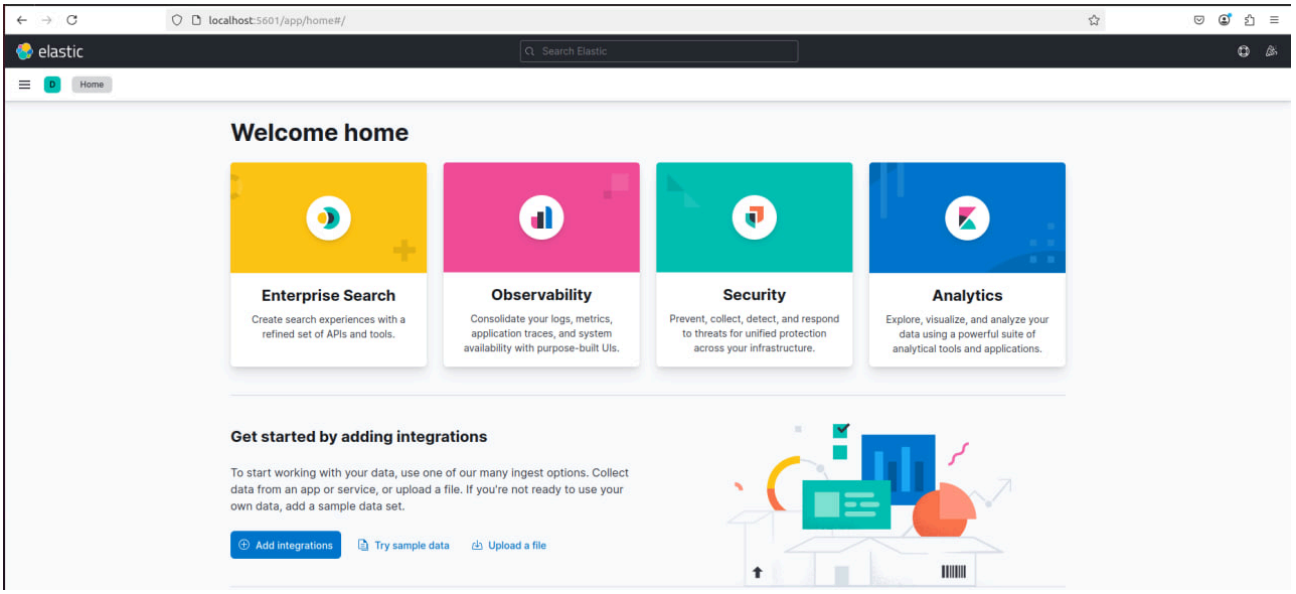


Рисунок 3.5 – Elasticsearch та Kibana на localhost

3.3 Тестування та оцінка роботи аналізатора

У даному підпункті будемо проводити тестування роботи аналізатора на основі Cuckoo та оцінювати його функціональність одразу у двох конфігураціях: із вбудованим аналізом в межах хостової ОС, тобто Linux, а також з інтегрованою гостьовою системою Windows. Кожен із цих двох підходів має свої особливості, обмеження та можливості. Будемо йти по порядку. Перед початком будь-яких тестувань зробимо снапшоп нашого середовища.

Тестування № 1. Аналіз тільки у середовищі Linux

Перший етап – це тестування Cuckoo у мінімальній конфігурації, тобто без підключення гостьової Windows-машини. Такий підхід дозволяє швидко перевірити базову працездатність системи та модуль без складного мережевого налаштування. На час тестування № 1 у файл cuckoo.conf було задано наступні параметри:

```
machinery = none
platform = linux
```

Рисунок 3.6 – параметри cuckoo.conf

Для аналізу використовувався шкідливий файл, взятий зі всесвітньо відомої бази шкідливого ПЗ MalwareBazaar. Можливостей у даного малвару безліч – маніпулювання командним рядком, створення файлів та процесів тощо. Далі цей файл було передано на Cuckoo Sandbox. Система одразу поставила його в чергу зі статусом «pending» та почала аналізувати. Як тільки аналіз завершився ми побачили наступне та отримали можливість переглянути звіт натиснувши на “reported”:

Date	Filename / URL	Package	
03/06/2025 20:09	92bf5c61bcefa13a20a2a49c3d9ecc59451e665df15446f55109e94440fd7cf7.exe @ 92bf5c61bcefa13a20a2a49c3d9ecc59451e665df15446f55109e94440fd7cf7.zip	exe	reported
03/06/2025 20:09	92bf5c61bcefa13a20a2a49c3d9ecc59451e665df15446f55109e94440fd7cf7.zip	7z	reported

Рисунок 3.7 – звіт шкідливого ПЗ в системі Cuckoo

Відкривши звіт по даному файлу можна отримати багато інформації (рис. 3.8, 3.9, 3.10): загальну оцінку шкідливості файлу, його розмір, хеші, YARA-Rules, що відпрацювали тощо.

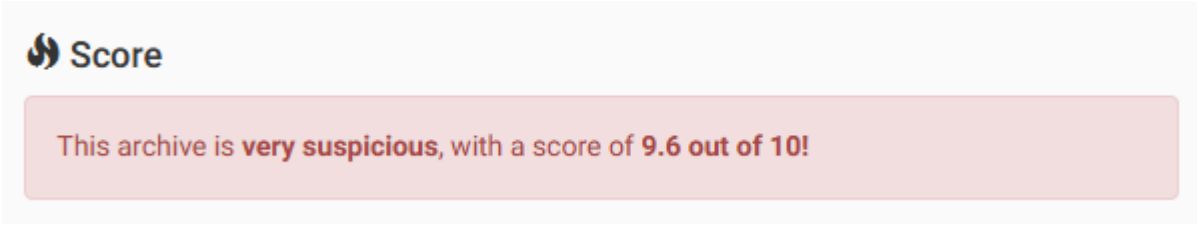
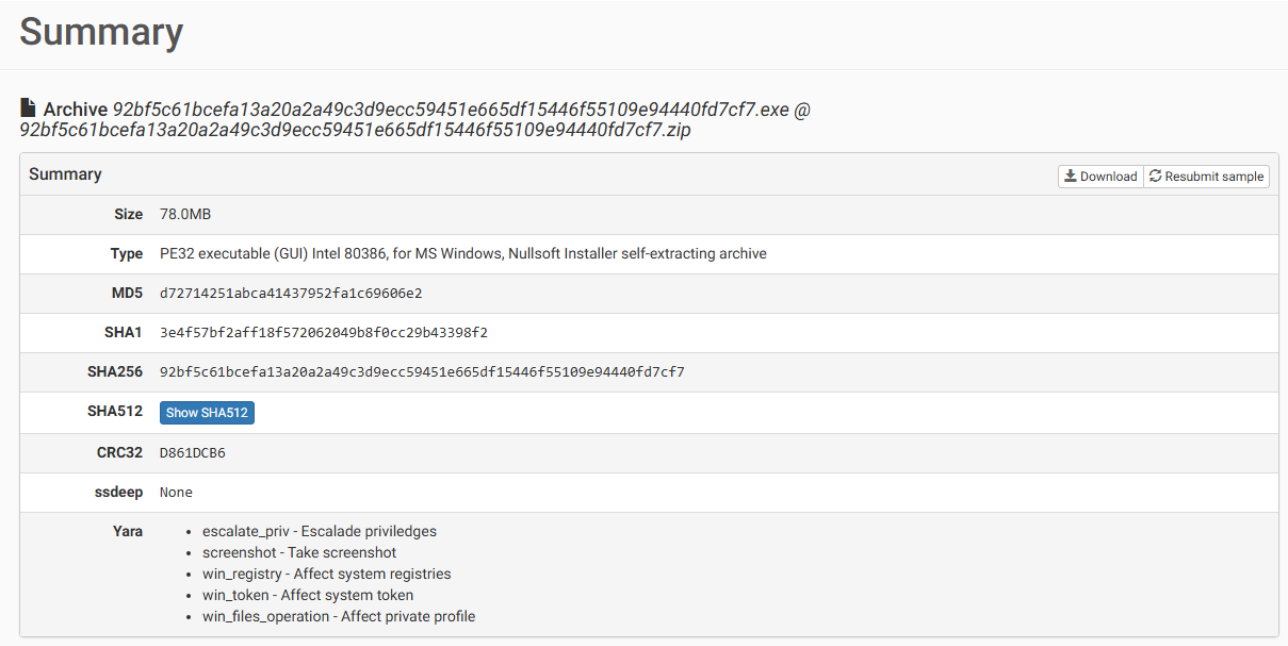


Рисунок 3.8 – загальна оцінка шкідливості файлу



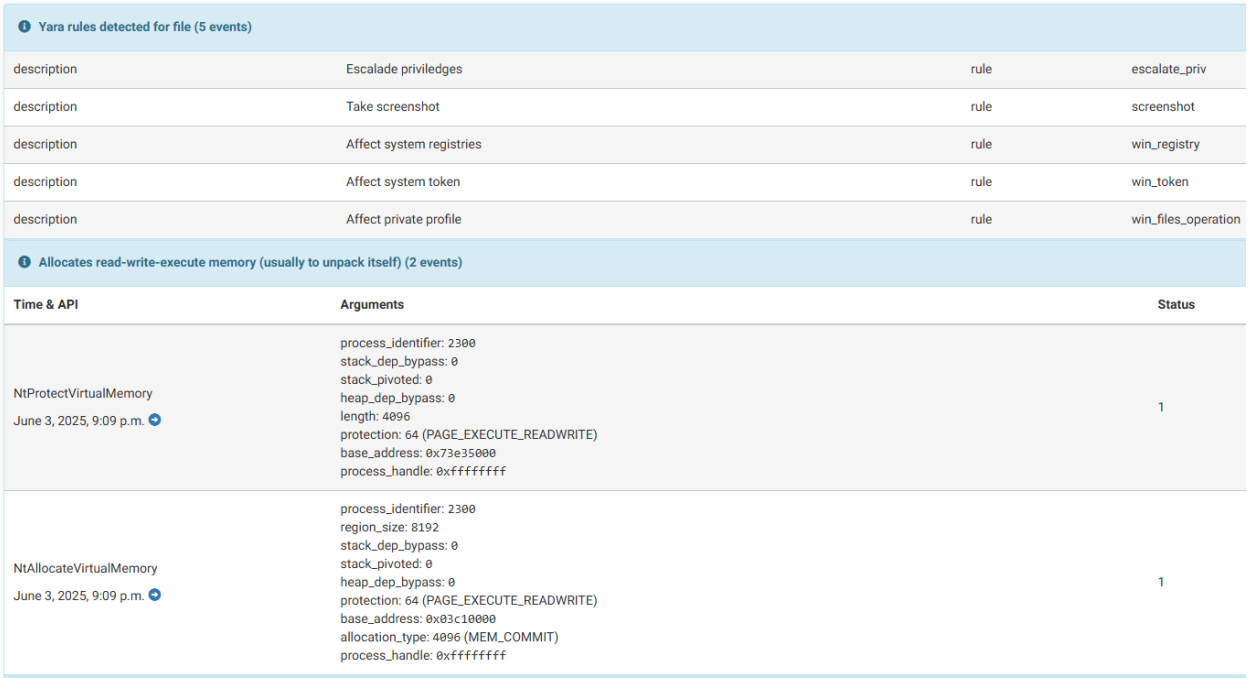
Summary

Archive *92bf5c61bcefa13a20a2a49c3d9ecc59451e665df15446f55109e94440fd7cf7.exe @ 92bf5c61bcefa13a20a2a49c3d9ecc59451e665df15446f55109e94440fd7cf7.zip*

Summary — Download | Resubmit sample

Size	78.0MB
Type	PE32 executable (GUI) Intel 80386, for MS Windows, Nullsoft Installer self-extracting archive
MD5	`d72714251abca41437952fa1c69606e2`
SHA1	`3e4f57bf2aff18f572062049b8f0cc29b43398f2`
SHA256	`92bf5c61bcefa13a20a2a49c3d9ecc59451e665df15446f55109e94440fd7cf7`
SHA512	Show SHA512
CRC32	`D861DCB6`
ssdeep	None
Yara	• escalate_priv - Escalade priviledges • screenshot - Take screenshot • win_registry - Affect system registries • win_token - Affect system token • win_files_operation - Affect private profile

Рисунок 3.9 – інформація про хеші



Yara rules detected for file (5 events)

description	Escalade priviledges	rule	escalate_priv
description	Take screenshot	rule	screenshot
description	Affect system registries	rule	win_registry
description	Affect system token	rule	win_token
description	Affect private profile	rule	win_files_operation

Allocates read-write-execute memory (usually to unpack itself) (2 events)

Time & API	Arguments	Status
NtProtectVirtualMemory June 3, 2025, 9:09 p.m.	process_identifier: 2300 stack_dep_bypass: 0 stack_pivoted: 0 heap_dep_bypass: 0 length: 4096 protection: 64 (PAGE_EXECUTE_READWRITE) base_address: 0x73e35000 process_handle: 0xffffffff	1
NtAllocateVirtualMemory June 3, 2025, 9:09 p.m.	process_identifier: 2300 region_size: 8192 stack_dep_bypass: 0 stack_pivoted: 0 heap_dep_bypass: 0 protection: 64 (PAGE_EXECUTE_READWRITE) base_address: 0x03c10000 allocation_type: 4096 (MEM_COMMIT) process_handle: 0xffffffff	1

Рисунок 3.10 – інформація про YARA-Rules, що спрацювали та інше

Протягом аналізу було зафіксовано наступні активності:

- Застосування YARA-Rules до шкідливого файлу;
- Логування результатів у відповідний каталог;
- Генерація звіту у форматі JSON.

Даний JSON звіт можна переглянути завдяки Kibana. Попередньо вони були конвертовані у NDJSON формат за допомогою Python скрипту, який згадувався раніше. JSON звіт у Elasticsearch має наступний вигляд:

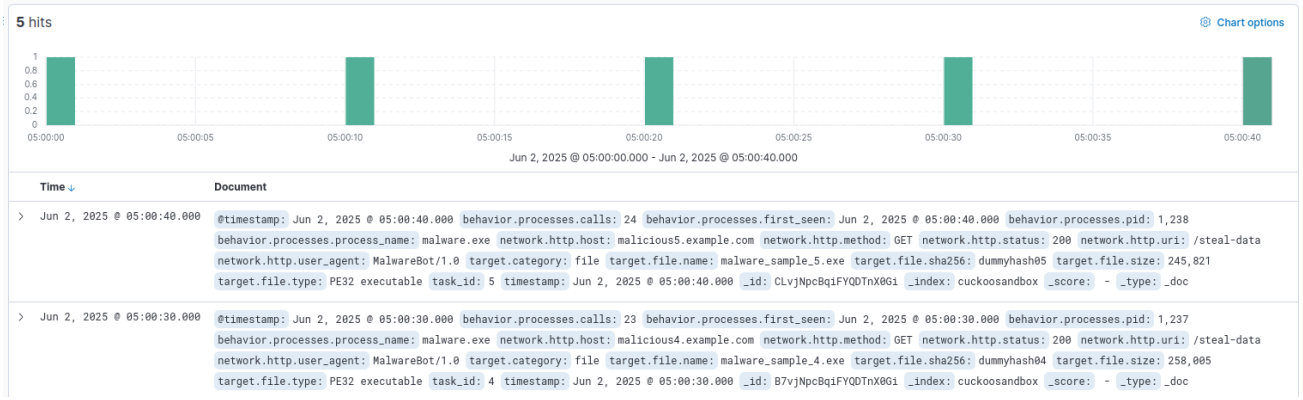


Рисунок 3.11 – JSON звіт в системі Elasticsearch

Дане тестування продемонструвало базовий функціонал нашої системи та підтвердило коректність налаштувань YARA-Rules, Elasticsearch, Kibana, праце спроможність конвертатора JSON файлів та, що дуже важливо, спроможність даної системи існувати незалежно, не спираючись на функціонал іншої. Перейдемо до тестування №2, в якому у нас буде фігурувати інтегрована гостьова система Windows 10.

Тестування № 2. Аналіз з використанням гостьової Windows 10

Другий етап – тестування за участю інтегрованої гостьової системи Windows 10. Спочатку відкатимо систему Linux на снапшот створений до тестування №1 та створимо снапшот для гостьової системи. У конфігураційному файлі cuckoo.conf на Ubuntu змінимо значення параметру “machinery” з рисунку 3.5 на “vmware”, а параметр “platform” видалимо. Файл “vmware.conf” повинен мати наступний вигляд у нашому випадку:

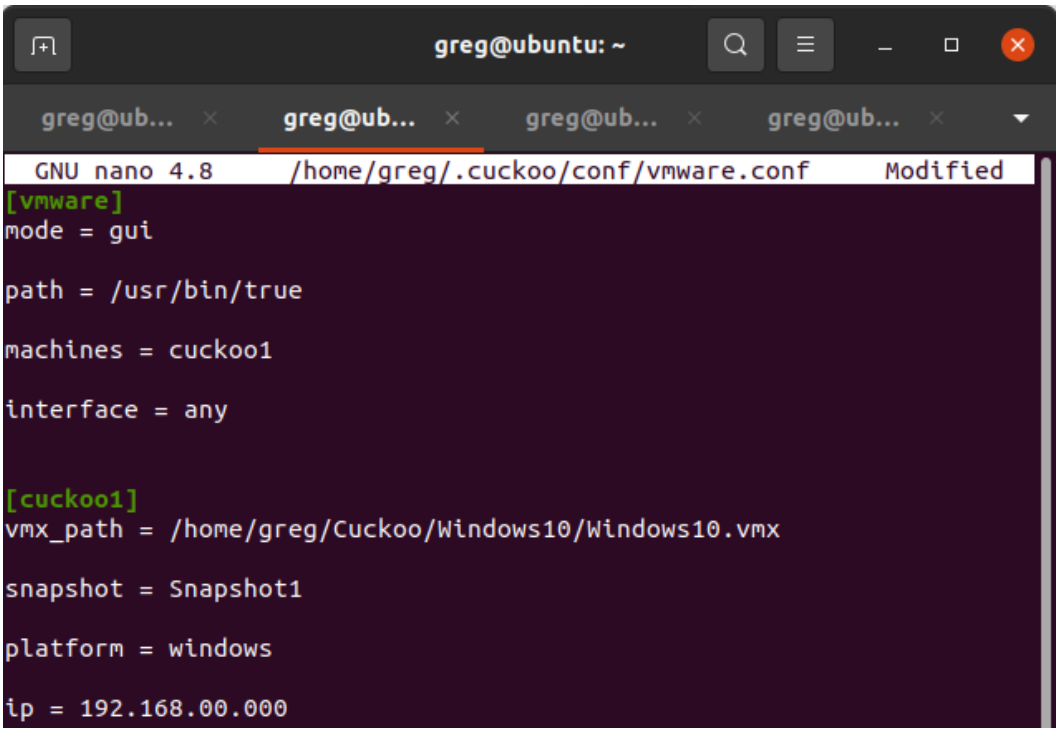


Рисунок 3.12 – параметри vmware.conf

В рамках тестування № 2 було використано ідентичний шкідливий файл, проте на цей раз відбувався вже його запуск на гостьовій машині. В ході нашого другого тестування в нагоді вже став Sysmon – за допомогою нього можна було аналізувати та відстежувати процеси, що відбувались в ОС Windows. Система Windows 10 попрацювали деякий час, що дало малвару можливість проявити. Всі процеси, які відбувались в цей час фіксувались Sysmon'ом. Деякі з процесів, що вдалось зафіксувати, наведено на рисунку 3.13 нижче.

Також результати автоматично записуються у базу даних MongoDB, яку ми підключали раніше. Аби переглянути існуючі записи треба перейти до бази Cuckoo та передивитись вміст колекції "analysis", наприклад (рис. 3.14). Це додатково дає Kibana більше можливостей для візуалізації, так як з'являється ширший набір полів, які в подальшому дозволяють візуалізувати такі аспекти як граф виклику процесів, файлову активність тощо. В ході даного тестування також відбувається генерація JSON звітів, які в подальшому також можна візуалізовувати.

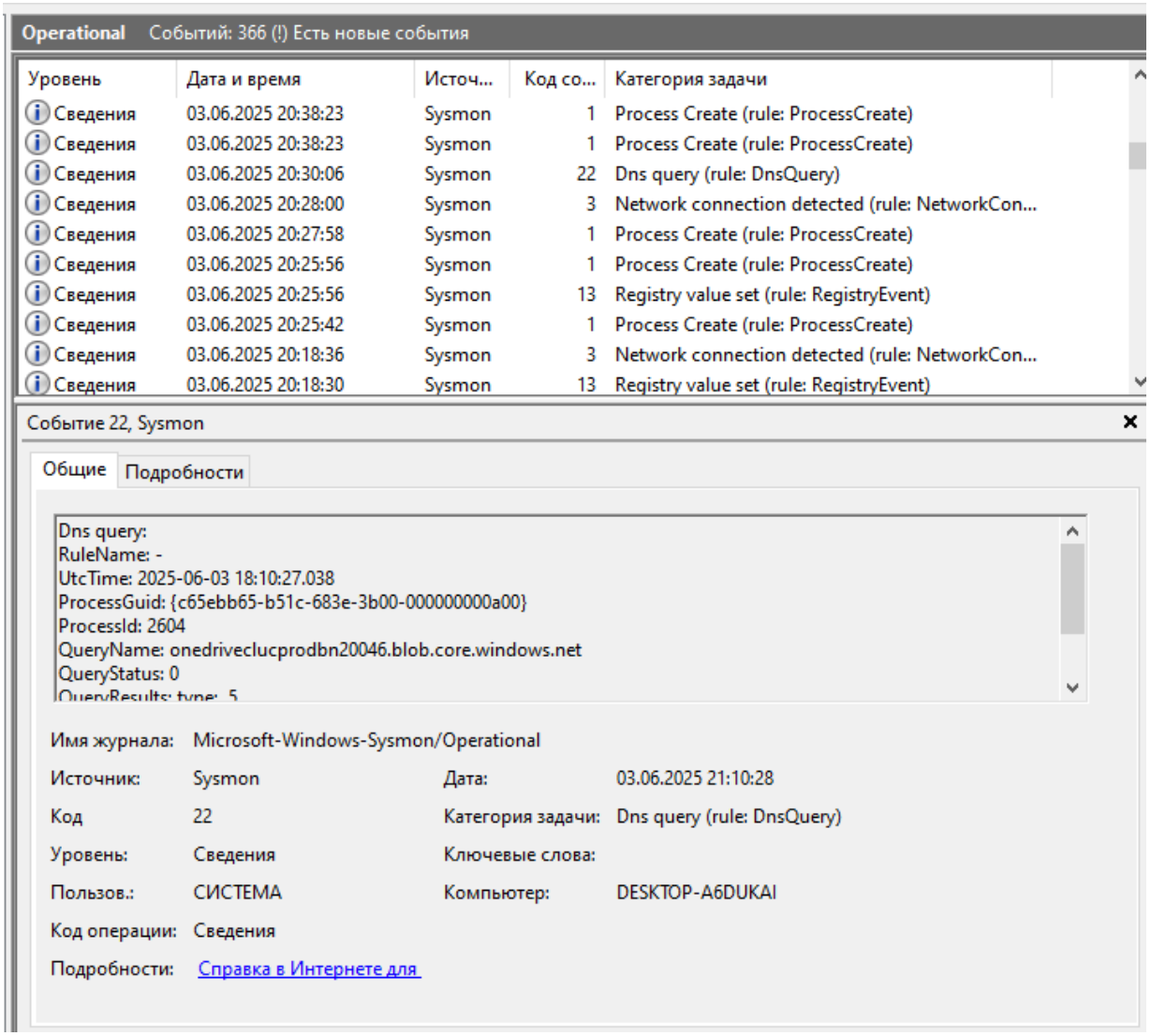


Рисунок 3.13 – зафіксовані події за допомогою Sysmon

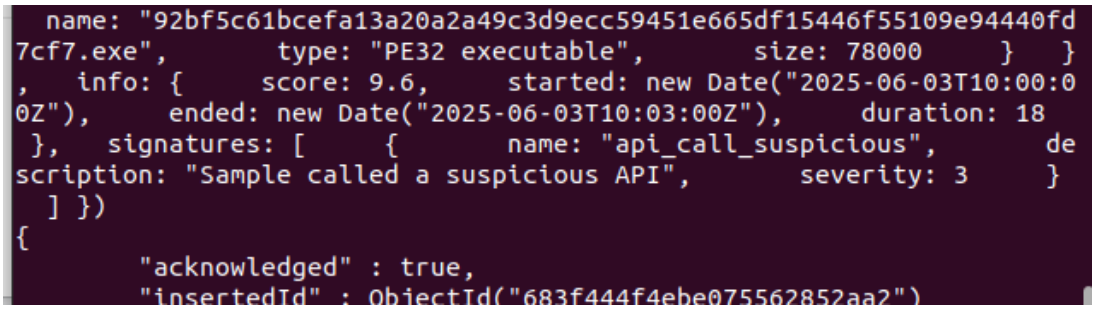


Рисунок 3.14 – невелика частина вмісту колекції “analysis”

Таким чином проведене тестування дозволили продемонструвати повноцінну динамічну аналітику з інтегрованою Windows-машиною. Загалом дані два тестування дуже чітко показують відмінність у рівнях деталізації та

кількістю доступної інформації: якщо ж у першому тесті, з використанням лише Linux середовища, доступний був лише статичний та обмежений аналіз, то у випадку з Windows – повноцінний поведінковий моніторинг. Обидва варіанти мають право на існування, потрібно лише обирати той, що більше підходить для певних умов та потреб.

3.4 Порівняння результатів до та після покращення

Фінальний етап – проведення порівняльної оцінки ефективності роботи аналізатора в двох станах: до внесення покращень (базова конфігурація «з коробки») та після впровадження покращень, описаних раніше. Підемо по порядку.

1. Базова конфігурація

У початковому варіанті Cuckoo Sandbox мав мінімальну конфігурацію, що включала лише базові компоненти:

- Відсутність інтеграції з Sysmon;
- Неактивовані YARA-Rules;
- Стандартне логування без розширеного збирання подій;
- Дуже обмежений формат звітності;
- Відсутність візуалізації через Kibana;
- Використання лише гостьового Ubuntu-аналізатора без підтримки Windows зразків.

У результаті запуску тестового аналізу простого шкідливого ПЗ, який ініціював shell-скрипт, було отримано мінімальний набір інформації, а саме:

- Створення процесу;
- Стандартна мережева активність без розгорнутої деталізації;
- Загальна інформація про файл (хеш, розмір, назва).

Розширене трасування системних подій було відсутнє, що ускладнювало можливість ідентифікації складніших загроз, зокрема шкідників із затриманим запуском.

2. Удосконалене середовище (після інтеграції модулів)

Після розширення функціональності в систему було інтегровано наступне:

- Sysmon для глибшого збирання системних подій;
- YARA для сигнатурного виявлення малвару;
- Elasticsearch та Kibana для візуалізації та інтерактивного аналізу результатів;
- Можливість роботи з NDJSON файлами для імпорту та аналізу даних;

У межах розширеної системи було проведено два тестування.

Тест № 1: Аналіз у Linux системі

Було запущено зразок шкідливого ПЗ, який створював файли, вносив зміни в реєстри тощо. В результаті відбулось наступне:

- Автоматично застосувались YARA-Rules, які успішно визначили патерн шкідливого ПЗ;
- JSON-формат виводу подій було конвертовано у NDJSON та підвантажено до Kibana для візуалізації результатів;
- В Kibana впроваджено можливість побудови дашбордів спираючись на різні вхідні дані;
- Можливий перегляд результатів з MongoDB через mongo shell.

Тест 2: Аналіз у Windows-гостьовій машині з інтегрованим Sysmon

Після розгортання Windows 10 та встановлення Sysmon система фіксувала значно більше даних:

- Можливість відстеження абсолютно всіх процесів, що відбуваються в системі в режимі реального часу;
- Логування навіть короткотривалих дій, які раніше зафіксувати було майже неможливо;
- Докладна інформація про всі порти, протоколи тощо, що були активні протягом запуску шкідливого ПЗ.

Крім того, збережені журнали з Sysmon можна підвантажувати в Elasticsearch, що в свою чергу дає виявити приховані зв’язки, зміни шаблонів та інші дії, якщо такі є.

Порівняльна таблиця результатів для більш наочного представлення:

Таблиця 3.4

Порівняння критеріїв системи «до» та «після»

Критерій	Базова конфігурація	Покращена конфігурація
Гостьова ОС	Ubuntu	Ubuntu + Windows
Інтерфейс	CLI	CLI + Web + Kibana
Аналіз YARA	Вимкнений	Активний + кастомізований
Sysmon	Немає	Наявний у Windows
Вивід у Elasticsearch	Відсутній	Активний
Глибина аналізу	Поверхневий	Поведінковий + сигнатурний
Можливість візуалізації	Відсутня	Присутня
Джерела подій	Логи Cuckoo	Cuckoo + Sysmon + MongoDB
Гнучкість	Обмежена	Є можливість розширення

Таким чином, можна сказати наступне: удосконалення середовища Cuckoo Sandbox призвело до значного розширення типів виявленої активності, покращення точності поведінкового аналізу, наявна можливість глибшого дослідження зразків у різних ОС та загалом значно підвищена зручність

користування. Якщо в початковому варіанті результати обмежувались загальними хешами, то після доопрацювання системи та впровадження зазначених змін вона надає повноцінну інфраструктуру на основі поведінкового аналізу.

Висновки за розділом 3

У третьому розділі було детально описано процес впровадження, налаштування та поступового вдосконалення середовища для динамічного аналізу шкідливого ПЗ на основі Cuckoo Sandbox. Покращення велось з точки зору практичного використання системи для аналізу як простих, так і складних зразків малвару у реальному/максимально наближеному до реального часу.

У підрозділі «Архітектура запропонованого рішення» було визначено загальну структуру майбутнього рішення. Обґрунтовано вибір основного інструментарію – Cuckoo Sandbox як основи, спираючись на його відкритість та можливість розширення. Описано компоненти, які складають основу аналітичної інфраструктури: аналізатор, базу даних, механізми моніторингу, гостьова ОС та модулі збирання/обробки результатів.

У підрозділі «Покращення модулів» було реалізовано низку технічних інтеграцій для розширення базових функціональностей. Одним з ключових моментів було впровадження YARA, що дозволило додати сигнатурний аналіз як спосіб виявлення загроз. Також розгорнуто Sysmon у гостьовій ОС для деталь-

ного моніторингу поведінкових подій на рівні ОС, що, в свою чергу, дає можливість в подальшому відслідковувати зловмисну активність. Ще одним доволі значним кроком стала інтеграція Elasticsearch та Kibana. Це аналітичний стек, що надає можливість візуалізації результатів у вигляді графіків, наприклад. Додатково було налаштовано конфігураційні файли відповідно до особливостей нашої системи.

У ході підрозділу «Тестування та оцінка роботи аналізатора» було проведено комплексне тестування у двох варіантах. Перше – з використанням

лише Linux як гостьової системи. У другому ж варіанті гостьовою системою був Windows. В обох випадках було обрано однакові тестові зразки з Malwarebazaar.

Під час обох тестів вдалось виявити та оцінити шкідливу активність: зафіксувати системні виклики, зміни параметрів, мережеву поведінку тощо. YARA-Rules також позитивно себе проявили. Загалом кожен аспект, впроваджений в нашу систему, відпрацював так як і було задумано, що продемонструвало праце спроможність нашої системи як такої.

У останньому, четвертому, підрозділі, що має назву «Порівняння результатів до та після» було проведено розгорнуту оцінку ефективності аналізатора. Якщо в початковій версії можливості системи обмежувались стандартним JSON-логуванням, базовою обробкою файлу-шкідника та мінімальним виводом інформації про даний файл, то після наших вдосконалень система перетворилась на нову. Було впроваджено інструменти моніторингу, сигнатурного аналізу, повноцінна візуалізація – всі ці моменти значно розшири спектр можливостей тої системи, яка була на початку.

Окремо можна виділити і виклики, які раніше не згадувались – необхідність налаштування доступу до віртуальних машин, встановлення специфічних залежностей під Python 2.7, оновлення конфігураційних файлів вручну, робота з мережею, інтеграція Elasticsearch вручну без додаткових плагінів тощо.

Загалом, у межах третього розділу, було створено повністю функціональну та легко масштабовану систему динамічного аналізу шкідливого ПЗ, яка потен-
ційно може бути використана як основа для подальших експериментів, досліджень та впроваджень у сфері кібербезпеки.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було здійснено повномасштабне дослідження, спрямоване на вивчення, покращення та практичну реалізацію системи динамічного аналізу шкідливого ПЗ на основі sandbox-технології. Робота охопила повний цикл – від теоретичного аналізу предметної області до побудови й вдосконалення функціонального інструментарію з подальшим тестуванням і порівнянням результатів.

У ході першого розділу було проведено глибокий аналіз класифікації шкідливого ПЗ, сучасних методів його виявлення (сигнатурних, поведінкових, евристичних), а також принципів роботи динамічного аналізу в ізольованому середовищі. Особливу увагу було приділено наявним sandbox-рішенням: як відкритим, так і комерційним. Це дало змогу сформувати цілісне уявлення про можливості, обмеження та перспективи використання подібних систем для потреб кіберзахисту. Було встановлено, що хоча традиційні методи виявлення, як, наприклад, сигнатурний, досі є актуальним, проте поведінковий аналіз набуває вирішального значення в умовах нових типів загроз.

Другий розділ був присвячений критичному аналізу сучасних sandbox-систем та виявленню їх обмежень. Визначено ключові проблеми, які починались недостатньою симуляцією дій користувача, а закінчувались відсутністю зручних візуалізаційних інструментів для результатів аналізу. На основі цього було обґрунтовано доцільність вдосконалення системи Cuckoo Sandbox – одного з найбільш поширених рішень з відкритим кодом. Основними напрямками покращення в даному випадку було обрано інтеграцію Sysmon для збору низькорівневої поведінкової інформації, підключення сигнатурного аналізу через YARA, а також візуалізацію зібраних результатів через Elasticsearch та Kibana. Окрім цього, було враховано потребу у підвищенні гнучкості системи, її масштабованості та легкості подальшого розширення.

У ході третього розділу відбувалась реалізація всіх запланованих впровад-

жень. Було детально описано архітектуру оновленої системи, виконано інсталяцію та конфігурацію всіх необхідних компонентів включно з YARA, Sysmon, PostgreSQL, MongoDB, Elasticsearch та Kibana. Було налаштовано конфігураційні файли системи, розгорнуто окремі середовища, протестовано роботу аналізатора на двох платформах щ використанням однакових зразків. Це дало змогу виявити рівні деталізації та повноти зібраної інформації, зокрема значний приріст цінних артефактів при роботі в середовищі Windows.

Порівняльний аналіз до і після впроваджень покращень наочно продемонстрував ефективність нової системи: було усунуто більшість раніше виявлених недоліків, значно розширено обсяг поведінки, що логується, покращено точність виявлення загроз та покращено інтерпретацію результатів шляхом віртуалізації. Додатково, реалізовано завантаження даних у Kibana, що відкриває можливість для інтерактивного аналізу з використанням сучасних інструментів бізнес-аналітики.

Загалом результати кваліфікаційної роботи доводять, що шляхом інтеграції відкритих та ефективних інструментів можливо створити повноцінне середовище для аналізу шкідливого ПЗ, яке може бути використано як в освітньому процесі, так і в реальних умовах дослідницької або корпоративної діяльності. Запропоноване рішення є не лише практично релевантним, але й гнучким у розширені – надалі його можна адаптувати для виявлення більш складних загроз, інтегрувати з SIEM-системами або автоматизувати процес обробки зразків за допомогою REST API.

Таким чином, поставлені у роботі задачі повністю реалізовані, а мета досягнута. Результати дослідження можуть бути основою для подальшого розвитку напрямів автоматизованого аналізу загроз у сфері інформаційної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is malware? [Електронний ресурс]. – Режим доступу: https://www.malwarebytes.com/malware

2. What is malware? Prevention, detection and how attacks work [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/searchsecurity/definition/malware

3. Malware analysis techniques [Електронний ресурс]. – Режим доступу: https://www.e-spincorp.com/malware-analysis-techniques/

4. Juliet Odii, JohnPaul Hampo Comparative Analysis of Malware Detection Techniques Using Signature, Behavior and Heuristics [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/350017172_COMPARATIVE_ANALYSIS_OF_MALWARE_DETECTION_TECHNIQUES_USING_SIGNATURE_BEHAVIOUR_AND_HEURISTICS

5. Cuckoo Sandbox Overview [Електронний ресурс]. – Режим доступу: https://www.varonis.com/blog/cuckoo-sandbox#basic-structure-of-cuckoo

6. Joe Sandbox Ultimate Overview [Електронний ресурс]. – Режим доступу: https://www.joesecurity.org/joe-sandbox-ultimate#overview

7. Statistics Malwarebazaar [Електронний ресурс]. – Режим доступу: https://bazaar.abuse.ch/statistics/

8. Sandbox Review [Електронний ресурс]. – Режим доступу: https://old.roi4cio.com/categories/sandbox/

9. What is Sandboxing? The Types, Benefits & Challenges [Електронний ресурс]. – Режим доступу: https://votiro.com/blog/what-is-sandboxing/

10. Sysmon v15.15 [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

11. SwiftOnSecurity on GitHub [Електронний ресурс]. – Режим доступу: https://github.com/SwiftOnSecurity/sysmon-config

12. YARA-Rules on GitHub [Електронний ресурс]. – Режим доступу:

https://github.com/Yara-Rules/rules

13. Elasticsearch on GitHub [Електронний ресурс]. – Режим доступу: https://github.com/elastic/elasticsearch

14. Kibana on GitHub [Електронний ресурс]. – Режим доступу: https://github.com/elastic/kibana

15. MongoDB Tutorial [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/mongodb/

16. Automate anything with Cuckoo Sandbox [Електронний ресурс]. – Режим доступу: https://mindflow.framer.website/integrations/cuckoo-sandbox

17. Log Analysis with Sysmon Walkthrough [Електронний ресурс]. – Режим доступу: https://stumblesec.medium.com/letsdefend-log-analysis-with-sysmon-walkthrough-775bb1dd2aa1

18. Virusshare [Електронний ресурс]. – Режим доступу: https://virusshare.com/