Міністерство освіти і науки України
Київський Національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА
Дипломної роботи

магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній рівень ***магістр***
(назва освітнього рівня)

освітньо-наукова програма *кібербезпека*
(код і назва кваліфікації)

На тему: ***Методи виявлення аномалій інформаційної безпеки на кінцевих системах***

Виконавець: студент _____ **курсу, групи** ***КБМ-21***

_____ ***Ландовський Юрій Володимирович***
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	Оцінка	Підпис
Науковий керівник	**Бабенко Т.В.**		
Рецензент	**Барановський О.М.**		
Нормоконтроль			

Київ
2022

Міністерство освіти і науки України
Київський Національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри
кібербезпеки та захисту інформації

_____ Н.В. Лукова-Чуйко

«_____» _____20__ року

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

студенту ***КБМ-21*** **Ландовський Юрій Володимирович**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Методи виявлення аномалій інформаційної безпеки на кінцевих системах

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол №5 від 29.10.2021.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень процес несанкціонованого доступу до ресурсів кінцевих систем.

Предмет досліджень несанкціонований криптомайнинг

Мета виявлення аномалії ІБ

Вихідні дані для проведення роботи методи виявлення криптомайнерів на кінцевих системах

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна розроблений метод виявлення криптомайнингу, який поєднує інформацію про виконані операції та мережеву активність додатку

Практична цінність запропоновний метод може бути використаним як додатковий механізм в засобах захисту на рівні програмних застосунків

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Уточнення постановки задачі	29.10.2021 – 30.10.2021
Аналіз літератури	01.11.2021 – 01.01.2022
Дослідження поведінки криптомайнерів	02.01.2022 – 01.03.2022
Створення методу виявлення криптомайнерів	02.03.2022 – 01.04.2022
Програмна реалізація запропонованого методу	02.04.2022 – 01.05.2022
Оформлення пояснювальної записки	10.05.2022 – 15.05.2022
Підготовка до захисту дипломної роботи	16.05.2022 – 19.05.2022

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект зниження збитків несанкціонованого використання ресурсів кінцевих систем

Соціальний ефект покращення технологій виявлення несанкціонованих операцій як особисто так і на підприємствах.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ _____
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ _____
(підпис) (прізвище, ініціали)

Дата видачі завдання: _____

Термін подання дипломної роботи до ЕК _____

РЕФЕРАТ

Пояснювальна записка: 80 с., 39 рисунків, 1 формула, 98 джерел.

Об’єкт дослідження: процес несанкціонованого доступу до ресурсів кінцевих систем.

Мета роботи: виявлення аномалій ІБ.

Методи дослідження: системний підхід, методи аналізу, спостереження, опису, експерименту, імітаційне моделювання.

У роботі досліджена поведінка криптомайнерів. Проведено аналіз актуальних рішень для виявлення криптомайнингу. Проаналізовані процеси, які виникають в додатках пов’язані з криптомайнингом, а саме аномалії в мережевому трафіку та операціях. На основі знайдених результатів, пропонується метод виявлення криптомайнерів, який відслідковує, які операції виконуються додатком під час його роботи. Для покращення точності виконання додатків пропонується поєднання аспектів мережевого трафіку криптомайнерів та запропонованого методу.

Наукова новизна дослідження полягає у тому, що вперше було розроблено метод виявлення криптомайнингу, який поєднує інформацію про виконані операції та мережеву активність додатку.

Напрямки подальших досліджень: розробка інтелектуальних моделей класифікації подій криптомайнингу; виявлення криптомайнингу на різних архітектурах (ARM), покращення роботи алгоритму виявлення аномалій криптомайнингу.

Ключові слова: аномалії ІБ, криптомайнинг, криптоджекінг, шкідливе програмне забезпечення, виявлення.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПЗ – Програмне забезпечення

ІБ – Інформаційна безпека

ІС – Інформаційна система

ВСТУП

Актуальність. З того дня, коли біткойн був випущений в 2009 році, криптовалюти на основі блокчейну викликали зростання інтересу за межами певних спільнот, таких як банківські та комерційні організації. Особливо у 2017 році інтерес до криптовалют досяг піку із загальною ринковою вартістю близько 1 трильйона доларів [98].

Покупка криптовалюти – не єдиний спосіб інвестування. Інвестори також можуть створювати пули для майнінгу, щоб генерувати нові монети, щоб отримати прибуток.

Такий ажіотаж навколо криптовалют створив цілий ринок для злочинців, які встановлюють у своїх ботнетах криптомайнери. Це дає їм змогу майнити криптовалюти і заробляти гроші. Також зловмисники отримують доступ до нешкідливих сайтів, встановлюють скрипти і тепер кожного разу, як завантажується сторінка, починається процес несанкціонованого майнингу.

Cryptojacking — це акт використання обчислювальної потужності жертви без згоди на майнинг криптовалюти. Ця несанкціонована операція майнінгу вимагає додаткового споживання електроенергії та різко знижує обчислювальну ефективність хоста-жертви. В результаті зловмисник перетворює цю несанкціоновану обчислювальну потужність у криптовалюту. Особливо після появи постачальників послуг, які пропонують готові до використання реалізації сценаріїв майнінгу в браузері, зловмисники можуть легко досягти великої кількості користувачів через популярні веб-сайти.

Тому існує потреба в розробленні методів захисту від несанкціонованого криптомайнингу.

Метою дипломної роботи є виявлення аномалій ІБ. Головна увага приділяється аномальним подіями пов’язаним з діяльністю криптомайнерів.

Для досягнення зазначеної мети дипломної роботи поставлені окремі завдання:

- дослідження існуючих рішень для виявлення криптомайнерів

- аналіз аномальних процесів пов’язаних з діяльністю криптомайнерів
- розробка методу виявлення криптомайнерів
- створення програмного модуля методу виявлення криптомайнерів
- синтез моделі виявлення аномальної поведінки програмного забезпечення
- аналіз мережевого трафіку криптомайнерів
- покращення методу виявлення аномальної поведінки поєднуючи інформацію про мережевий трафік та синтезовану модель.

Об’єкт дослідження – процес несанкціонованого доступу до ресурсів кінцевих систем.

Предмет дослідження – несанкціонований криптомайнинг.

При вирішенні поставлених завдань у дипломній роботі були використані: системний підхід, методи аналізу, спостереження, опису, експерименту, імітаційне моделювання.

Наукова новизна дослідження полягає у тому, що вперше було розроблено метод виявлення криптомайнингу, який поєднує інформацію про виконані операції та мережеву активність додатку.

Практична цінність роботи полягає в тому, що розроблений метод бути використаний як додатковий механізм в засобах захисту на рівні програмних застосунків;

Основні наукові положення і результати роботи доповідалися та обговорювалися на Міжнародній науково-практичній конференції «Проблеми кібербезпеки інформаційно-телекомунікаційних систем» 2022 (PCSITS) та міжнародній конференції «38th IBIMA Conference»

РОЗДІЛ 1. КРИПТОМАЙНИНГ ЯК РІЗНОВИД ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Поняття блокчейну

Блокчейн — це технологія розподіленого цифрового реєстру, що зберігає в мережі однорангові транзакції (P2P) незмінним чином. Структура блокчейна складається з ланцюжка блоків. Наприклад, у біткойнах [1] кожен блок має дві частини: заголовок і транзакції. Заголовок блоку складається з хешу попереднього блоку, версії, відмітки часу, цільової складності, nonce (число, яке криптомайнери пробують підібрати) і корінь дерева Меркла. Кожен блок математично пов’язаний з попереднім. Ця прив’язка робить неможливим змінювати дані в будь-якому блоку ланцюжка. З іншого боку, друга частина кожного блоку включає в себе набір індивідуально підтверджених транзакцій.

1.2 Основи криптомайнингу

Незмінність блокчейну забезпечується через механізм консенсусу, який зазвичай реалізується шляхом протоколу «Proof-of-Work» (PoW). Незмінність кожного блоку та всього блокчейну зберігається завдяки ланцюговій та блочній структурі. У PoW, деякі вузли в мережі вирішують хеш-головоломку, де треба знайти унікальне хеш-значення та передати його всім іншим вузлам в мережі. Перший вузол, що транслює правильне хеш-значення винагороджується нагородою за блок та збирає комісію за транзакції. Хеш-значення перевіряється відповідно до цілі складності, тобто, якщо вона задовольняє ціль складності, вона приймається всіма іншими вузлами, і вузол, який знайшов правильне хеш-значення, отримує винагороду. Різні реалізації PoW можуть мати різні методи для цілі складності.

Майнери намагаються знайти правильне хеш-значення методом проб і помилок, збільшуючи значення nonce для кожного випробування. Коли знайдене

правильне хеш-значення, весь блок транслюється до мережі, а блок додається в кінець, як останній блок. Цей процес відомий як криптомайнинг, і це єдиний спосіб створення нової криптовалюти. Шанс знайти правильний хеш прямо пропорційний хеш-потужності майнера, тобто вона пов'язана з обчислювальною потужністю самого обладнання. Однак збільшення кількості обладнання також збільшує споживання електроенергії. Через це зловмисники шукають нові шляхи збільшення обчислювальної потужності без збільшення власного споживання електроенергії. Після винаходу біткойна з'явилося багато інших альтернативних криптовалют, їх ще називають альткойнами. Ці нові криптовалюти стверджують, що вирішують деякі проблеми, пов'язані з архітектурою біткойна (наприклад, масштабованість [2], конфіденційність) або пропонують нові рішення (наприклад, розумні контракти [3]).

На початку існування біткойна, майнінг проводився за допомогою звичайного центрального процесора (ЦП), і користувачі могли легко використовувати свої звичайні ЦП для видобутку біткойнів. З часом майнери на основі графічного процесора (GPU) отримали значні переваги перед майнерами CPU, оскільки графічні процесори були спеціально розроблені для високої обчислювальної продуктивності для важких програм. Пізніше Field Programmable Gate Array (FPGA) змінили ландшафт майнінгу криптовалюти, оскільки, це було гнучким обладнанням і забезпечувало значно більший прибуток, ніж майнінг на основі CPU або GPU. Нарешті, використання специфічних інтегральних схем (ASIC) домінує в галузі майнінгу, оскільки, вони спеціально виготовлені та налаштовані для видобутку криптовалюти.

Альтернативні криптовалюти почали використовувати різні хеш-функції у своїй структурі блокчейна, що призвело до розбіжностей у процесі майнінгу. Наприклад, Monero [4] використовує алгоритм CryptoNight як хеш-функцію. CryptoNight спеціально розроблений для майнінгу використовуючи CPU та GPU. Використовуючи алгоритм RandomX [4], блокчейн Monero повністю усунув ASIC-майнери і значно збільшив ефективність майнингу на ЦП. Ця функція робить Monero єдиною великою криптовалютною платформою, яка була розроблена спеціально для поширення майнінгу на ЦП. Вона забезпечує функції

невідстежуваності та від’єднання за допомогою змішувачів та кільцевих підписів. Всі ці характеристики роблять Monero привабливим для зловмисників.

1.3 Типи зловмисного криптомайнингу. Криптоджекінг

Криптоджекінг, також відомий як шкідливе програмне забезпечення для майнінгу криптовалют, ставить під загрозу обчислювальні ресурси пристрою жертви (тобто комп’ютери, мобільні пристрої) без дозволу користувача на майнінг криптовалют та отримання за це винагород. Життєвий цикл криптоджекінгу складається з трьох основних фаз:

- підготовка скрипта;
- ін’єкція скрипта;
- атака.

Підготовка скрипта та етапи атаки однакові для всіх типів шкідливих програм криптоджекінгу. При цьому, фаза ін’єкції скрипта здійснюється або шляхом введення шкідливого скрипта на веб-сайти, або локального вбудовування шкідливого програмного забезпечення в інші програми. Виходячи з цього, класифікується зловмисне програмне забезпечення криптоджекінгу на дві категорії:

- криптоджекінгу в браузері
- криптоджекінгу на основі хосту.

1.3.1 Криптоджекінг в браузері

Розвиток веб-технологій, таких як JavaScript (JS) і WebAssembly (Wasm), створив інтерактивний веб-контент, який може отримати доступ до кількох обчислювальних ресурсів (наприклад, ЦП) пристрою жертви (наприклад, комп’ютера або мобільного пристрою). Шкідливе програмне забезпечення для криптоджекінга в браузері використовує ці веб-технології для створення

несанкціонованого доступу до системи жертви для майнінгу криптовалюти за допомогою взаємодії з веб-сторінкою на центральному процесорі жертви.

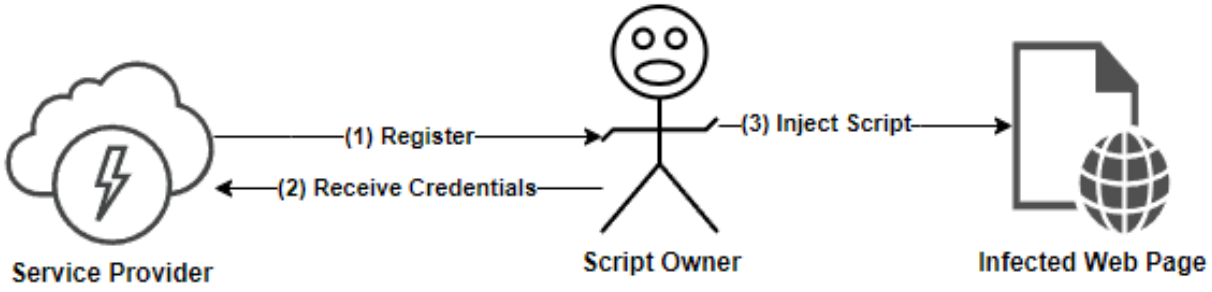


Рисунок 1.1 Етапи поширення зловмисного програмного забезпечення криптоджекінгу в браузері

На рисунок 1.1 показані етапи підготовки скрипта та впровадження зловмисного програмного забезпечення для криптоджекінгу в браузері. Власник скрипта спочатку реєструється (крок 1) і отримує облікові дані служби та готові до використання скрипти для майнінгу від постачальника послуг (крок 2). Постачальник послуг розділяє завдання майнінгу між своїми користувачами і збирає весь дохід від пулу майнінгу, щоб потім розподілити його між користувачами. Отримавши облікові дані служби, власник скрипта вводить зловмисний скрипт у вихідний код веб-сайту (крок 3).

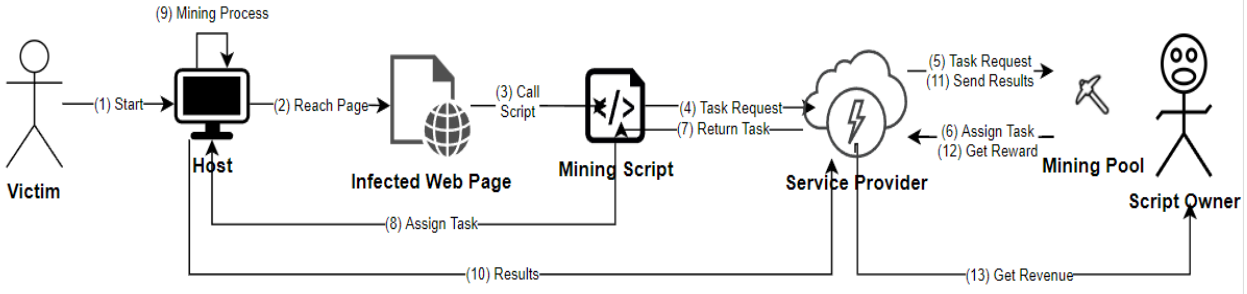


Рисунок 1.2 Повний цикл зловмисного програмного забезпечення криптоджекінгу

На фазі атаки, як показано на рисунку 1.2, жертви спочатку потрапляють до вихідного коду веб-сайту зі своїх пристроїв (Крок 1,2). Веб-браузер завантажує веб-сайт і автоматично викликає скрипт майнінгу криптоджекінгу (Крок 3). Після

виконання скрипта він запитує завдання майнінгу від постачальника послуг (Крок 4). Постачальник послуг передає запит завдання до майнінгового пулу (Крок 5). Потім пул для майнінгу призначає завдання майнінгу (Крок 6). Постачальник послуг повертає завдання в скрипт майнінгу (Крок 7). Сценарій майнінгу повертає це нове призначення майнінгу на комп'ютер жертви (Крок 8), а пристрій жертви починає процес майнінгу (Крок 9). Поки сценарій майнінгу та постачальник послуг залишаються онлайн, скрипт продовжується процес майнінгу на комп'ютері жертви (Крок 9), а потім повертає результати майнінгу безпосередньо постачальнику послуг (Крок 10). Постачальник послуг збирає всі дані з різних джерел і надсилає результати до майнінгового пулу (Крок 11). Нарешті, майнінговий пул надсилає винагороду назад постачальнику послуг у вигляді видобутої валюти (Крок 12). Власник скрипта отримує свою частку від постачальників послуг, використовуючи свої облікові дані після того, як постачальник послуг зменшує плату за послуги. У цій екосистемі зловмисники використовують потужність ЦП своїх жертв, а жертви не отримують жодної оплати чи вигоди від будь-якої іншої особи.

1.3.2 Криптоджекінг на основа хоста

Криптоджекінг на основі хоста — це безшумне зловмисне програмне забезпечення, яке зловмисники використовують для доступу до ресурсів хоста-жертви та для того, щоб зробити його комп'ютером-зомбі для власника зловмисного програмного забезпечення. Порівняно зі зловмисним програмним забезпеченням для криптоджекінгу в браузері, зловмисне програмне забезпечення на базі хостів не отримує доступ до обчислювальної потужності жертви через веб-скрипт; замість цього їх потрібно встановити на хост-системі. Тому вони, як правило, доставляються до хост-системи за допомогою таких методів, як вбудовування в програми сторонніх розробників [5][6], з використанням вразливостей [7] або методів соціальної інженерії [8], або як через техніку drive-by-download [9].

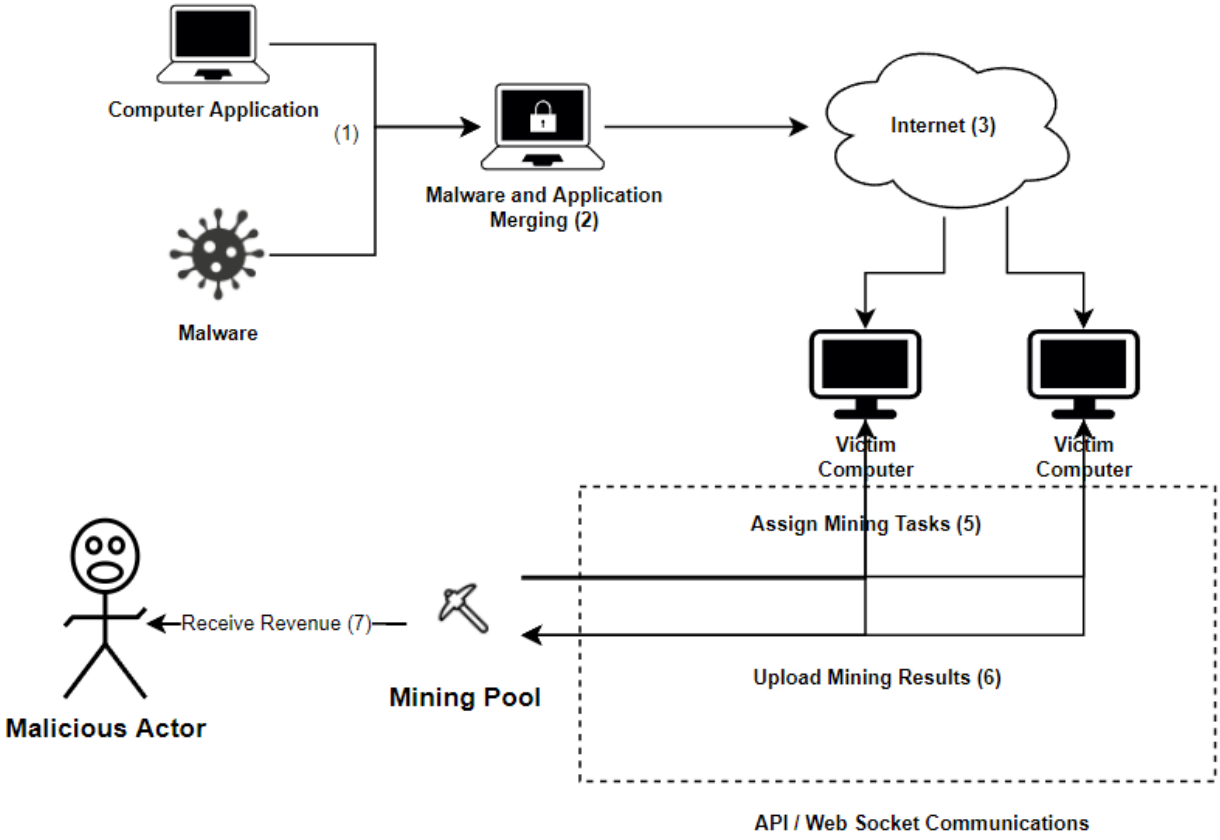


Рисунок 1.3 Життєвий цикл шкідливого програмного забезпечення криптоджекінгу на основі хоста

На рисунку 1.3 показано життєвий цикл зловмисного програмного забезпечення криптоджекінгу на основі хоста. Етап підготовки скрипта починається зі створення несанкціонованого шкідливого ПЗ для майнінгу криптовалюти. Потім зловмисник об'єднує це зловмисне програмне забезпечення з законною програмою, щоб обдурити жертву. Після підготовки зловмисного програмного забезпечення процес ін'єкції шкідливого програмного забезпечення починається із завантаження цієї шкідливої програми на онлайн-платформи обміну даними (наприклад, торренти, загальнодоступні хмари). Коли жертва завантажує будь-яку із заражених програм і встановлює їх на свої хост-комп'ютери (наприклад, персональний комп'ютер, пристрій IoT, сервер), фаза ін'єкції шкідливого програмного забезпечення завершується.

На фазі атаки зловмисне програмне забезпечення для криптоджекінгу на основі хоста підключається до пулу майнінгу через веб-сокет або API і отримує

завдання хеш-головоломки для обчислення значень хешування. Розраховані хеш-значення відправляються назад до майнінгового пулу. Нарешті, зловмисник отримує весь дохід без споживання енергії і нічим не ділиться з жертвою.

Отримавши весь свій дохід у вигляді криптовалюти від постачальника послуг, зловмисник має три варіанти використання свого доходу:

- конвертувати в фіатну валюту через біржі або p2p-транзакції;
- використовувати її як криптовалюту для своїх послуг [10];
- використання сервісів змішування для приховування слідів [11] [12].

Подальший наскрізний аналіз економіки/платежів криптоджекінгу виходить за рамки цього дослідження, і подібні дослідження можна знайти в області програм-вимагачів [13]–[16].

1.4 Методи, які використовуються в криптоджекінгу

У цьому підрозділі будуть розглянуті методи, які використовуються для зловмисного програмного забезпечення – криптоджекінгу, а саме:

- джерела криптоджекінгу;
- способи зараження;
- типи платформ жертв;
- цільові криптовалюти;
- методи виявлення і уникнення;
- техніки ухилення та обфускації.

1.4.1 Джерела криптоджекінгу

Постачальники послуг є провідними творцями та розповсюджувачами скриптів криптоджекінгу. Постачальники послуг надають кожному користувачеві унікальний ідентифікатор, щоб розрізняти його з точки зору потужності хешування. Постачальник послуг створює сценарій для користувача, незалежно від того, чи є

користувач шкідливим чи ні. Все, що користувачеві потрібно зробити, це скопіювати та вставити скрипт, щоб створити зразок шкідливого для атаки.

Coinhive [17] був першим постачальником послуг, який запропонував у 2017 році готовий до використання скрипт для майнінгу в браузері, щоб створити альтернативний дохід для власників веб-сайтів і вмісту. Незважаючи на те, що початкова ідея Coinhive полягала в тому, щоб надати власникам веб-сторінок альтернативний дохід, вона швидко стала популярною серед зловмисників. Під час роботи Coinhive вони володіли значною часткою загального хешрейту Monero. Після різкого зниження ціни Monero [18], Coinhive було закрито власниками в березні 2019 року через те, що бізнес більше не був прибутковим.

Деякі з альтернативних постачальників послуг, які продовжили/продовжили свою діяльність: Authedmine, Browsermine, Coinhave, Coinimp, Coinnebula, Cryptoloot, DeepMiner, JSECoin, Monerise, Nerohut, Webmine, WebminerPool і Webminepool. Деякі з цих постачальників послуг також придумали кілька нових функцій, наприклад, пропонуючи метод сповіщення користувача або графічний інтерфейс користувача, щоб налаштувати параметри криптомайнінгу.

Мережі Blockchain покладаються на кілька мережевих протоколів і методів криптографічної аутентифікації. Майнери повинні бути частиною цих протоколів і дотримуватися правил, наданих і розроблених спільнотами. Криптовалюти на основі PoW також мають особливі правила для своїх блокчейн-мереж. Через публічний та відкритий характер технології блокчейн вихідний код цих майнерів публікується спільнотами через платформи спільного використання коду та спільної розробки. Зловмисники можуть легко отримати та модифікувати ці майнери та використовувати їх для майнінгу на хост-машинах своїх жертв. Крім того, є також кілька програм для майнінгу в стилі plug-and-play, які надаються кількома пулами для майнінгу. Зловмисники також модифікують ці програми для криптоджекінгу. Наприклад, XMRig є законною високопродуктивною реалізацією майнера Monero і є відкритим вихідним кодом.

1.4.2 Способи зараження

Власники веб-сайтів, які мають доступ адміністратора до серверів веб-сайту, можуть використовувати скрипти для майнінгу в браузері, щоб отримати додатковий дохід або надати в обмін альтернативу наданому ними преміум-контенту. Тільки за допомогою цього методу власники веб-сторінок можуть отримувати дохід від скрипта на своїй веб-сторінці. Хоча деякі власники веб-сайтів повідомляють своїх відвідувачів про використовуваний ними скрипт криптомайнінгу, інші не повідомляють своїх відвідувачів. Така поведінка може вважатися злочином [19] у кількох країнах.

Зловмисники можуть впроваджувати зловмисне програмне забезпечення у довільні веб-сторінки, які мають кілька вразливостей. Спеціалісти зробили дослідження і стверджують, що всього десять користувачів створили 85% усіх знайдених скриптів Coinhive [20]. Власники самих веб-сторінок не мають жодної інформації про ці скрипти; крім того, вони не отримують від них прибутку.

У певних роботах стверджується, що зловмисники зазвичай використовують один і той же ідентифікатор для всіх заражених веб-сторінок, що дозволяє легше їх виявити. Наприклад, автори в [21] розкривають кампанії криптоджекінгу за допомогою цього методу і виявляють, що більшість із цих кампаній використовують такі вразливості, як уразливості віддаленого виконання коду. Під час дослідження було виявлене зловмисне програмне забезпечення на веб-сторінках уряду Індії [22], яке впливає на всі домени та субдомени ap.gov.in.

Деякі зловмисники вбудовують зловмисне програмне забезпечення для криптомайнингу в рекламу на основі JavaScript і поширюють їх за допомогою скриптів для майнінгу. За допомогою цього методу зловмисники можуть поширюватися без додаткових зусиль. Щоб здійснити цю атаку, зловмисниками не потрібно заражати якусь веб-сторінку чи програму. Навіть сервіси YouTube [23] та Google Ads [24] були уражені, а користувачі цих веб-сайтів та їхніх сервісів стали жертвами атак криптоджекінга. Зловмисники разом зі своїми відвідувачами успішно

майнили криптовалюту Monero. Перевага даного методу полягає в тому, що він дозволяє зловмисникам охопити велику кількість відвідувачів, коли рекламу вбудовують у популярні веб-сайти, і таким чином нема потреби в доступу до серверів веб-сайту.

Розширення для браузера також можуть отримати доступ до ресурсів ЦП комп'ютера і діяти як зловмисне програмне забезпечення, розташоване на веб-сторінці. Ці розширення мають основну відмінність - вони можуть працювати та займатися майнінгом, доки заражений браузер залишається відкритим незалежно від веб-сайтів, до яких звертається жертва. Однак великі оператори браузерів, такі як Google, оголосили, що заборонять усі розширення криптомайнінгу на своїй платформі незалежно від їх наміру, оскільки на практиці ними зловживають [25].

Об'єднання зловмисного програмного забезпечення з будь-яким ринковим додатком та його публікація на кількох платформах для спільного використання є добре відомим серед зловмисників методом поширення шкідливого програмного забезпечення. Зловмисники модифікують програмне забезпечення криптомайнера, щоб виконувати криптоджекінг у фоновому режимі та об'єднувати його з легальними програмами. Зловмисники, як правило, використовують додатки з інтенсивними обчисленнями (наприклад, анімаційні програми, ігри з високими потребами в апаратному забезпеченні, інженерні програми), оскільки використання цих програм означає, що система жертв має потужне обчислювальне обладнання та програму, в яку вбудовано зловмисне програмне забезпечення для криптоджекингу і яка має дозвіл на доступ до необхідних апаратних компонентів хост-системи жертви.

Вже траплялися кілька серйозних випадків, наприклад, один зловмисник об'єднав програму для відеодзвінків Zoom зі звичайним майнером біткойнів і поширив її через кілька платформ обміну [5]. Під час іншої атаки зловмисники використали популярну відеогру Fortnite для поширення вірусу для видобутку біткойнів [27]. На відміну від майнінгу в браузері, який став популярним у 2017 році, були знайдені екземпляри атаки за допомогою цього методу навіть у 2013 році, коли скрипт майнінгу біткойна знаходився як частина коду самої гри [28].

У певних випадках зловмисники використовують декілька вразливостей нульового дня, які вони знайшли в апаратному та програмному забезпеченні. Зловмисники вводять своє шкідливе програмне забезпечення для майнінгу в кілька пристроїв і змушують їх добувати криптовалюту. Найяскравіший приклад безпосередньо вплинув на 1,4 мільйона маршрутизаторів Mikrotik [29], де була знайдена вразливість на апаратному рівні. Дослідники стверджують, що основний відсоток атак віддаленого запуску коду [30] спрямований на те, щоб знайти сценарії майнінгу всередині хост-систем.

Соціальна інженерія є широко використовуваною технікою зловмисників зловмисного програмного забезпечення для обходу методів безпеки. Подібним чином зловмисники також використовують атаки соціальної інженерії, щоб маніпулювати людською психологією та перевіряти доступ жертв або встановлювати шкідливе програмне забезпечення на їхні комп'ютери. Дослідники помітили, що зловмисники все ще використовують старі методи, такі як соціальна інженерія, для встановлення зловмисного програмного забезпечення для криптоджекінгу на комп'ютерах своїх жертв [31]. Більш того, таким методом можна ефективно поширювати програмне забезпечення, оскільки, можна просто зробити розсилку шкідливого програмного забезпечення.

Drive-by download — це ще один прийом, який використовується зловмисниками для доставки та встановлення шкідливих файлів на пристрої жертв без їхнього відома. Жертви можуть зіткнутися з цією атакою під час відвідування веб-сторінки, відкриття спливаючого вікна або перевірки вкладення електронної пошти. В одному випадку зловмисники скористалися цим методом, щоб ввести зловмисне програмне забезпечення для криптоджекінгу в пристрої своїх жертв [32]. Вони скористалися вразливістю, щоб завантажити своє зловмисне програмне забезпечення для криптоджекінгу безпосередньо на комп'ютери жертв і цей вектор атаки дозволяє ефективно і швидко поширювати шкідливе програмне забезпечення для криптоджекінгу.

1.4.3 Типи платформ жертв

Браузери є найбільш часто використовуваними платформами-жертвами, оскільки зловмисникам не потрібно доставляти шкідливе ПЗ жертві, щоб використовувати обчислювальні ресурси жертви. Іншими словами, коли жертва потрапляє на заражену веб-сторінку, зловмисне програмне забезпечення автоматично починає майнінг і не залишає жодних даних. Друга істотна перевага середовища браузера полягає в тому, що завдяки постачальникам послуг готові до використання скрипти для майнінгу можна дуже легко та швидко застосувати у будь-якій веб-сторінці. Однак зловмисники можуть отримати доступ лише до процесорів жертв через браузери, що робить їх неможливими для валют, які дозволяють майнинг ASIC, наприклад біткойн. Тому зразки зловмисного програмного забезпечення для криптоджекінгу, які використовують браузери, здебільшого видобувають Monero або інші криптовалюти, що дозволяє криптомайнінг персональних комп'ютерів на архітектурах ЦП, які не є ASIC.

Персональні комп'ютери, як правило, призначені для того, щоб дозволити кінцевим користувачам виконувати свої щоденні завдання. Персональні комп'ютери постійно оновлюються, щоб могти проводити високорівневі обчислення і дозволити їх користувачам використовувати важкі обчислювальні програми (наприклад, відеоігри, програми візуалізації відео). Зловмисники, які націлені на персональні комп'ютери, прагнуть досягти багатьох жертв, оскільки обмежена кількість жертв не буде прибутковою. Криптоджекінг у браузері, вбудований у популярні веб-сайти, ідеально підходить для цього типу атаки криптоджекінгу. Крім того, вони також можуть ініціювати таку атаку за допомогою широкомасштабних кампаній. Наприклад, дослідники Cisco задокументували свої висновки дворічної кампанії, яка доставляла XMRig у їхнє ПЗ [33]. Вони також помітили, що зловмисне програмне забезпечення «докладає мінімальні зусилля, щоб приховати свої дії» і розміщує шкідливе програмне забезпечення «в онлайн-формах та соціальних мережах» для збільшення кількості жертв.

Внутрішні сервери — це сервери, на яких зберігаються та захищаються дані. Їм віддають перевагу дуже критичні організації, такі як урядові організації для підвищення безпеки та повного контролю над обладнанням та даними. Однак локальні сервери також є ще одним типом платформи-жертви, яку атакують зразки зловмисного програмного забезпечення для криптоджекінгу. Порівняно з персональними комп'ютерами, локальні сервери є більш потужними в обчислювальному відношенні і містять численні служби, доступ до яких здійснюється через багато з'єднань. Це дозволяє зловмисникам отримати більш широку поверхню атаки. Тим не менш, зловмисники повинні знайти спосіб доставки та встановлення сценарію криптомайнінгу на локальному сервері, щоб отримати доступ до цієї платформи. У кількох випадках зловмисники використовували вразливості системи [34], програмне забезпечення сторонніх розробників [35] та кілька методів соціальної інженерії [31], щоб встановити шкідливе програмне забезпечення для криптоджекінгу на локальний сервер жертви.

Зловмисне програмне забезпечення криптоджекінгу також використовує хмарні ресурси для майнінгу криптовалют. Хмарна атака криптоджекінгу є проблемою, яка швидко поширюється протягом останніх двох років, коли вона стала популярною, особливо після закриття Coinhive, коли зловмисники шукали нові платформи для зараження. Зловмисники націлені на кілька вразливостей, щоб захопити хмарні сервери жертв і запустити майнери криптовалюти на цих системах. Хмарні сервери такі як Amazon Web Services (AWS), є мішенню зловмисників через:

- практично нескінченні ресурси;
- велику поверхню атаки через структуру сервера;
- можливості розповсюдження шкідливих програм;
- надійне підключення до Інтернету;
- довший період майнінгу/прибутку завдяки можливостям хоста.

На хмарних серверах було виявлено декілька прикладів цього типу шкідливого програмного забезпечення для криптоджекінгу [36], [37], [38], [39]–[41]. Під час атак зловмисники використовували різні методи, щоб захопити хмарний

сервер, щоб запровадити зловмисне програмне забезпечення для криптоджекінгу. Наприклад, у своєму щорічному звіті за 2020 рік Check Point Research [39] помітив, що зловмисники інтегрують криптомайнер у популярні DDoS-ботнети, такі як KingMiner, націлені на сервери Linux та Windows для отримання прибутків. В іншому випадку [37] дослідники знайшли відкритий каталог, що містить шкідливі файли. Подальший аналіз показав, що файл містить DDoS-бота, який націлений на відкриті порти демона Docker на серверах Docket і в кінцевому підсумку встановлює та запускає зловмисне програмне забезпечення криптоджекінгу після виконання ланцюга зараження. У подібному прикладі атаки [38] дослідники відзначили зловмисне програмне забезпечення для криптоджекінгу, доставлене за допомогою використання CVE, націленого на сервери WebLogic. Amazon сервери, які належали Tesla [36], і клієнти кластерів Azure Kubernetes [41] зазнали атак криптоджекінгу через погано налаштовані хмарні сервери. Jayasinghe et al. [42] показали, що кількість зловмисних програм для криптоджекінгу, спрямованих на хмарну інфраструктуру, збільшується з кожним роком.

Пристрої Інтернету речей (IoT) зазвичай мають невеликі обчислювальні потужності для виконання основних завдань. Очікується, що до 2025 року буде більше 21,5 мільярда пристроїв IoT [43]. Зловмисники мають на меті створити ботнети разом із тисячами цих пристроїв IoT та здійснити атаки, таких як DDoS, завдяки їх невеликому процесору, низькому рівню безпеки та слабких облікових даних, все це було вже використано на прикладі DDoS-атаки ботнету Mirai. Пізніше дослідники IBM також виявили, що модифікована версія ботнету Mirai також почала майнити біткойн. Бартіно та ін. [45] стверджує, що в пристроях IoT є кілька хробаків, які захопили ці пристрої з метою майнінгу, а Ahmad et al. [46] пропонує систему ідентифікації IoT криптоджекінгу для виявлення будь-якої атаки криптоджекінгу, яка зосереджена на пристроях IoT.

Зразки зловмисного програмного забезпечення криптоджекінгу, націленого на мобільні пристрої, вводять скрипт криптоджекінгу в свою програму та показують додаток на ринках програм. Як і будь-який інший тип атак криптоджекінгу, зразків криптоджекінгу на базі мобільних пристроїв також значно стало суттєво більше.

Через це і Google, і Apple видалили програми для криптомайнінгу зі своїх платформ. Проте вони все ще існують на альтернативних ринках. Дослідження Dashevskyi et al. [47] зосереджено на зловмисному програмному забезпеченні для криптоджекінгу на базі Android. Більше того, мобільні пристрої, як правило, не вважаються достатньо потужними для майнінгу криптовалюти, оскільки вони зазвичай використовують більш обмежене обладнання та оптимізовані операційні системи (наприклад, iOS та Android). Крім того, процес майнінгу криптовалюти сильно споживає батарею та процесорну потужність, що може спричинити проблеми з апаратним забезпеченням, таким як перегрів та зависання додатків або збій на мобільних пристроях. Через ці причини зловмисники не надають перевагу мобільним пристроям. Вони можуть добавити фільтр у ПЗ, який буде ідентифікувати чи це мобільний пристрій і просто не запускатися.

```
<script>
 var miner=new CoinHive.Anonymous('Key', {
 Threads:4, autoThreads:false, throttle:0.8);
 if (!miner.isMobile()) &&!miner.didOptOut(14400)
 { miner.start(); } }
</script>
```

Рисунок 1.4 Зразок скрипту Coinhive

На рисунку 1.4 зображений зразок криптоджекінгу з методом фільтрації мобільних пристроїв, знайденим у зразку в нашому наборі даних. У рядку 4 скрипт автоматично викликає функцію виявлення мобільного пристрою і запускає процес майнінгу криптовалюти, тільки якщо це не мобільний пристрій.

1.4.4 Цільові криптовалюти

Monero має ряд переваг перед іншими криптовалютами, що робить його вигідним для зловмисників. Перш за все, Monero успішно реалізує та модифікує

алгоритм майнінгу RandomX і алгоритм хешування CryptoNight, щоб запобігти майнерам ASIC і дати конкурентну перевагу майнерам CPU над GPU через кеш L3. Спільнота Monero прагне зберегти свою мережу децентралізованою і дозволяє навіть невеликим майнерам добувати Monero. Monero надає функції анонімності за допомогою криптографічних кільцевих підписів, що робить зловмисників невідстежуваними. Завдяки цим функціям Monero, зловмисники, як правило, майнять Monero за допомогою зловмисного програмного забезпечення для криптоджекингу в браузері.

В останні роки майнингу біткойнів приділено величезну увагу, що призвело до різкого збільшення цільової складності. Майнери ASIC і FPGA є основною причиною такого різкого зростання, оскільки структура майнінгу біткойна дозволяє створювати та використовувати спеціальне обладнання для майнінгу, яке є набагато потужнішим і вигіднішим, ніж CPU та GPU. Тому зловмисники, які здійснюють атаку криптоджекінгу в браузері, не віддають перевагу майнінгу біткойнів.

1.4.5 Методи виявлення і уникнення

У традиційній літературі з виявлення шкідливих програм розглядають два основних методи аналізу: статичний і динамічний. Обидва методи аналізу мають свої плюси і мінуси з точки зору точності та зручності використання.

Статичний аналіз є широко використовуваним методом для перевірки програми без їх запуску. Інструменти статичного аналізу зазвичай шукають конкретні ключові слова, сигнатури шкідливих програм і хеш-значення. Існують розширення браузера, що блокують майнінг [48], [49]. Вони блокують будь-який домен, зазначений у попередньо визначеному чорному списку. Головним недоліком статичних методів виявлення полягає в тому, що їх легко обійти.

Під час динамічного аналізу зразок шкідливого програмного забезпечення виконується в контрольованому середовищі, а його поведінкові особливості записуються для подальшого аналізу та виявлення. Аналізатори шкідливих програм зазвичай використовують автоматизовані [50] або неавтоматизовані пісочниці [51]

для запуску коду та спостереження за поведінкою зловмисного програмного забезпечення.

1.4.6 Техніки ухилення та обфускації

Метою зловмисного програмного забезпечення криптоджекінгу є експлуатація ресурсів жертви якомога довше; тому залишатися в системі без виявлення є надзвичайно важливо. Для цього вони використовують кілька методів обфускації.

Висока завантаженість ЦП все ще є найважливішою спільною точкою всіх видів шкідливих програм для криптоджекінгу, оскільки використання ЦП є основною вимогою процесу майнінгу криптовалюти. Тому обмеження ЦП є дуже бажаним методом для зловмисників для обфускації скрипта майнінгу. За допомогою цього методу власники скриптів можуть обійти системи виявлення на основі високого використання ЦП і уникнути потрапляння в чорний список. Крім того, обмеження ЦП також використовується законними власниками веб-сайтів, які займаються майнінгом криптовалюти як альтернативний дохід, оскільки це забезпечує кращу роботу користувачів. На малюнку 1.4 показано приклад методу обмеження ЦП, який використовується зловмисним програмним забезпеченням для криптоджекінга, коли зловмисник встановлює параметр throttle на 0,8, наприклад, зловмисник хоче використовувати лише 20% навантаження ЦП для майнінгу криптовалюти.

Виклики бібліотеки [52] — це добре відомий прийом, який використовується програмістами, щоб зробити код більш ефективним, систематичним і читабельним. Однак його також можуть використовувати зловмисники, щоб приховати свої скрипти. Зокрема, щоб приховати код майнінгу від методів виявлення, зловмисники створюють нові скрипти, які не мають конкретних ключових слів. Шкідлива частина скрипту переміщується до зовнішньої бібліотеки, яка викликається під час виконання скрипта, і лише фрагмент коду для виклику цієї бібліотеки включається в основний код.

Кодування вихідного коду шкідливого програмного забезпечення за допомогою кількох алгоритмів кодування забезпечує невидимість проти методів статичного аналізу на основі ключових слів, таких як чорні списки. Цей метод перетворює текстові дані в іншу форму, таку як Base64 [70], і після цього процесу дані можуть бути прочитані лише комп'ютерами.

Подібно до техніки кодування коду, бінарна обфускація є практикою серед авторів шкідливих програм, щоб приховати шкідливий код від стандартних алгоритмів зіставлення рядків і ускладнити відновлення за допомогою пісочниць та інших динамічних методів виявлення зловмисного програмного забезпечення. Однак вони відрізняються за типом криптоджекінгу, який використовується для приховування, тобто бінарна обфускація використовується зловмисним програмним забезпеченням для криптоджекінгу на базі хоста, тоді як кодування коду використовується зловмисним програмним забезпеченням для криптоджекінгу у браузері. Автори роботи [53] зазначили, що 30% 1,2 млн зразків зловмисного програмного забезпечення для шкідливого програмного забезпечення для криптоджекінгу обфусковані, що показує, що це також поширена практика серед зловмисників криптоджекінгу.

1.5 Аналіз джерел на тему криптоджекінгу

Сплеск шкідливого ПЗ, особливо після 2017 року, також привернув увагу науковців і призвів до багатьох публікацій. Виявлено, що дослідження переважно зосереджені на трьох темах: дослідження виявлення криптоджекінгу, дослідження запобігання криптоджекінгу та емпіричні дослідження криптоджекінгу. Серед наукових дослідницьких робіт виявлено, що 15 з них зосереджені на експериментальному аналізі набору даних криптоджекінгу. У той же час 3 з них разом пропонують метод виявлення та запобігання зловмисному програмному забезпеченню криптоджекінгу, а 24 з них пропонують лише метод виявлення зловмисного програмного забезпечення для криптоджекінгу. У наступних

підрозділах ми даємо огляд цих досліджень і оцінимо, які методи найбільш досліджені, щоб краще розуміти загальну ситуацію.

1.5.1 Дослідження виявлення криптоджекінгу

Датасети зазвичай використовується для оцінки ефективності запропонованого методу виявлення. Деякі датасети активно використовуються в літературі з виявлення зловмисного програмного забезпечення криптоджекінга. Найпоширенішим є веб-сторінки Alexa [54]-[59]. Alexa сортує найбільш відвідувані веб-сайти в Інтернеті; однак він не надає вихідний код для цих веб-сайтів. Тому в цих дослідженнях також використовувався протокол налагодження Chrome для інструментування браузера та збору необхідної інформації з веб-сайтів, за винятком дослідження [59], яке працює з обмеженою кількістю (500) веб-сайтів. Більше того, у дослідженні [55] також використовувалися відомі та часто оновлювані чорні списки [48], [49], [60], щоб створити основу для їхніх тренувальних датасетів, а потім вони провели аналіз за допомогою веб-сайтів Alexa. Також у дослідженні [61] використовувався набір даних криптоджекінгу, отриманий від VirusTotal [68]. Вони зібрали 1500 активних зразків шкідливих програм для майнінгу криптовалюти Windows Portable Executable (PE32), зареєстрованих у 2018 році, і використали Cuckoo Sandbox [62] для отримання детальних звітів про поведінку цих зразків. Крім того, дослідження в [63]-[66] проводили свій аналіз шляхом встановлення законних скриптів для майнінгу, а дослідження в [56], [67] вручну вводили майнери на веб-сайти, щоб перевірити їх механізми виявлення.

Більшість механізмів виявлення криптоджекінгу в літературі пропонуються для виявлення криптоджекінгу у браузері. Існує лише кілька досліджень [61], [69], запропонованих щодо зловмисного програмного забезпечення для криптоджекінгу на основі хоста. Крім того, Conti та інші [66] пропонують механізм виявлення на апаратному рівні, який можна використовувати для виявлення зловмисного програмного забезпечення для криптоджекінгу як на хості, так і в браузері.

У області криптоджекінгу більшість запропонованих методів виявлення використовують динамічний аналіз. Основна причина цього полягає в тому, що скрипти майнінгу використовують набір відомих інструкцій. Наприклад, майнери використовують криптографічні хеш-бібліотеки та багаторазово збільшують значення статичної змінної (тобто nonce) або підключаються до деяких відомих постачальників послуг, щоб продовжувати завантажувати деякі результати та отримувати нові завдання. Ці типові дії зловмисного програмного забезпечення для криптоджекінгу створюють шаблон, що дозволяє їх виявити за допомогою динамічного аналізу. У літературі лише кілька досліджень використовують статичні функції, такі як коди операцій [61] та інструкції WebAssembly (Wasm) [72]. WebAssembly [73] — це низькорівневий формат інструкцій, який дозволяє програмам працювати ближче до мови машинного рівня та забезпечувати вищу продуктивність за допомогою віртуальних машин на основі стека [74]. Ця низькорівнева модель інструкцій дозволяє WebAssembly виконувати код більш ефективно і ця функція забезпечує більший прибуток, оскільки скрипт криптоджекінгу працює швидше. Усі основні браузери на ринку зараз підтримують WebAssembly.

Коди операцій — це інструкції на машинній мові, які визначають операції, які мають виконуватися, і використовуються системними викликами. Запропонована система виявлення в [61] використовує коди операцій для статичного аналізу, де коди операції витягуються за допомогою IDA Pro [26]. У прикладі криптоджекінгу коди операцій зосереджені на запитах між скриптами майнінгу та ядром операційної системи. За допомогою цього методу вони досягають 95% точності за допомогою класифікатора Random Forest.

З іншого боку, є багато механізмів виявлення, де були використані динамічні параметри. У цих дослідженнях найчастіше використовуються такі динамічні характеристики:

- Події ЦП [54]-[56], [58], [64], [75]-[80]: події ЦП є найбільш часто використовуваними характеристиками серед механізмів виявлення на основі динамічного аналізу, оскільки скрипти криптоджекінгу в браузері

повинні отримати інструкції ЦП для виконання майнінгу, незалежно від використаного обладнання. Якщо операція в браузері використовує криптографічні бібліотеки занадто часто, що є ненормальним для звичайних веб-сайтів, це може бути безпосередньо виявлено інструкціями ЦП. Незважаючи на те, що ЦП є найважливішою особливістю майнінгу криптовалюти, використання лише подій ЦП як функцій може спричинити високі хибнопозитивні показники (FPR), оскільки веб-сайти флеш-ігор або веб-сайти з онлайн-рендерингом також активно використовують ЦП системи для своїх операцій. Щоб підтримувати FPR якомога нижчим, більшість методів виявлення використовують більше однієї функції одночасно.

- Діяльність в пам’яті [54], [64], [56], [70]: діяльність в пам’яті є ще однією часто використовуваною ознакою серед методів динамічного виявлення.
- Мережеві пакети [54], [56], [63]-[65], [69]: мережеві пакети також є зручним і корисним методом для виявлення активності криптоджекінгу через масивний мережевий трафік, який створюється під час завантаження обчислених хеш-значень до постачальника послуг. У дослідженнях [54], [56], [63], [64] використовувалася швидкість мережевого трафіку як додаткова функція разом з іншими функціями, такими як функції, пов’язані з пам’яттю та процесором. З іншого боку, дослідження в [65], [69] використовували лише мережеві пакети для виявлення зловмисного програмного забезпечення криптоджекінга. Зокрема, Neto та інші. [65] використовують мережевий потік як функцію, тоді як Caprolu та ін [69] використовували час між надходженням і розміри пакетів як особливості в їхньому алгоритмі виявлення.
- Час компіляції та виконання JavaScript (JS): У [55], [76] було показано, що на час виконання движка JS і час компіляції JS значно впливає

зловмисне програмне забезпечення криптоджекінгу. Однак онлайн-ігри та інші онлайн-платформи рендеринга також можуть викликати подібну поведінку, викликаючи помилкові спрацьовування в механізмі виявлення. Тому в дослідженні [55] також використовуються навантаження на процесор, збирач сміття та навантаження ресурсів iframe [81] як вторинні функції для отримання більш точних результатів і зменшення хибних результатів.

- Збірник сміття — це функція мови програмування JS для оптимізації використання пам’яті, вона видаляє непотрібні дані з пам’яті та запобігає перевантаженню пам’яті. Пам’ять і ЦП безперервно взаємодіють один з одним під час майнінгу, і ЦП надсилає обчислені дані в пам’ять. Збірник сміття видаляє всі обчислені хеш-значення по черзі після відправки постачальнику послуг; тому процес видобутку спричиняє нерегулярне використання сміттєзбірника. Завдяки такій поведінці збирач сміття можна використовувати як функцію для механізму динамічного виявлення. Iframe — це теги HTML, які використовуються для вбудовування іншої програми/функції у вихідний код HTML. Скрипти для майнінгу вставляються в ці теги і працюють під HTML-кодами. Подібно до попередніх функцій, скрипти криптоджекінгу викликають нерегулярне використання під час завантаження ресурсів iframe. Цю функцію не можна використовувати як основну функцію, оскільки занадто багато сучасних веб-додатків використовують ресурси iframe нерегулярно, і це може спричинити високий рівень помилкових спрацьовувань.
- Лічильник продуктивності обладнання (HPC) [67], [76], [80]: значення HPC [82] використовуються в процесорах сучасних комп’ютерів і ведуть облік внутрішніх подій ЦП (наприклад, цикли, промахи кешу). Значення регістрів із тактовими циклами ЦП та виконуваними інструкціями надають унікальну інформацію про поведінку запущеної програми.

Кілька досліджень перевіряють апаратну діяльність і пов’язані програми зі значеннями HPC, щоб виявити операції з видобутку криптовалюти на системи.

- Системні виклики [61]: системні виклики - це структури API, які забезпечують з'єднання між додатками та ядром запущеної системи. Системні виклики виконуються з привілеями рівня 0, щоб викликати виклики та запитувати послуги з ядра ОС. Запропонована система виявлення в [61] використовує системні виклики для динамічного аналізу, а системні виклики записуються за допомогою Cuckoo Sandbox. Потім системні виклики використовуються для навчання моделей глибокого навчання, і вони досягають 99% точності.

1.5.2. Дослідження запобігання криптоджекінгу

Більшість механізмів виявлення не зосереджені на запобіганні чи перериванні зловмисного програмного забезпечення для криптоджекінгу; однак є ще кілька досліджень [75], [77], [83], які зосереджуються як на виявленні, так і на запобіганні зловмисного програмного забезпечення для криптоджекінгу. Використання динамічних функцій для виявлення постійного криптоджекінгу схоже на інші дослідження динамічного аналізу, але методи запобігання їм відрізняються. Тоді як Юліанто та ін. [77] створюють нотифікації, Віапта ін. [75] усипають процес майнінгу, а Razali та ін. [83] безпосередньо вбивають пов'язаний процес.

Для запобігання криптоджекінгу на ринку також є кілька інструментів. Проти зловмисного програмного забезпечення для криптоджекінгу на базі хостів зазвичай перевага надається запатентованим антивірусним програмам. Проти зловмисного програмного забезпечення для криптоджекінгу в браузері широко використовуються розширення браузера з відкритим кодом, такі як NoCoin [48] і MinerBlock [49]. Ці розширення для браузера з відкритим вихідним кодом засновані на чорному списку, де списки оновлюються в міру виявлення нових шкідливих доменів. Розширення

браузера попереджають користувача, коли користувач хоче отримати доступ до веб-сайту з чорного списку.

Чисте блокування на основі чорного списку не є ефективним способом зупинити зловмисне програмне забезпечення криптоджекінга, оскільки зловмисники можуть легко змінити свій домен за допомогою зміни домену або інших методів, щоб зменшити вплив чорних списків. Існують також деякі нові методи [84], запропоновані дослідниками для кращого та більш оптимізованого створення чорного списку, але навіть методи динамічного внесення до чорного списку не є повністю ефективними та не захищають [85] від методів змінення домену.

1.5.3. Емпіричні дослідження криптоджекінгу

У додачу до досліджень виявлення та запобігання зловмисному програмному забезпеченню, деякі дослідники також проводили емпіричні дослідження, щоб краще зрозуміти ландшафт загроз криптоджекінгу. У цих дослідженнях криптомайнери мають або двійковий формат або скрипт.

Дослідники проаналізували кілька різних точок зору криптоджеккінгу. У першому дослідженні [86] автори аналізують бінарні вибірки, щоб охарактеризувати їх обсяг, операції та дохід. Це перше і єдине дослідження з аналізу майнерів біткойна, бо зразки, використані в інших дослідженнях, видобувають Monero. Увагу дослідників також привернуло збільшення випадків атак зловмисного програмного забезпечення криптоджекінг у 2017 році. [87] є першим дослідженням, що аналізує зразки криптоджекінгу Monero, де автори використовували понад 30000 веб-сайтів, які використовують бібліотеку coinhive.min.js для аналізу поширеності зразків криптоджекінгу. У дослідженнях [88], [89] проведено аналіз впливу криптоджекінгу. Зокрема, [89] проаналізував вплив атаки на споживчі пристрої та роздратування користувачів, а також [88] проаналізував вплив зловмисного програмного забезпечення криптоджекінгу на веб-користувачів.

1.6 Постановка задачі

Об’єктом дослідження є процес несанкціонованого доступу до ресурсів кінцевих систем, де головна увага зосереджена на дослідження процесів криптомайнерів.

Оскільки дана задача включає в себе виконання широкого діапазону робіт (на основі попереднього аналізу) і орієнтована на постійне доопрацювання та удосконалення, то метою цієї дипломної роботи є виявлення аномалії ІБ, які вказують на діяльність криптомайнера на хості, розробка необхідного програмного забезпечення до основних етапів виявлення та ідентифікації, візуалізація отриманих результатів.

Відповідно до цілі роботи, вирішувались наступні завдання:

- дослідження існуючих рішень для виявлення криптомайнерів
- аналіз аномальних процесів пов’язаних з діяльністю криптомайнерів
- розробка методу виявлення криптомайнерів
- створення програмного модуля методу виявлення криптомайнерів
- синтез моделі виявлення аномальної поведінки програмного забезпечення
- аналіз мережевого трафіку криптомайнерів
- покращення методу виявлення аномальної поведінки поєднуючи інформацію про мережевий трафік та синтезовану модель.

Криптомайнери містять свої технологічні особливості та відмінності. Для проведення комплексного дослідження в роботі розглядалося декілька прикладів криптомайнерів для виявлення достовірної аномальної поведінки.

Висновки до розділу 1

У першому розділі були дослідження різні наукові роботи, які зосередженні на виявлення діяльності криптомайнера, як в середовищі браузерів так і на хостах. Більша частина робіт зосередженні на аналізі мережевого трафіку пов’язаного з криптомайнингом і на певних статистичних характеристикам, які притаманні криптомайнингу.

У другому розділі буде проведено аналіз поведінки криптомайнерів на рівні машинного коду і системних процесах, які їм характерні, запропонований метод, який дозволить виявляти аномальну поведінку та розроблений програмний модуль, який спростить аналіз такої поведінки.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ДЛЯ ВИЯВЛЕННЯ КРИПТОДЖЕКІНГУ

2.1 Моделювання поведінки криптомайнерів

Криптомайнинг з технічної сторони це постійне вичислення певних хеш-функцій. Криптографічна хеш-функція (CHF) — це математичний алгоритм, який відображає дані довільного розміру (часто званого «повідомленням») у бітовий масив фіксованого розміру («хеш-значення», «хеш» або «дайджест повідомлення»). Це одностороння функція, тобто функція, для якої практично неможливо інвертувати чи повертати обчислення. В ідеалі, єдиний спосіб знайти повідомлення, яке рахує певна хеш-функція, — це спробувати пошук методом грубої сили можливих вхідних даних, щоб побачити, чи збігаються вони, або використовувати райдужну таблицю відповідних хешів. Криптографічні хеш-функції є основним інструментом сучасної криптографії. [90]

Криптографічна хеш-функція повинна бути детермінованою, що означає, що одне й те саме повідомлення завжди призводить до того самого хеша. В ідеалі він також повинен мати такі властивості [90]:

- він швидко обчислює хеш-значення для будь-якого даного повідомлення
- неможливо створити повідомлення, яке дає задане хеш-значення (тобто повернути назад процес, який створив дане хеш-значення)
- неможливо знайти два різних повідомлення з однаковим хеш-значенням
- невелика зміна в повідомленні повинна змінити хеш-значення настільки сильно, що нове хеш-значення здається некорелюваним зі старим хеш-значенням (ефект лавини)

Криптографічні хеш-функції мають багато додатків для захисту інформації, зокрема в цифрових підписах, кодах аутентифікації повідомлень (MAC) та інших формах аутентифікації. Їх також можна використовувати як звичайні хеш-функції,

для індексації даних у хеш-таблицях, для відбитків пальців, для виявлення повторюваних даних або однозначної ідентифікації файлів, а також як контрольні суми для виявлення випадкового пошкодження даних. Дійсно, в контекстах інформаційної безпеки криптографічні хеш-значення іноді називають (цифровими) відбитками пальців, контрольними сумами або просто хеш-значеннями, хоча всі ці терміни означають більш загальні функції з досить різними властивостями та цілями.

Для біткойна майнери повинні вирішити математичну головоломку з хешуванням, знайшовши хеш нижче заданої цільового хешу за допомогою вимоги складності майнингу.

Цільовий хеш — це числове значення, якому хешований заголовок блоку (який використовується для ідентифікації окремих блоків у блокчейні) має бути меншим або рівним, щоб майнер отримав новий блок. Цільовий хеш, що зберігається в заголовку, виражається у вигляді 67-значного числа, яке визначатиме складність майнінгу на основі кількості майнерів, які змагаються за вирішення хеш-функції.

Складність майнингу відноситься до складності розв’язання математичної головоломки та генерації біткойнів. Складність майнингу впливає на швидкість генерації біткойнів. Складність видобутку змінюється кожні 2016 блоків або приблизно кожні два тижні. Наступний рівень складності залежить від того, наскільки ефективними були майнери в попередньому циклі. На це також впливає кількість нових майнерів, які приєдналися до мережі біткойн, оскільки це збільшує хешрейт або кількість обчислювальної потужності, що використовується для майнінгу криптовалюти. Чим більше майнерів змагаються за рішення, тим складнішою буде проблема. Якщо обчислювальна потужність виключена з мережі, складність зменшується, щоб полегшити майнинг.

Щоб вирішити математичну хеш-головоломку, майнери намагатимуться обчислити хеш блоку, додаючи одноразовий номер до заголовка блоку кілька разів, поки отримане хеш-значення не стане менше цільового. Як тільки комп’ютер для майнингу розв’язує головоломку, успішно створюється новий блок, який

перевіряється в мережі Bitcoin після досягнення консенсусу між вузлами. Коли блок перевіряється, транзакції, об’єднані в нього, перевіряються, і блок додається до ланцюжка.

Оскільки за вирішення головоломки змагатимуться багато майнерів (систем), перший майнер, який отримає правильне значення хеша, отримує винагороду в біткойнах. Потрібно врахувати, що різні хеш-функції використовуються різними криптовалютами.

SHA-256

Алгоритм SHA спочатку був розроблений Агентством національної безпеки США в 2002 році. У 2009 році Proof-of-Work SHA256 був реалізований в біткойні, а пізніше в інших подібних криптовалютах.

Алгоритм SHA256 створює 256-бітний хеш. Швидкість і ефективність декодування залежать від обчислювальної потужності майнера. Ймовірність знаходження цільового хешу дорівнює відношенню обчислювальної потужності майнера до потужності всієї мережі. Тому з’явилося спеціальне майнингове обладнання для підвищення потужності – ASIC-майнери.

Scrypt

Scrypt є одним з найпопулярніших алгоритмів хешування PoW разом із SHA256. Зараз він використовується в Litecoin, Dogecoin та інших криптовалютах. Цей алгоритм є більш складним, оскільки вимагає багато пам’яті, наявної на обладнанні для майнінгу. Це було великою проблемою для ASIC-майнерів.

Scrypt мав на меті запобігти монополізації майнінгу ASIC, і спочатку це дійсно вийшло. Сьогодні обладнання ASIC також використовується для майнінгу криптовалюти за алгоритмом scrypt, що робить процес більш централізованим і менш доступним.

Cryptonight

CryptoNight — це алгоритм PoW. Він розроблений, щоб бути придатним для центрального процесора звичайного комп’ютера, але на даний момент немає реалізованих пристроїв спеціального призначення для майнінгу. CryptoNight спочатку був зроблений в кодовій базі CryptoNote. CryptoNight поки що можна

добувати лише на процесорі. На відміну від алгоритму Scrypt, алгоритм Cryptonight залежить від усіх попередніх блоків для кожного нового блоку.

Ethash

Алгоритм dagger-Hashimoto дозволив майнити Ethereum (а пізніше Ethereum Classic). Алгоритм був схожий на scrypt, але потребував ще більше доступної пам'яті. Це захистило майнинг Ethereum на ASIC. Потім алгоритм був удосконалений і названий Ethash. Користувачі мають можливість майнити Ethereum за допомогою відеокарт [95].

Ethash використовується лише для майнінгу та використовує алгоритм хешування «Proof of Work», розроблений спеціально для та Ethereum (ETH). Основною причиною створення функції Ethash PoW є опір машинам ASIC. Таким чином, Ethash стійкий до ASIC, і це алгоритм із інтенсивною пам'яттю, який можна видобути лише за допомогою графічного процесора.

X11

Алгоритм X11 використовує 11 різних алгоритмів, які взаємопов'язані один з одним, тому це назва алгоритму ланцюжкового хешування. Для захисту криптовалюти в криптовалютній мережі алгоритм x11 використовує обчислення «Proof of Work».

Алгоритм X11 вимагає лише на 30% менше потужності, тому що він більш енергоефективний, ніж Scrypt, це його основна перевага, тому йому віддають перевагу над Scrypt.

Якщо оцінювати хеш-функції зі сторони машинного коду, то це постійна робота з бітами, тобто їх перестановка, здвиги тощо. Тобто, можна створити гіпотезу, що додатки, які задіяні в криптоманингу буду мати абнормальну кількість операцій, які пов'язані зі зміною бітів.

Зі всього стеку операцій на архітектурі «x86» виділено наступні категорії операцій [96]:

- Rotate
- Shift
- Logical Exclusive OR

Rotate

Зсуває біти першого операнда (операнда призначення) кількість бітових позицій, зазначених у другому операнді (операнді лічильника), і зберігає результат в операнді призначення. Операнд призначення може бути регістром або місцем пам'яті; операнд count — це ціле число без знака, яке може бути безпосереднім або значенням у регістрі CL. Процесор обмежує підрахунок числом від 0 до 31, маскуючи всі біти в операнді лічильника, крім 5 найменш значущих бітів.

До цієї категорії відносяться наступні операції:

- RCL
- RCR
- ROL
- ROR

Shift

Зсуває біти в першому операнді (операнді призначення) вліво або вправо на кількість бітів, зазначену в другому операнді (операнді лічильника). Біти, зміщені за межі операнда призначення, спочатку зміщуються в прапор CF, а потім відкидаються. Наприкінці операції зсуву прапор CF містить останній біт, зміщений з операнда призначення.

Операнд призначення може бути регістром або місцем пам'яті. Операнд рахунку може бути безпосереднім значенням або регістром CL. Підрахунок маскується до 5 біт, що обмежує діапазон підрахунку від 0 до 31. Для підрахунку 1 передбачено спеціальне кодування коду операції.

До цієї категорії відносяться наступні операції:

- SHL
- SHR
- SHRD
- SHLD

Logical Exclusive OR

Виконує операцію порозрядного виключного АБО (XOR) над операндами призначення (перший) і вихідним (другим) і зберігає результат у місці операнда

призначення. Операнд джерела може бути безпосереднім, регістром або місцем пам'яті; операнд призначення може бути регістром або місцем пам'яті. (Однак два операнди пам'яті не можна використовувати в одній інструкції.) Кожен біт результату дорівнює 1, якщо відповідні біти операндів різні; кожен біт дорівнює 0, якщо відповідні біти однакові.

2.2 Дослідження закономірностей криптомайнерів

У попередньому підрозділі була описана важливість хеш-функцій і операцій Rotate, Shift, XOR. Для того, щоб дослідити додатки і визначити кількість операцій, які були зроблені протягом виконання ПЗ буде використовуватися Intel Software Development Emulator.

Intel SDE — це емулятор, який дозволяє запускати код з наборами інструкцій на системах, які не підтримують ці інструкції. Слід зазначити, що SDE корисний для оцінки функціональності, але не продуктивності, оскільки він запускає програми в рази повільніше, ніж, якщо б його запускати на обладнанні. [71]

SDE враховує лише інструкції, які виконуються (динамічні) під час роботи програми, а не інструкції, що існують у коді (статичні). Це дуже хороший спосіб налагодити програму на випадок, якщо виникне проблема, якщо передбачається, що певні інструкції будуть виконуватися в певних частинах коду. Якщо в певному блоці адрес, що відповідають певним частинам коду, не було виявлено очікуваних інструкцій, то це може бути індикатором неочікуваної/несвідомої умови розгалуження.

Запуск програми з різними параметрами або введенням також спричинить різну поведінку та динамічне виконання, тому не можна вважати, що один запуск програми передає всю історію. Рекомендується запускати програму під різними робочими навантаженнями під час дослідження процесів, які виконуються в додатках.

На рисунок 2.1 зображений запуск програми «notepad.exe» в середовищі емулятора.

```
C:\sde-external-9.0.0-2021-11-07-win\sde-external-9.0.0-2021-11-07-win>sde.exe -mix -omix out_notepad.txt -- notepad.exe
```

Рисунок 1.1 Приклад запуску «notepad.exe» через SDE

Після виконання, SDE створить файл out_notepad.txt. Цей файл буде містити інформації про те, які бібліотеки були завантажені, найпоширеніші блоки інструкцій, які функції були задіяні і інформацію про операції, які були виконані. Приклад інформації з файлу зображений на рисунку 2.2.

```
# Mix output version 10
# Intel(R) SDE version: 9.0.0 external
# Starting tid 0,  OS-TID 24492
# Starting tid 1,  OS-TID 19756
# Starting tid 2,  OS-TID 12668
# Starting tid 3,  OS-TID 22504
# Starting tid 4,  OS-TID 26732
# Starting tid 5,  OS-TID 18596
# Starting tid 6,  OS-TID 2532
# Starting tid 7,  OS-TID 26536
# Starting tid 8,  OS-TID 25880
# Starting tid 9,  OS-TID 12968
# Starting tid 10,  OS-TID 24668
# FINI: end of program

# EMIT_IMAGE_ADDRESSES
#
#    IMAGE NAME                                                                                                  LOW ADDRESS   HIGH ADDRESS
#
C:\Windows\SysWOW64\notepad.exe                                                                                  000000200000  00000022dfff
C:\Windows\SysWOW64\kernel32.dll                                                                                 000075d60000  000075e4ffff
C:\Windows\SysWOW64\KernelBase.dll                                                                               000075e70000  000076083fff
C:\Windows\SysWOW64\ntdll.dll                                                                                    0000776b0000  000077852fff
C:\Windows\SysWOW64\gdi32.dll                                                                                    0000759a0000  0000759c2fff
C:\Windows\SysWOW64\win32u.dll                                                                                   000075e50000  000075e67fff
C:\Windows\SysWOW64\gdi32full.dll                                                                                000076e60000  000076f3bfff
C:\Windows\SysWOW64\msvcp_win.dll                                                                                000075920000  00007599afff
C:\Windows\SysWOW64\ucrtbase.dll                                                                                 0000755c0000  0000756dffff
C:\Windows\SysWOW64\user32.dll                                                                                   000076c50000  000076debfff
C:\Windows\SysWOW64\combase.dll                                                                                  000075a80000  000075cfffff
C:\Windows\SysWOW64\rpcrt4.dll                                                                                   0000770b0000  00007716dfff
C:\Windows\SysWOW64\SHCore.dll                                                                                   000075730000  0000757b6fff
C:\Windows\SysWOW64\msvcrt.dll                                                                                   000077170000  00007722efff
C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.19041.1110_none_a8625c1886757984\COMCTL32.dll
```

Рисунок 2.2 Приклад файлу, який отримуються утилітою SDE

Найважливіша інформація для нас знаходиться в кінці файлу, де перераховані всі інструкції, які були здійснені за період виконання програми. Приклад такого списку можна побачити на рисунку 2.3.

CMOVNB	45
CMOVNBE	48
CMOVNL	2
CMOVNLE	15
CMOVNS	258
CMOVNZ	1153
CMOVS	10
CMOVZ	568
CMP	8440186
CMPXCHG8B_LOCK	1100
CMPXCHG_LOCK	87969
CPUID	90
CVTTSD2SI	5443
CWDE	2338
DEC	219315
DEC_LOCK	486
DIV	11133
DIVSD	2
FADD	3489
FADDP	3156
FCHS	36
FCOM	302
FCOMP	276
FCOMPP	53
FCOS	1
FDIV	3343
FDIVP	15
FDIVRP	15
FILD	7029
FISTP	8
FLD	26672
FLD1	12081
FLDCW	366
FLDZ	1387
FMUL	3324
FMULP	49
FNCLEX	2
FNSTCW	369
FNSTSW	7787

Рисунок 2.3 Перелік операцій у вихідному файлі утиліти SDE

Таким чином, ми можемо зібрати інформацію про інструкції, які були виконані програмою.

Але перед тим, як йти далі, необхідно оптимізувати виконання SDE. На даному етапі, без додаткових конфігурацій out_notepad.txt сягав 7 мБайт, що є дуже великою цифрою для такої простої програми.

SDE дозволяє ігнорувати багато інформації і таким чином наша кінцева команда буде мати наступний вигляд:

sde.exe -mix_omit_per_function_stats 1 -top_blocks 0 -mix_max_cumulative 0 -mix_omit_per_thread_stats 1 -omix out_notepad.txt -- notepad.exe

Після добавлення прапорців оптимізації, кінцевий файл важив тільки 30 кБайт, а це зменшення в обсязі в 233 рази. SDE був налаштований тільки збирати інформацію операції і робити це вже на глобальному рівні, тобто не створювати жодної статистики для конкретних потоків, блоків та функцій, тільки сумарну інформацію.

2.3 Метод дослідження поведінки додатків

Для створення програми для методу дослідження додатків буде задіяна мова програмування Python. Python — об'єктно-орієнтована мова загального призначення, розроблена з метою підвищення продуктивності програміста. Синтаксис Python є більш зрозумілим для новачка [97]. Ця мова програмування логічна, лаконічна та зрозуміла. У порівнянні з багатьма іншими мовами, Python має легкочитаємий синтаксис.

Python кросплатформенний: підходить для різних платформ: і Linux, і Windows. В Python також є реалізація інтерпретаторів для мобільних пристроїв та непопулярних систем.

Варто відзначити широке застосування цієї мови, адже вона використовується для розробки веб-застосунків, ігор, зручна для автоматизації, математичних обчислень, машинного навчання, в області інтернету речей. Навіть існує реалізація під назвою Micro Python, оптимізована для запуску мікроконтролерів (можна писати інструкції, логіку взаємодії пристроїв, організовувати зв'язок, реалізовувати розумний будинок).

Розроблене програмне забезпечення буде виконувати наступні функції:

- запуск SDE з програмним забезпеченням, яке досліджується
- відкриття файлу результатів SDE
- парсинг результатів
- аналіз і візуалізація отриманих результатів

На рисунку 2.4 зображена схема програми

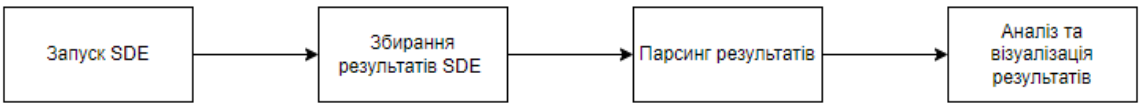


Рисунок 2.4 Етапи виконання програми

Щоб запустити команду SDE буде використана бібліотека subprocess. Модуль subprocess дозволяє створювати нові процеси, підключатися до їх каналів введення/виводу/помилки та отримувати їх коди повернення.

З цього модуля для нас важливий клас Popen() та метод wait(). Створення базового процесу та керування ним у цьому модулі subprocess обробляється класом Popen. Він пропонує велику гнучкість, щоб розробники могли обробляти менш поширені випадки, не охоплені зручними функціями. Метод wait() дозволяє нам чекати до моменту, як виконання програми буде завершено і не продовжувати виконання коду до поки цей момент не настане.

На рисунку 2.5 зображено, як ці методи задіяні в запропонованому рішенні:

```
def main(filepath, minutes, outfile="out_python.txt", ):
    print(f"Initiating File: {filepath}")
    command = f"C:\sde-external-9.0.0-2021-11-07-win\sde-external-9.0.0-2021-11-07-win\sde.exe
    print(f"Prepared Command: {command}")
    process = subprocess.Popen(command, shell=False)
    process.wait()
    print(f"Finished File: {filepath}\nStarting reading statistics...")
    process_file(filename=outfile, minutes=minutes)
```

Рисунок 2.5 Функція «main» програми

Тобто, спочатку скрипт отримує на вхід інформацію за те, який додаток повинен бути дослідженим, задається назва для вихідного файлу. Далі готується команда для SDE з урахуванням всіх прапорців оптимізації і безпосереднє виконання цієї команди. Програма не продовжить виконання, поки не буде закінчений запущений процес.

Наступний крок, це відкриття створеного файлу, зчитування з нього інформації та парсинг.

Для відкриття і зчитування задіяний метод open. Незважаючи на те, що було зроблено багато оптимізації, вихідний файл все одно має багато інформації, яка для дослідження цінності не має. На рисунку 2.6 зображений результат виконання SDE з прапорцями для оптимізації:

```
C:\Windows\SysWOW64\notepad.exe                                                                    000000200000
C:\Windows\SysWOW64\kernel32.dll                                                                   000075d60000
C:\Windows\SysWOW64\KernelBase.dll                                                                 000075e70000
C:\Windows\SysWOW64\ntdll.dll                                                                      0000776b0000
C:\Windows\SysWOW64\gdi32.dll                                                                      0000759a0000
C:\Windows\SysWOW64\win32u.dll                                                                     000075e50000
C:\Windows\SysWOW64\gdi32full.dll                                                                  000076e60000
C:\Windows\SysWOW64\msvcp_win.dll                                                                  000075920000
C:\Windows\SysWOW64\ucrtbase.dll                                                                   0000755c0000
C:\Windows\SysWOW64\user32.dll                                                                     000076c50000
C:\Windows\SysWOW64\combase.dll                                                                    000075a80000
C:\Windows\SysWOW64\rpcrt4.dll                                                                     0000770b0000
C:\Windows\SysWOW64\SHCore.dll                                                                     000075730000
C:\Windows\SysWOW64\msvcrt.dll                                                                     000077170000
C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.19041.1110_none_a8625c18867579
C:\Windows\SysWOW64\imm32.dll                                                                      000077500000
C:\Windows\SysWOW64\bcryptprimitives.dll                                                           000076f40000
C:\Windows\SysWOW64\advapi32.dll                                                                   0000757c0000
C:\Windows\SysWOW64\sechost.dll                                                                    000075840000
C:\Windows\SysWOW64\kernel.appcore.dll                                                             000075530000
C:\Windows\SysWOW64\uxtheme.dll                                                                    0000721e0000
C:\Windows\SysWOW64\clbcatq.dll                                                                    000076580000
C:\Windows\SysWOW64\MrmCoreR.dll                                                                   00007c2d0000
C:\Windows\SysWOW64\shell32.dll                                                                    000076600000
C:\Windows\SysWOW64\windows.storage.dll                                                            000072770000
C:\Windows\SysWOW64\wldp.dll                                                                       000072740000
C:\Windows\SysWOW64\shlwapi.dll                                                                    0000756e0000
C:\Windows\SysWOW64\msctf.dll                                                                      0000772f0000
C:\Windows\SysWOW64\oleaut32.dll                                                                   0000764e0000
C:\Windows\SysWOW64\TextShaping.dll                                                                000064130000
C:\Windows\SysWOW64\efswrt.dll                                                                     00007a970000
C:\Windows\SysWOW64\WinTypes.dll                                                                   0000656c0000
C:\Windows\SysWOW64\mpr.dll                                                                        000073680000
C:\Windows\SysWOW64\twinapi.appcore.dll                                                            0000641d0000
C:\Windows\SysWOW64\oleacc.dll                                                                     000072100000
C:\Windows\SysWOW64\TextInputFramework.dll                                                         000063e40000
C:\Windows\SysWOW64\CoreUIComponents.dll                                                           0000653a0000
C:\Windows\SysWOW64\CoreMessaging.dll                                                              000065620000
C:\Windows\SysWOW64\ws2_32.dll                                                                     000076df0000
C:\Windows\SysWOW64\ntmarta.dll                                                                    000075540000
C:\Windows\SysWOW64\ole32.dll                                                                      000076fc0000
C:\Windows\SysWOW64\duser.dll                                                                      00007c390000
C:\Windows\SysWOW64\xmllite.dll                                                                    000050c90000
C:\Windows\SysWOW64\atlthunk.dll                                                                   000055e40000
# END_IMAGE_ADDRESSES

# EMIT_STATIC_STATS
#
# $static-counts
#
#       opcode                count
#
*total                                                          0

# END_STATIC_STATS
# EMIT_GLOBAL_DYNAMIC_STATS   EMIT# 8
#
# $global-dynamic-counts
#
```

Рисунок 2.6 Файл SDE із запущеними прапорцями оптимізації

Інформація, яка зображена на малюнку вище, нам не потрібна. Тому був зроблений фільтр, який читає кожну лінію і вирішує, чи її потрібно ігнорувати чи ні.

У кінцевому результаті, були виявлені певні закономірності і на рисунку 2.7 зображений блок коду, який очищує всю лишню інформацію.

```
def clean_lines(lines):
    result = []
    bad_chars = ["*", "#", ":", "/", "["]
    for line in lines:
        if "*" in line:
            continue
        elif "#" in line:
            continue
        elif ":" in line:
            continue
        elif "/" in line:
            continue
        elif "[" in line:
            continue
        elif "\n" == line:
            continue
        elif " " == line[0]:
            continue
        else:
            chunks = re.split(' +', line)
            result.append((chunks[0], int(chunks[1].split("\n")[0])))
    return result
```

Рисунку 2.7 Функція відповідальна за очистку файлу

Таким чином, було залишено лише інформацію відносно виконаних операцій.

Наступний крок полягає в виділенні конкретних операцій, які будуть рахуватися. На рисунку 2.8 зображений блок коду, який відповідальний за збір інформації лише з конкретних операцій, які задаються користувачем.

```
def leave_operations(data, fields):
    return [(i[0], i[1]) for i in data if i[0] in fields]
```

Рисунок 2.8 Функція, яка залишає інформацію лише про конкретні операції

Як було попередньо зазначено лише певні операції для нас є важливими, а саме:

- RCL
- RCR
- ROL

- ROR
- SHR
- SHL
- SHRD
- SHLD
- XOR

Для візуалізації результатів була використана бібліотека matplotlib [91]. Matplotlib — це кросплатформна бібліотека візуалізації даних та графічного побудови графіків для Python та його числового розширення NumPy. Таким чином, він пропонує життєздатну альтернативу MATLAB з відкритим кодом. Розробники також можуть використовувати API (інтерфейси програмного забезпечення) matplotlib для вбудовування графіків у програми GUI.

Приклад гістограми створеної використовуючи matplotlib зображений на рисунку 2.9.

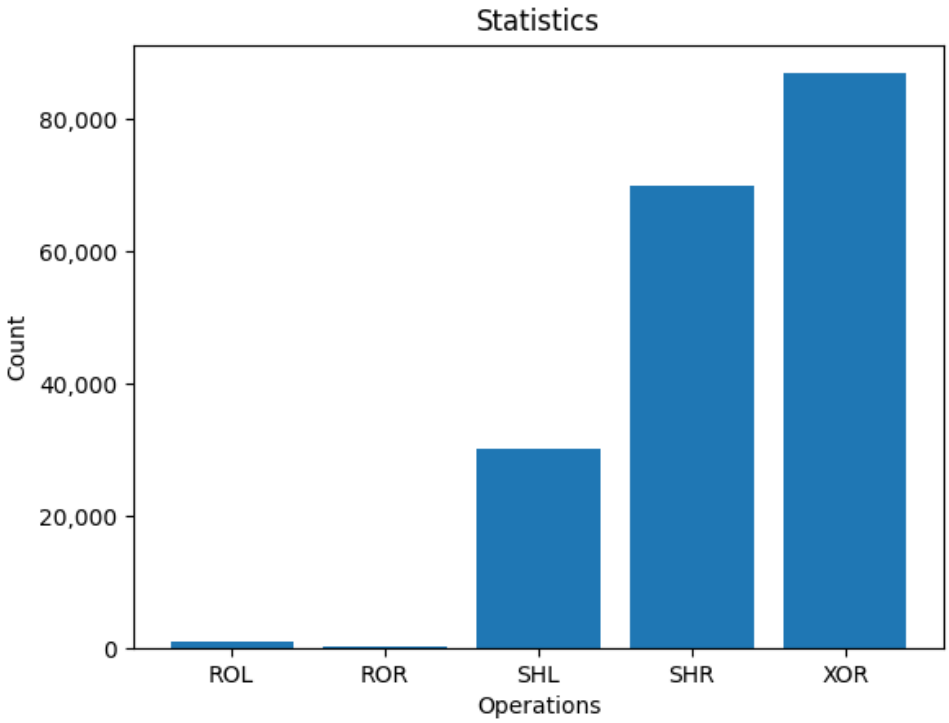


Рисунок 2.9 Приклад гістограми, яка створюється ПЗ

На рисунку 2.9 зображена кількість досліджуваних операцій з виконання певного додатку через у віртуальному середовищі SDE. За цю гістограму відповідальний код зображений на рисунку 2.10.

```
def create_chart(data, name="Table Name", x_name="X Name", y_name="Y Name"):
    x = [i[0] for i in data]
    y = [i[1] for i in data]

    plt.bar(x, y)
    plt.title(name)
    plt.xlabel(x_name)
    plt.ylabel(y_name)
    current_values = plt.gca().get_yticks()
    plt.gca().set_yticklabels(['{:,.0f}'.format(x) for x in current_values])
    plt.show()
```

Рисунку 2.10 Функція створення гістограми

Таким чином було розроблено скрипт, який дозволяє аналізувати додатки і виявляти закономірності, які пов’язані з операціями Shift, XOR, Rotate.

2.4 Аналіз поведінки програмного забезпечення різного призначення

Спочатку для аналізу буде задіяний криптомайнер XmRig. XMRig — це високопродуктивний з відкритим вихідним кодом, кросплатформний уніфікований майнер CPU/GPU. Він доступний для Windows, Linux, macOS і FreeBSD.

На рисунку 2.11 зображений консольний інтерфейс майнера XMrig.

Рисунок 2.11 Консольний інтерфейс XMRig

Після запуску майнера на 2 хв були отримані результати зображені на рисунку 2.12.

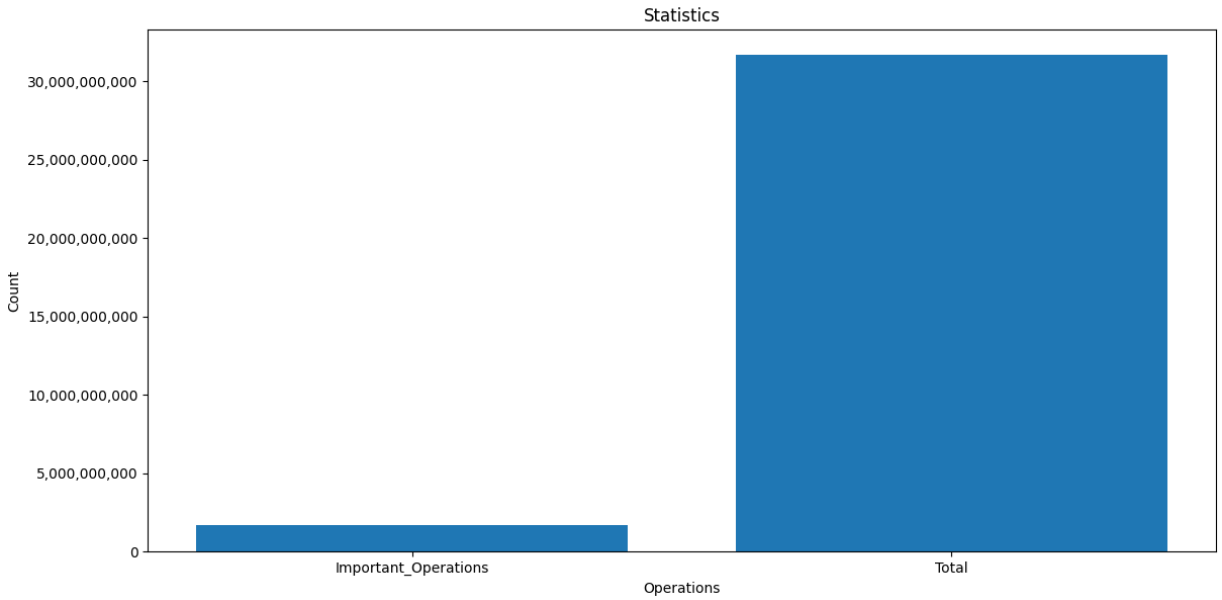


Рисунок 2.12 Відношення операцій Shift, XOR та Rotate до всіх операцій в криптомайнері XMRig

За період 2 хв більше 30 мільярдів операцій було здійснено. Біля 5% серед всіх операцій були виділені на операції Shift, Rotate та XOR.

Розподіл операцій зображений на рисунку 2.13.

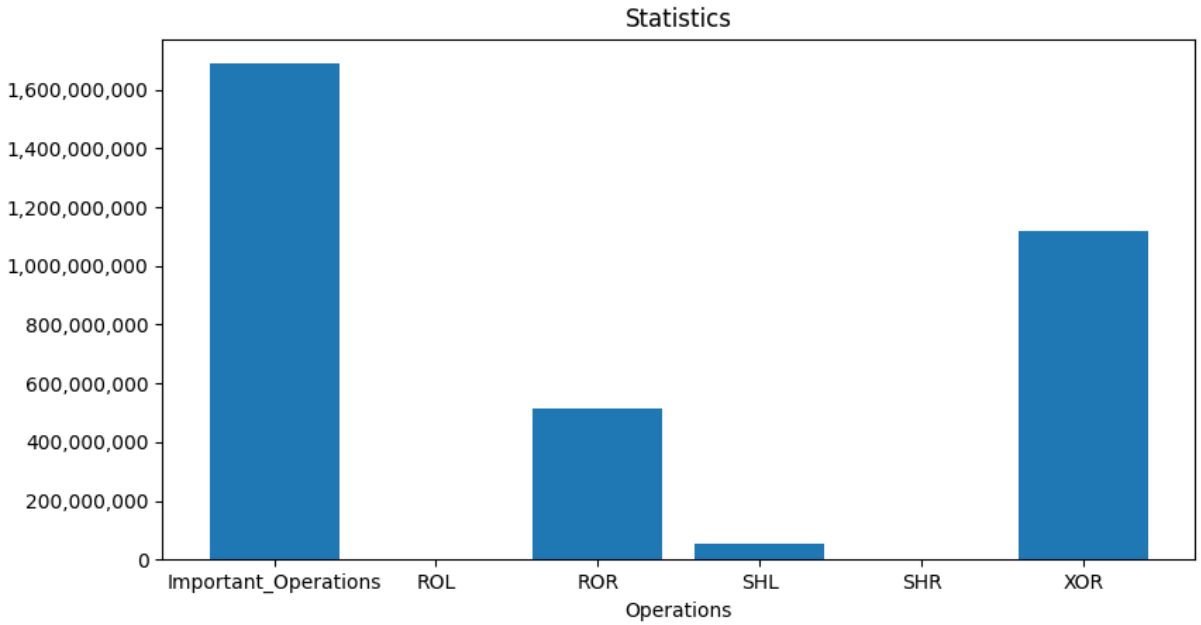


Рисунок 2.13 Розподіл операцій в криптомайнері XMRig

Таким чином, ми бачимо, що переважна кількість операцій – це операції Rotate (1.62% від всіх операцій) та XOR (3.53% від всіх операцій). Незважаючи на те, що операції Shift також можуть бути задіяними для виконання обчислень хеш-функцій в даному випадку ці операції не були настільки поширеними і становили всього 0.18% від всіх операцій.

Розглянемо виконання простої утиліти gedit. На рисунку 2.14 зображена кількість операцій за період виконання.

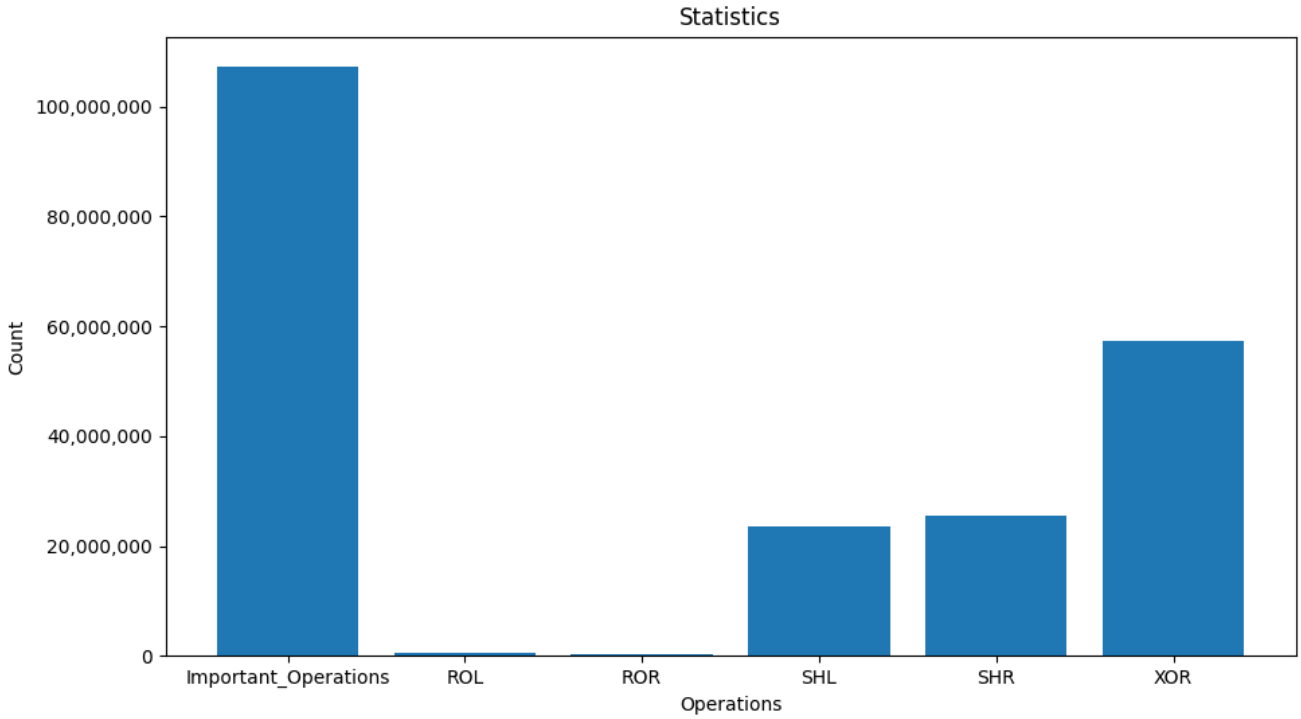


Рисунок 2.14 Розподіл операцій в утиліті gedit

Результати тут суттєво відрізняються від результатів майнера. Ми бачимо, що операцій Rotate (0.03% від всіх операцій) майже взагалі нема, але при цьому є велика кількість операцій Shift (1.77% від всіх операцій). Операцій XOR, як і у випадку з майнером, було найбільше, а саме 2.07% від всіх операцій. Потрібно також зазначити, що загальна кількість важливих операцій у відношенні до всіх операцій була суттєво менша і становила 3.7%, а не 5%, як у випадку з криптомайнером. Це зображено на рисунку 2.15.

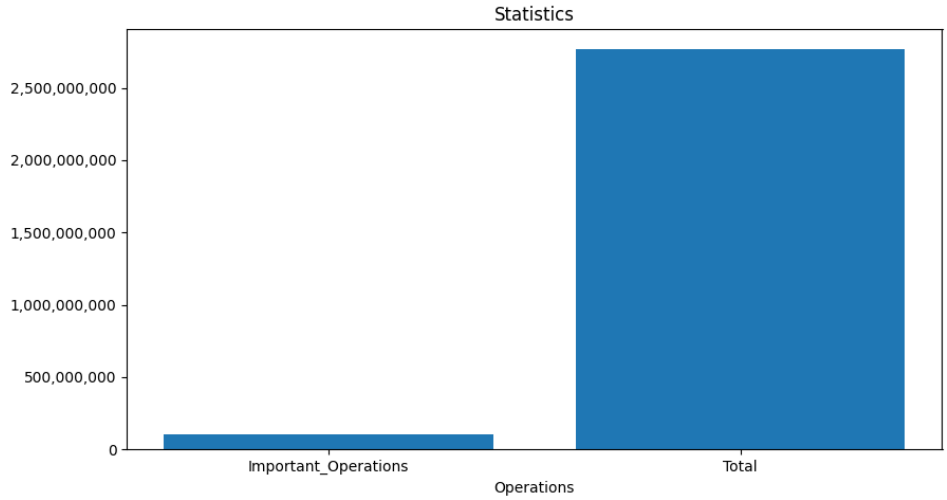
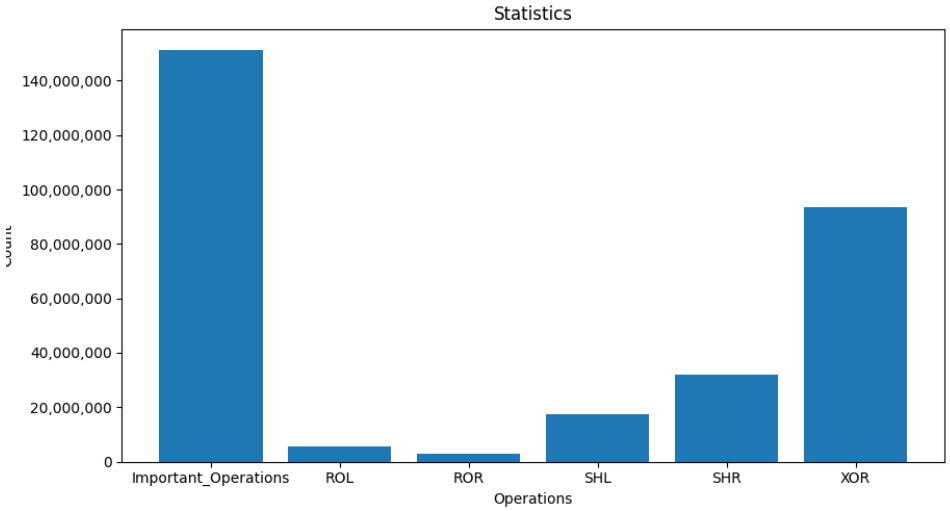


Рисунок 2.15 Відношення операцій Shift, XOR та Rotate до всіх операцій в утиліті gedit

Тепер розглянемо більш складне програмне забезпечення, яке може дати навантаження на кінцеву систему. Для аналізу буде використане ПЗ Audacity [92]. Audacity — це простий у використанні багатодоріжковий аудіоредактор і записувач для Windows, macOS, GNU/Linux та інших операційних систем.

На рисунку 2.16 зображені результати аналізу.



Рисунку 2.16 Розподіл операцій в ПЗ Audacity

З малюнку можна побачити, що всі операції в певній мірі задіяні, але при цьому їх все одно дуже мала кількість відносно всіх інших операцій. Rotate операції складають всього 0.2% від всіх операцій. Shift операції складають 1.14% від всіх

операцій, а XOR складають 2% від всіх операції. Сумарно важливі операції складають 3.5% від всіх операцій. Це зображено на рисунку 2.17.

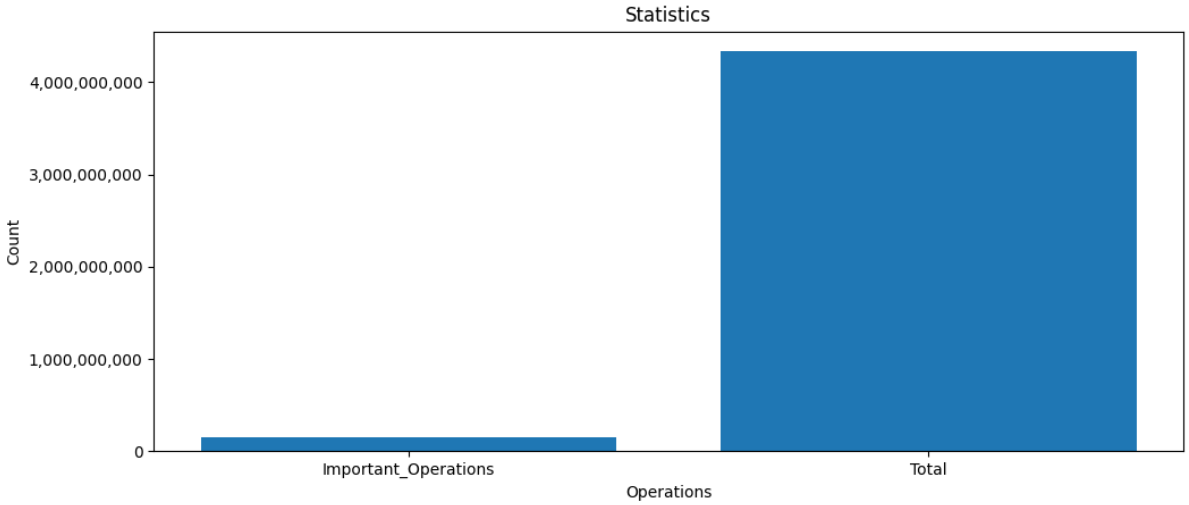


Рисунок 2.17 Відношення операцій Shift, XOR та Rotate до всіх операцій в ПЗ Audacity

Розглянемо ще один відомий майнер під назвою cpuminer. cpuminer — це багатопоточний високооптимізований процесорний майнер для Litecoin, Bitcoin та інших криптовалют. Наразі підтримуються алгоритми SHA-256d і scrypt(N, 1, 1). Він підтримує протокол майнінгу getblocktemplate, а також протокол майнінгу Stratum, і може використовуватися як для самостійного, так і для об’єднаного майнінгу. На рисунку 2.18 можна побачити результати виконання.

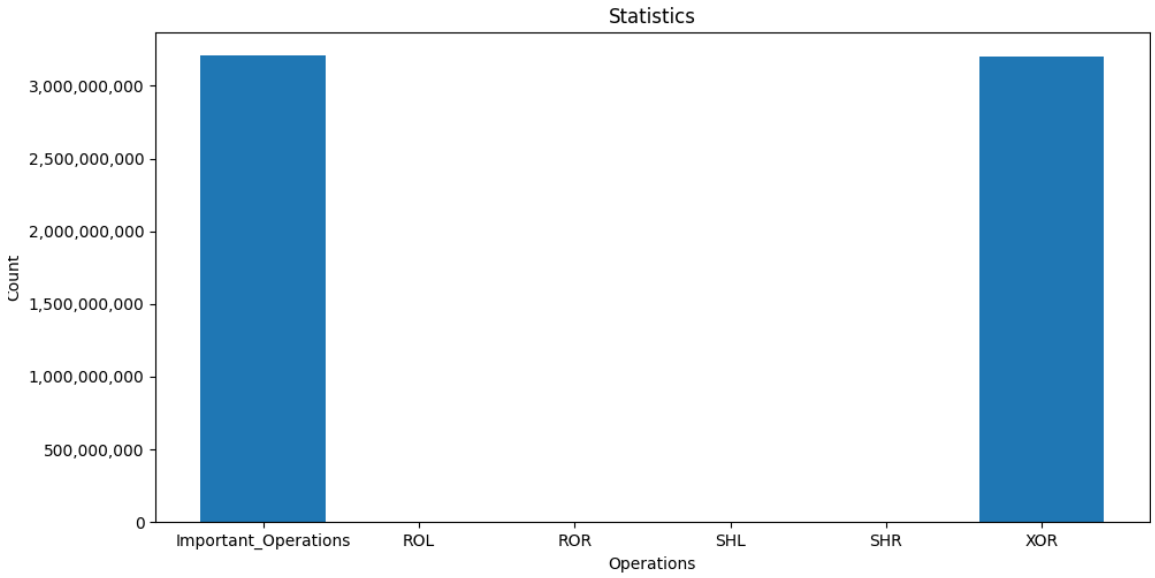


Рисунок 2.18 Розподіл операцій в cpuminer

Cpuminer суттєво відрізняється від інших отриманих результатів. Операції XOR домінують серед всіх операції і становлять аж 6.71% від всіх виконаних операцій. Операцій Shift + Rotate становили всього лише 0.01%. На рисунок 2.19 показане загальне відношення всіх операцій до важливих операцій.

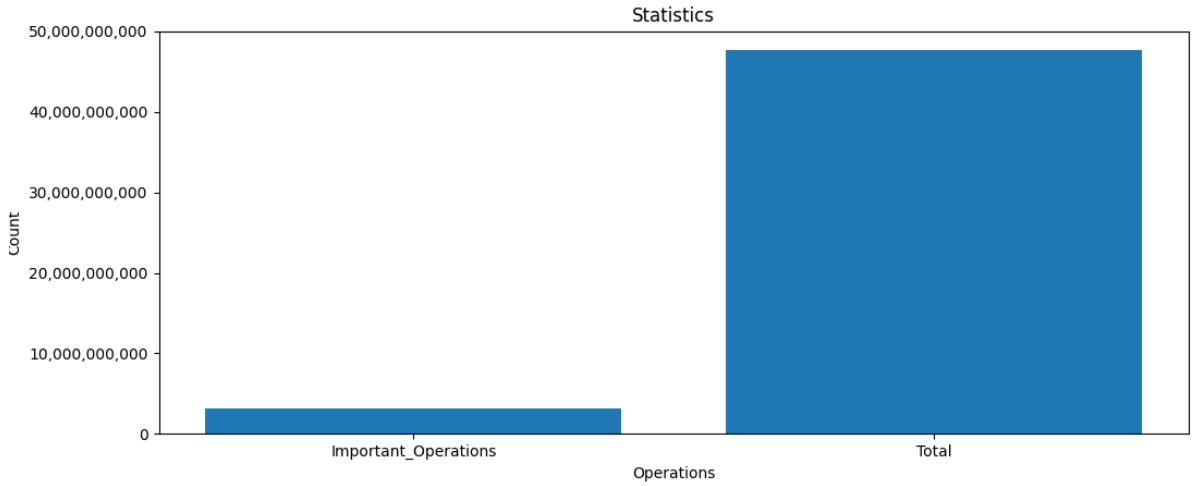


Рисунок 2.19 Відношення операцій Shift, XOR та Rotate до всіх операцій в ПЗ Audacity

2.5 Синтез моделі виявлення аномальної поведінки програмного забезпечення

У ході дослідження, було виявлено, що всі 3 типи операцій характерні для криптомайнингу. Виділити конкретні закономірності лише з певних операцій не вдалося, оскільки, певні криптомайнери мають багато операцій Rotate, а інші Shift або XOR.

Ключовим аспектом в аналізі є відношення всіх операцій, до важливих операцій. Серед чистих додатків виділенні операції в сумі не сягали більше 4% від всіх операцій, а в криптоманерах це відношення було завжди не менше 5%.

Тому для виявлення підозрілої активності пропонується наступна формула (2.1) :

$$K = \frac{\sum O_{imp}}{\sum O_{all}} * 100, \quad (2.1)$$

де K - відсоток операцій Shift, Rotate і XOR до всіх операцій;

O_{imp} - масив операцій Shift, Rotate і XOR;

O_{all} - масив всіх операцій.

З отриманих результатів зроблений висновок, що якщо коефіцієнт K перевищує 5%, то це є валідним індикатором для подальшого аналізу додатка, оскільки, це є аномалією для нешкідливих додатків, які не потребують великої кількості ресурсів кінцевої системи.

Висновки до розділу 2

Поставлені задачі аналізу аномальних процесів пов’язаних з діяльністю криптомайнерів, розробки методу виявлення криптомайнерів, створення програмного модуля методу виявлення криптомайнерів та синтезу моделі виявлення аномальної поведінки програмного забезпечення були успішно виконані.

Основою для аномалій слугують операції, які виконуються програмним забезпеченням. Була виявлена закономірність, що криптомайнерам характерна велика кількість операцій Rotate, Shift, XOR. Таким чином, метод будується на спостереженні за операціями програмного забезпечення.

Для спрощення аналізу поведінки додатків було розроблене програмне забезпечення на Python, яке використовує віртуальне середовище Intel Software Emulator і дозволяє швидко перевірити наявність аномальної поведінки.

У кінцевому результаті, була синтезована модель аномальної поведінки програмного забезпечення. У наступному розділі буде проаналізований мережевий трафік пов’язаний з криптомайнингом та використання цього трафіку для покращення моделі.

РОЗДІЛ 3. ПОКРАЩЕННЯ ЗАПРОПОНОВАНОГО РІШЕННЯ ВИКОРИСТОВУЮЧИ ІНФОРМАЦІЮ ПРО МЕРЕЖЕВУ АКТИВНІСТЬ

3.1 Аналіз мережевої активності криптомайнерів

У першому розділі було зазначено, що одним з параметрів, які досліджуються для виявлення діяльності криптомайнингу – це мережевий трафік. У другому розділі було запропоноване рішення, яка дозволяє виявити аномальну поведінку додатку базуючись на кількості операцій Rotate, Shift та XOR. Для покращення точності виявлення криптомайнингу, пропонується поєднання двох методів, тобто паралельно слідкувати, як за мережевим трафіком так і за операціями, які виконуються додатком.

Найпопулярнішим протоколом для комунікації клієнта з сервером пула майнера – Stratum [93]. Stratum — це протокол, що використовує звичайний TCP-сокет з корисним навантаженням, закодованим як повідомлення JSON-RPC. Таким чином, клієнт відкриває TCP-сокет і записує запити на сервер у вигляді повідомлень JSON, закінчених символом нового рядка «\n». Кожен рядок, отриманий клієнтом, знову є дійсним фрагментом JSON-RPC, що містить відповідь. Таке рішення має вагомі переваги: його дуже легко реалізувати і налагодити, оскільки обидві сторони комунікують у форматі, зрозумілому людині. Протокол на відміну від багатьох інших рішень легко розширюється без порушення зворотної сумісності. JSON широко підтримується на всіх платформах, і поточні майнери вже мають бібліотеки JSON, тому пакування та розпакування повідомлення дійсно просто та зручно.

Спілкування над Stratum починається з клієнта (майнера), що зв'язується з сервером пулу і виконує аутентифікація. Як тільки майнер успішно ввійде в систему пул майнінгу, він починає отримувати завдання через повідомлення про нові завдання, які мають загальний формат, показаний на рисунок 3.1.

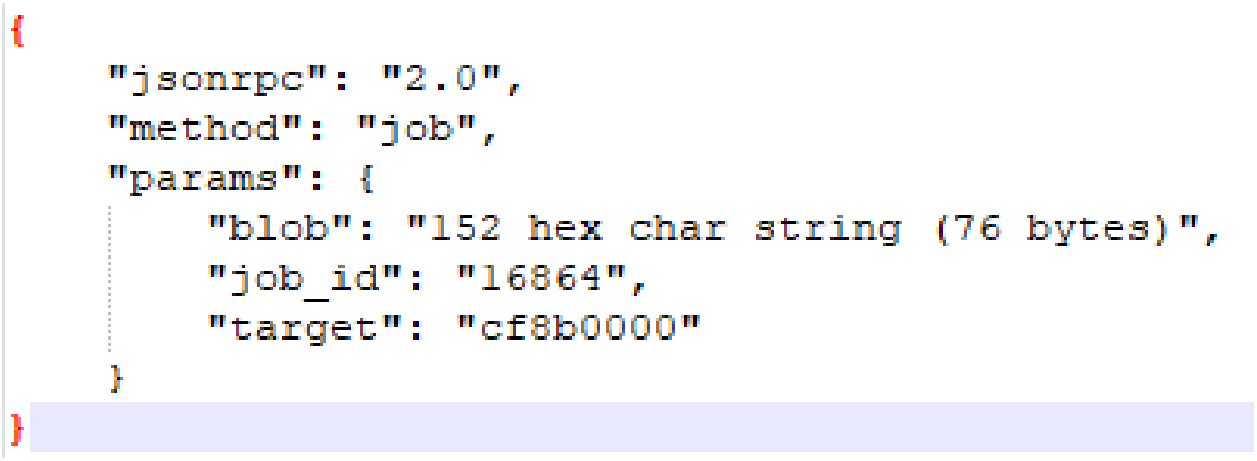


Рисунок 2 Stratum повідомлення про нову роботу

Основні властивості областей повідомлень «Нове завдання»:

- blob: шістнадцятковий рядок ASCII (76 байт/152 символи), що представляє вміст, який хешується майнерами для отримання монет Monero. Частина цього рядка доступна для редагування і представляє одноразовий номер, який може бути довільно встановлений майнерами;
- job_id: рядок, який використовується для узгодження завдань з їх відповідними результатами; він має змінну довжину, яка залежить від реалізації пулу;
- цільова: для того, щоб блок був прийнятий пулом, хеш його заголовка повинен бути нижчим або дорівнювати поточному цільовому об'єкту, таким чином зниження цілі ускладнює обчислення хешу.

Коли клієнт пулу майнингу отримує від сервера повідомлення «Нове завдання», він починає процес майнингу. Це включає в себе пошук одноразового значення, у якому значення, створене шляхом хешування нового BLOB-об’єкта за допомогою хеш-функції PoW, буде нижчим за цільове. Інформації, включеної в повідомлення «Нове завдання», достатньо для визначення проблеми.

Як тільки майнер знаходить рішення, він передає його результат до пулу у вигляді повідомлення про подання рішення. Приклад такого типу повідомлення можна побачити на рисунку 3.2. Він містить кілька полів, але 3 є релевантними для нашого аналізу. Ідентифікатор job_id відповідає виконаному завдання. Одноразовий код nonce (8 шістнадцяткових символів, що позначають 4-байтове значення) був

знайдений майнером і використаний для створення відповідного хешу. Як результат повертається хеш-значення заголовка блоку, що використовує знайдений одноразовий номер.

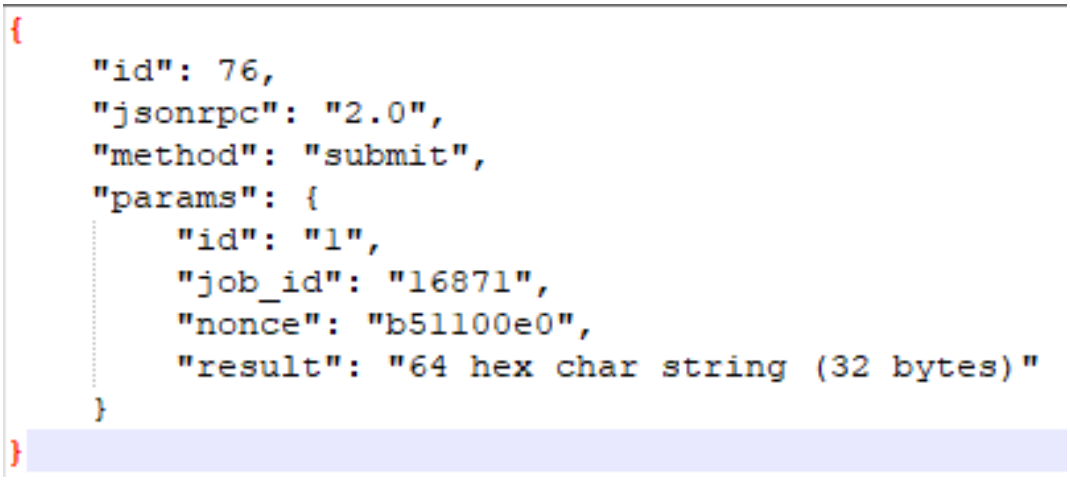


Рисунок 3.2 Повідомлення про подання рішення Stratum

Сервер пулу майнінгу отримує повідомлення про подання рішення. Після перевірки правильності вбудованого рішення він надсилає назад повідомлення про результати подання. Приклад показано на рисунку 3.3.

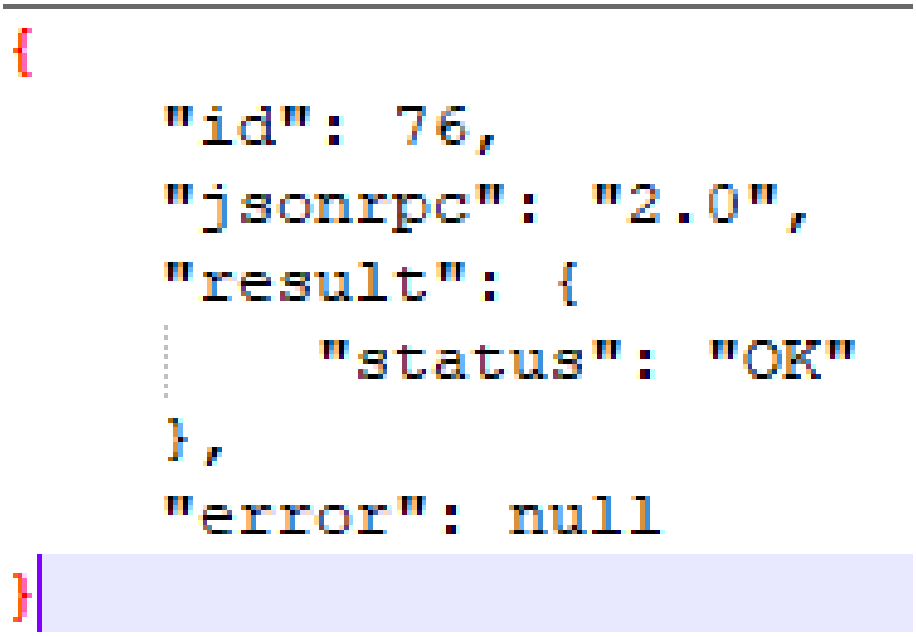


Рисунок 3.3 Повідомлення про результати подання Stratum.

Схема зв'язку між клієнтом пулу і сервером, зображено на рисунку 3.4. Цей мережевий трафік повторюється до тих пір, поки мережеве з'єднання не буде розірвано.

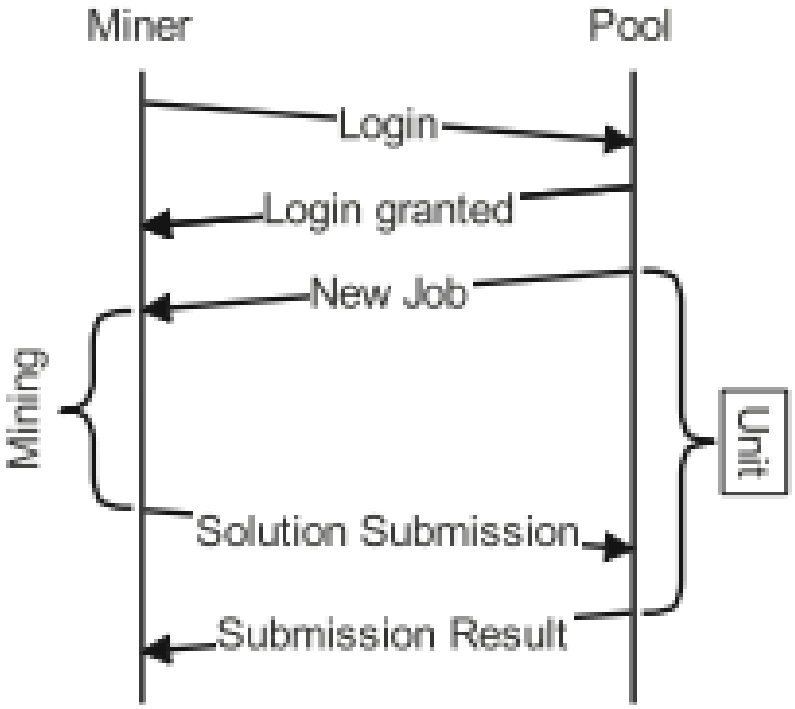


Рисунок 3 Типовий мережевий трафік Stratum

Майнери іноді можуть отримувати повідомлення «Нова робота» з пулу, коли вони вже працюють над завданням. Це відбувається з двох причин: або новий блок був здобутий, або з'явилися нові транзакції для поточного блоку. Тому співвідношення між кількістю робочих місць і кількістю поданих рішень не є однозначним.

У цілому аналізуючи протокол Stratum виявлено такі властивості:

- Розмір повідомлень Stratum, якими обмінюються сервер і клієнт, залишається майже незмінним.
- «Нове завдання» та «Подача рішення» — це найбільші повідомлення, надіслані пулом і майнером відповідно.
- Після входу в систему майнер зазвичай надсилає 2 типи повідомлень: «Подача рішення» та «Keep-Alive».
- Кількість підтверджень TCP (ACK), надісланих майнером, за винятком підтримки активності, дорівнює кількості повідомлень про результати подання та нового завдання, надісланих пулом. Кількість ACK, надісланих пулом, не дорівнює кількості подання рішення. Це відбувається тому, що пул часто використовує прапор ACK для подання рішення в пакеті результатів подання.

Але в протоколі Stratum існує ще трохи інший процес роботи з пул сервером, де використовуються інші методи. Цей процес зображений на рисунку 3.5.

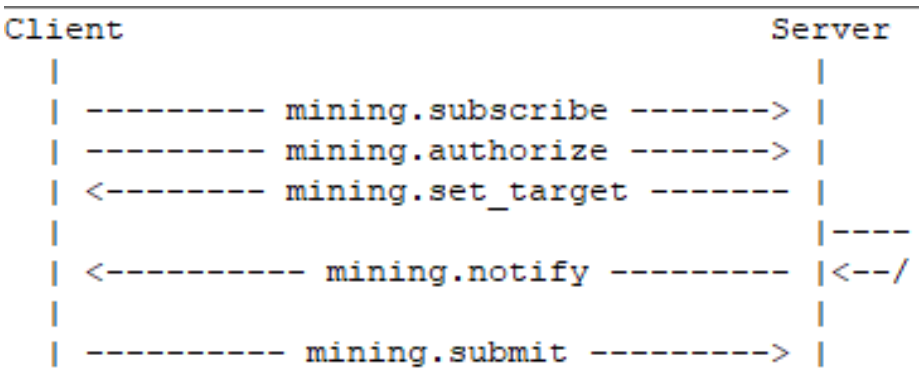


Рисунок 4 Приклад комунікацій між сервером використовуючи інші методи

Спочатку, щоб розпочати або відновити сеанс із сервером, клієнт повинен викликати метод підписки (mining.subscribe). Приклад запиту зображений на рисунку 3.6.

```
{
    "id": 1,
    "method": "mining.subscribe",
    "params": [
        "MyMiner/1.0.0",
        null,
        "my.pool.com",
        1234
    ]
}
```

Рисунок 3.6 Приклад запиту mining.subscribe

Перш ніж клієнт зможе передати рішення на сервер, він повинен авторизувати принаймні одного працівника. Це робиться за допомогою методу аутентифікації (mining.authorize). Приклад запиту зображений на малюнку.

```
{
    "id": 2,
    "method": "mining.authorize",
    "params": [
        "WORKER_NAME",
        "WORKER_PASSWORD"
    ]
}
```

Рисунок 3.7 Приклад запиту mining.authorize

Час від часу, цільова складність для блоку може змінитися, і сервер повинен мати можливість повідомити про це клієнтів. Для цього використовується метод встановлення цілі (mining.set_target). Приклад запиту зображений на рисунку 3.8.

```
{
    "id": 1,
    "method": "mining.set_target",
    "params": [
        "FFFF0000000000000000000000000000"
    ]
}
```

Рисунок 3.8 Приклад запиту mining.set_target

Виклик сповіщення використовується для поширення інформації про стосовно нового блоку, який потрібно вирахувати. Ця операція здійснюється пул сервером. Приклад запиту зображений на рисунку 3.9.

```
{
    "id": 1,
    "method": "mining.notify",
    "params": [
        "d70fd222",
        "abad8f99f3918bf903c6a909d9bbc0fdfa5a2f4b9cb1196175ec825c66101126c",
        true
    ]
}
```

Рисунок 3.9 Приклад запиту mining.set_target

За допомогою методу submit працівник може подавати рішення для головоломки майнінгу, що принесе винагороду клієнту.

Тобто Stratum – це справді легкий для сервера протокол, який дозволяє тримати клієнтів з оновленою інформацією і мережеві пакети, які пов’язані з даним протокол є чітким знаком того, що в мережі задіяна діяльність криптомайнерів.

Наприклад, коли запущений майнер cpuminer, який був розглянутий у другому розділі, то в логах можна бачити, кожен раз, як новий блок був знайдений. Це зображено на рисунку 3.10.

```
[2022-05-04 09:34:31] Starting Stratum on stratum+tcp://us2.litecoinpool.org:3333
[2022-05-04 09:34:32] thread 0: 4104 hashes, 24.01 khash/s
[2022-05-04 09:35:32] thread 0: 1440408 hashes, 23.85 khash/s
[2022-05-04 09:35:36] Stratum detected new block
```

Рисунку 3.10 Приклад запиту mining.set_target

Під час аналізу трафіку було задіяне ПЗ Wireshark [94] для того, щоб зрозуміти кількість запитів, які виконуються майнером за визначений період часу. Протягом 20 хв майнер виконав 121 запитів, тобто кожні 10 секунд відбувається синхронізація із сервером. Приклад відправленого пакету зображений на рисунку 3.11.

```
fa f0 b3 91 00 00 7b 22  69 64 22 3a 6e 75 6c 6c   ......{" id":null
2c 22 6d 65 74 68 6f 64  22 3a 22 6d 69 6e 69 6e   ,"method ":"minin
67 2e 6e 6f 74 69 66 79  22 2c 22 70 61 72 61 6d   g.notify ","param
73 22 3a 5b 22 61 36 33  66 22 2c 22 62 39 65 37   s":["a63 f","b9e7
31 65 39 63 63 33 32 37  30 62 30 37 66 61 33 39   1e9cc327 0b07fa39
64 31 34 34 32 66 37 63  66 31 38 34 36 36 63 66   d1442f7c f18466cf
30 64 66 33 39 34 37 66  32 65 30 65 32 37 39 38   0df3947f 2e0e2798
61 30 32 64 61 34 64 66  65 64 62 61 22 2c 22 30   a02da4df edba","0
31 30 30 30 30 30 30 30  31 30 30 30 30 30 30 30   10000000 10000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30   00000000 00000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30   00000000 00000000
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30   00000000 00000000
30 30 30 30 30 30 30 30  30 66 66 66 66 66 66 66   00000000 0fffffff
66 34 34 30 33 34 63 37  31 32 32 30 34 36 32 37   f44034c7 12204627
32 61 66 36 61 32 63 66  61 62 65 36 64 36 64 63   2af6a2cf abe6d6dc
65 65 35 66 36 61 37 63  63 65 63 36 63 61 65 61   ee5f6a7c cec6caea
37 37 31 30 38 39 31 64  39 65 36 61 36 61 63 65   7710891d 9e6a6ace
36 34 31 37 66 61 34 30  36 61 66 63 30 30 36 35   6417fa40 6afc0065
65 32 30 36 61 39 61 39  36 34 30 34 65 66 38 34   e206a9a9 6404ef84
30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30   00000000 00000000
34 32 66 34 63 35 30 32  66 30 38 22 2c 22 66 66   42f4c502 f08","ff
66 66 66 66 66 66 30 32  36 33 32 61 38 62 34 61   ffffff02 632a8b4a
30 30 30 30 30 30 30 30  31 39 37 36 61 39 31 34   00000000 1976a914
35 37 37 35 37 65 64 39  65 32 66 61 37 64 38 36   57757ed9 e2fa7d86
36 33 36 62 64 64 38 30  36 36 30 37 38 65 32 32   636bdd80 66078e22
63 66 61 31 39 32 34 34  38 38 61 63 30 30 30 30   cfa19244 88ac0000
30 30 30 30 30 30 30 30  30 30 30 30 32 36 36 61   00000000 0000266a
```

Рисунок 3.11 Приклад перехопленого запиту через Wireshark

Таким чином, можна слухати за трафіком, який відбувається в мережі, але слідкувати чисто за трафіком протоколу Stratum недостатньо, оскільки, Stratum трафік не шифрується, але обмін інформацією з пул серверами можна здійснювати й через протокол SSL.

Наприклад, криптомайнер XMRig, який був також досліджений в другому розділі здійснює обмін з серверами зашифровано. За 10 хв були здійснено 50 запитів

синхронізації з головним сервером. Приклад пакету з трафіку XMRig зображений на рисунку 3.12.

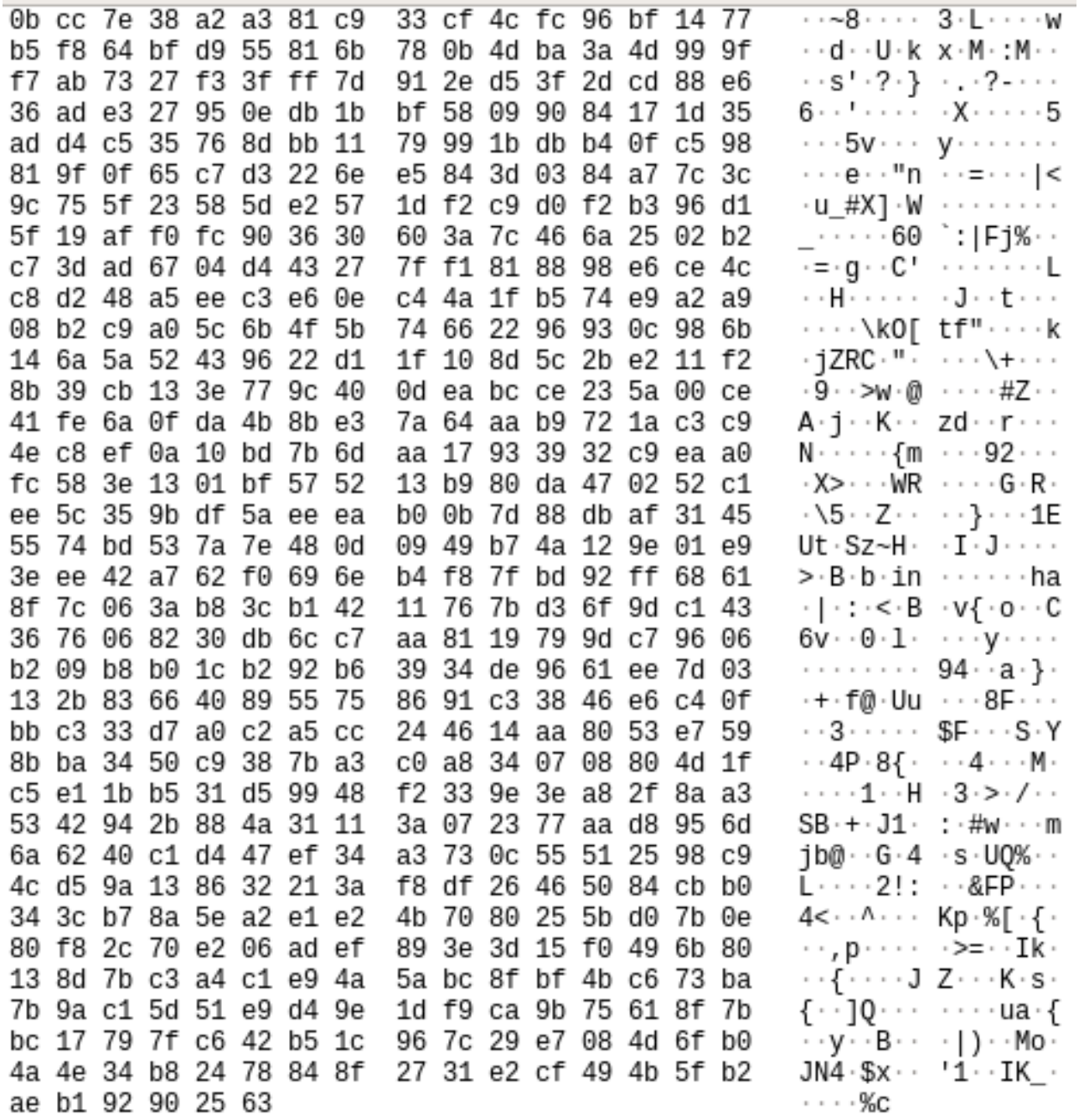


Рисунок 3.12 Приклад пакету зашифрованого мережевого трафіку

Тобто з даних, які знаходяться в пакеті неможливо буде виявити, що відбувається діяльність криптомайнингу, але все одно потрібно враховувати, що система буде постійно виконувати запити. Більше того, якщо дивитися на розмір пакету, то можна побачити певну закономірність. Переважно розмір пакетів, які були надіслані криптомайнером сягали за розміром 54 байти.

Під час дослідження XMRig був запущений на 25 хв, за цей період часу було здійснено 194 запити. Серед них 67 запитів були 54 байтів у розмірі, що становить майже 35% від всіх здійснених запитів. На рисунку 3.13 зображені найпоширеніші розміри пакетів для криптомайнера XMRig за 25 хв виконання.

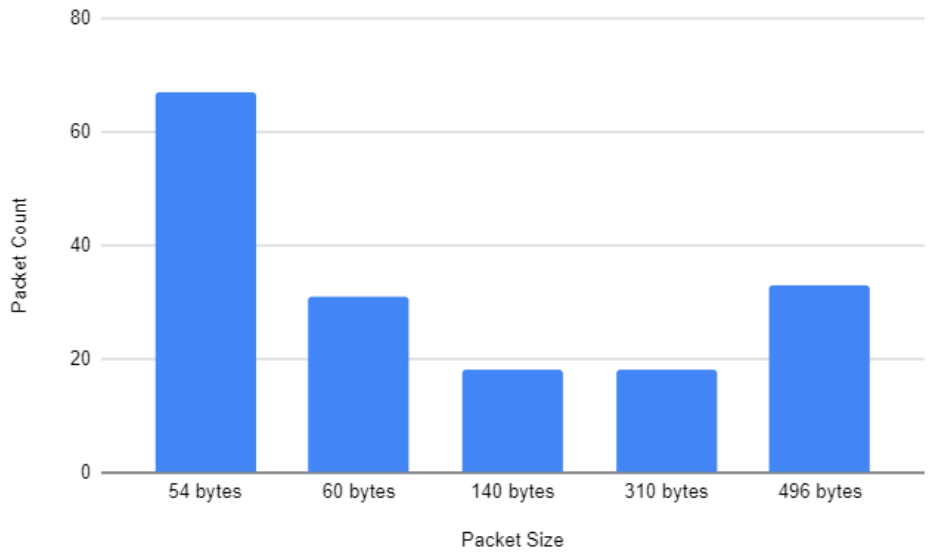


Рисунок 3.13 Найбільш поширені розміри пакетів криптомайнера XMRig

Криптомайнер cpuminer, за такий самий період часу здійснив 150 запитів. Серед них 74 пакетів становили 54 байти в розмірі, що становить майже 50% всіх пакетів. На рисунку 3.14 зображені найпоширеніші розміри пакетів для криптомайнера cpuminer за 25 хв виконання.

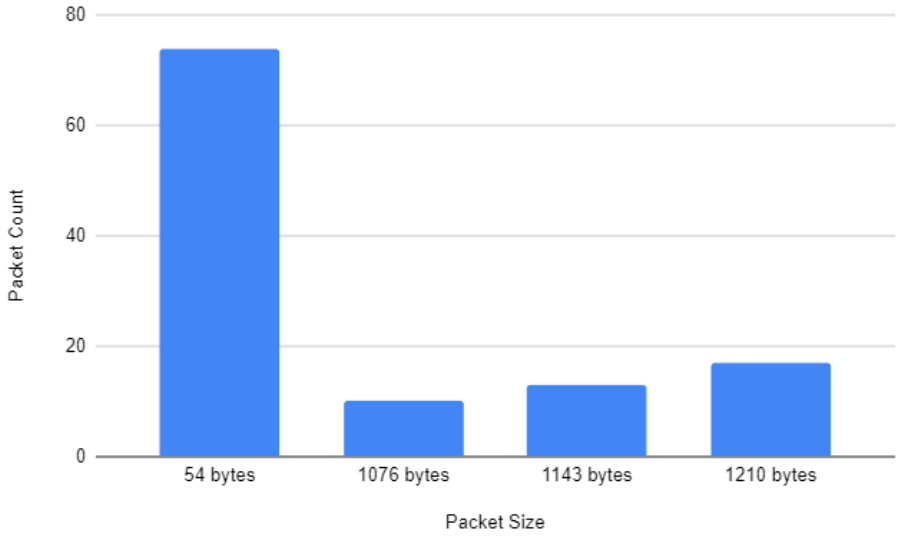


Рисунок 3.14 Найбільш поширені розміри пакетів криптомайнера cpuminer

Звідси, можна робити гіпотезу, що якщо в мережі існує постійний потік пакетів розміром в 54 байти, то це також можна рахувати індикатором для виявлення діяльності криптомайнера.

Розглядаючи періодичність запитів з пакетами обсягом в 54 байти, то для cpuminer можна побачити на рисунку 3.15.

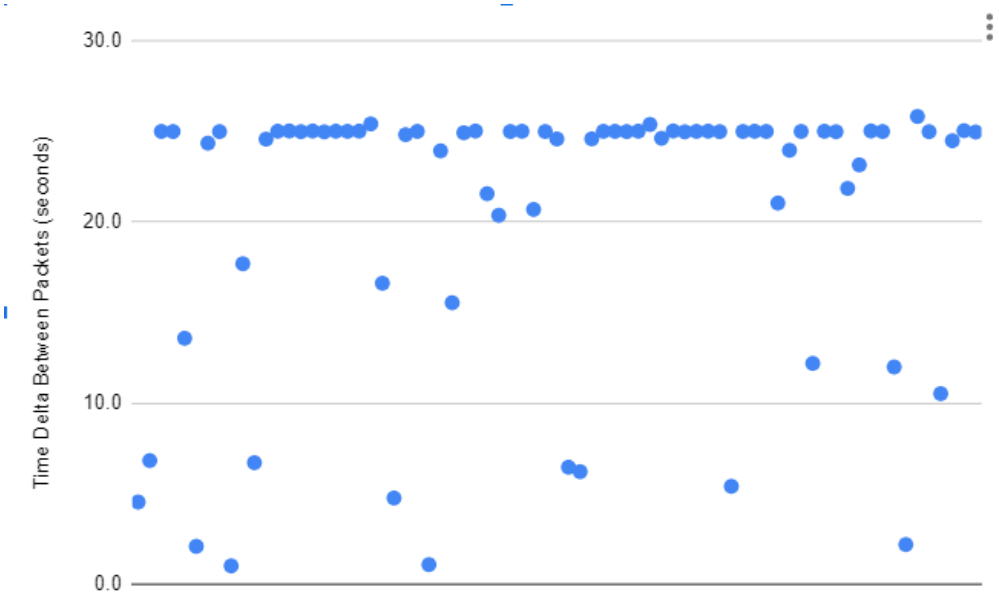


Рисунок 3.15 Періодичність запитів пакетів розміром 54 байти для cpuminer

З отриманих результатів можна побачити, що такі запити відбуваються переважно кожні 25 секунд для даного типу криптомайнера. На рисунку 3.16 зображена інформація відносно XMRig.

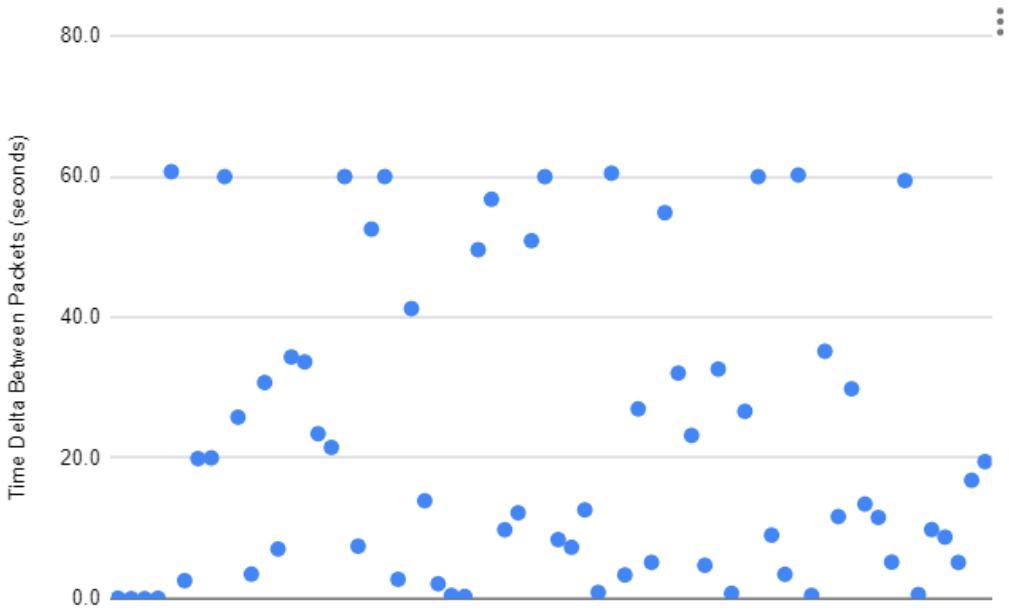


Рисунок 3.16 Періодичність запитів пакетів розміром 54 байти для XMRig

З результатів XMRig, ми бачимо, що не існує якоїсь сталої закономірності відносно періодичності запитів. Для cpuminer час між запитами не становив більше 25 секунд, а для XMRig становить інколи 60 секунд, тому тут не можна створити правило для виявлення аномалій.

3.2 Покращення методу виявлення аномалій поєднуючи мережеву активність та методу виявлення на основі операцій

З результатів отриманих в попередньому підрозділі зроміло, що криптомайнери оперують за певними протоколами і ці протоколи чітко описані. Більш того, мережевий трафік генерується постійно, оскільки майнери повинні бути синхронізовані з пул сервером. Таким чином можна виділити певні аспекті мережового трафіку, які можуть слугувати індикаторами для виявлення аномалій пов’язаних з криптомайнингом:

1. Якщо в мережі виявлено обмін даними використовуючи протокол Stratum, то це гарант того, що в на кінцевій системі відбувається криптомайнинг.
2. Якщо в мережі виявлено постійний трафік, де пакети я розміром в 54 байти, то це також індикатор аномалії і кінцева система може бути враженою на криптомайнинг.

З першим пунктом все очевидно, оскільки, протокол Stratum використовується лише в середовищах криптомайнерів. Тобто, будь-яка активність, яка проводиться через цей протокол є чітким індикатором криптомайнингу.

Другий пункт не є настільки надійним, оскільки, пакет на 54 байта – це мінімальний за обсягом ACK пакет, який містить інформацію про здійснене з’єднання, але при цьому це все одно важливий індикатор, бо для синхронізації постійно буде здійснюватися з’єднання з пул сервером і спеціаліст по кібербезпеці може на основі цього факту робити вже певні припущення стосовно різноманітного додатку.

Поєднуючи ці два правила з дослідженням кількості операцій Shift, Rotate та XOR отримується нове рішення, яке дозволяє виявляти аномалії пов’язані з криптомайнингом.

Обидва рішення грають важливу роль, оскільки, теоретично можуть бути різні сценарії поведінки додатку, яке буде складніше класифікувати, як діяльність криптомайнера:

1. Додаток має велику кількість операцій XOR, Rotate, Shift, але при цьому мережева активність зашифрована

2. Додаток не має велику кількість операцій XOR, Rotate, Shift, але при цьому існує постійна мережева активність протоколу Stratum

3. Додаток не має велику кількість операцій XOR, Rotate, Shift, не існує мережева активність протоколу Stratum, але постійно відбувається певна зашифрована мережева активність.

Тому для повноцінного рішення критично враховувати як аспекти виконаних операцій, так і мережевої активності.

Висновки до розділу 3

Поставлені задачі дослідження у вигляді аналізу мережевого трафіку криптомайнерів та покращення методу виявлення аномальної поведінки поєднуючи інформацію про мережевий трафік та синтезовану модель були успішно виконані.

Таким чином, були виділені 2 ключових аспекти серед мережевої активності криптомайнерів – використання протоколу Stratum та постійні з’єднання з пул сервером для синхронізації даних. Оскільки, протокол Stratum не обмінює інформацію в зашифрованому форматі, то будь-яка наявність пакетів пов’язаних з цим протокол слугують важливим індикатором для виявлення процесів криптомайнингу.

Поєднуючи метод виявлення запропонований у розділі 2 з мережевою активністю описаною в розділі 3 – отримується більш надійне рішення, яке може виявити несанкціоновану діяльність криптомайнерів у ширшому спектрі.

ВИСНОВКИ

У науковій роботі була досліджена поведінка криптомайнерів і на основі отриманої інформації був запропонований метод виявлення криптомайнингу.

Відносно завдання дослідження було виконано збір та аналіз даних для моделювання, розроблені алгоритми попередньої обробки даних, синтезовані моделі нейронних мереж та виконано їх навчання. За результатами дослідження проведений аналіз адекватності синтезованих моделей.

Підсумовуючи результати дослідження, слід зазначити, що запропонований метод може виявляти активність криптомайнерів використовуючи дані отримані про операції XOR, Rotate, Shift та у комбінації з аналізом мережевої активності може в перспективі слугувати одним із модулів детектування для SIEM-систем.

Подальшою роботою в даній сфері є розробка інтелектуальних моделей для виявлення більшого спектру атак криптоджекінгу та централізація моделей в єдиному програмному забезпечені та візуалізація їх роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. / Nakamoto. – 2008.
2. Kurt A. Lnbot: A covert hybrid botnet on bitcoin lightning network for fun and profit / Kurt. – 2020.
3. Bhargavan K. Formal verification of smart contracts: Short paper / Bhargavan. – 2016.
4. N V. Cryptonote v 2.0 [Електронний ресурс] / Van Saberhagen N. – 2013. – Режим доступу до ресурсу: https://bytecoin.org/old/whitepaper.pdf.
5. Olenick D. Miner into third party zoom [Електронний ресурс] / Olenick – Режим доступу до ресурсу: https://www.trendmicro.com/en_us/research/20/d/zoomed-in-a-look-into-a-coinminer-bundled-with-zoom-installer.html.
6. Santos M. utorrent update smuggles shady cryptocurrency miner into your computer [Електронний ресурс] / Santos – Режим доступу до ресурсу: https://99bitcoins.com/utorrent-update-cryptocurrency-miner/.
7. Cve-2019-2725 exploited, used to deliver monero miner [Електронний ресурс] – Режим доступу до ресурсу: https://www.trendmicro.com/en ca/research/19/f/cve-2019-2725-exploited-and-certificate-files-used-for obfuscation-to-deliver-monero-miner.html.
8. Cryptojacking malware hid into emails [Електронний ресурс] – Режим доступу до ресурсу: https: //www.mailguard.com.au/blog/brandjacking-malware-hiding.
9. A vulnerability used to deliver cryptojacking malware [Електронний ресурс] – Режим доступу до ресурсу: https://www.fireeye.com/blog/threat-research/2018/02/cve-2017-10271-used-to-deliver-cryptom iners.html.
10. Lee S. Cybercriminal minds: An investigative study of cryptocurrency abuses in the dark web / Lee. – 2019.
11. Goriacheva A. Anonymization technologies of cryptocurrency transactions as money laundering instrument / Goriacheva. – 2018.
12. Bonneau J. Mixcoin: Anonymity for bitcoin with accountable mixes / Bonneau. – 2014.
13. Huang D. Tracking ransomware end-to-end / Huang. – 2018.

14. Paquet-Clouston M. Ransomware payments in the bitcoin ecosystem / Paquet-Clouston. – 2019
15. Conti M. On the economic significance of ransomware campaigns: A bitcoin transactions perspective / Conti. - 2018
16. Kharraz A. Cutting the gordian knot: A look under the hood of ransomware attacks / Kharraz. – 2015
17. Coinhive snapshots that taken from webarchive [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org/web/20181101000000*/coinhive.com
18. Monero price chart [Електронний ресурс] – Режим доступу до ресурсу: https://coinmarketcap.com/currencies/monero/
19. Ukrainian man faces up to 6 years in jail for cryptojacking on his own websites [Електронний ресурс] – Режим доступу до ресурсу: https://cointelegraph.com/news/ukrainian-man-faces-up-to-6-years-in-jail-for-cryptojacking-onhis-own-websites
20. Ruth J. Digging into browser-based crypto mining / Ruth. - 2018
21. Bijmans H. Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale / Bijmans. – 2019
22. Hackers mined a fortune from indian websites [Електронний ресурс] – Режим доступу до ресурсу: https://economictimes.indiatimes.com/small-biz/startups/newsbuzz/hackers-mined-a-fortune-from-indian websites/articleshow/65836088.cms
23. Miners in youtube ads [Електронний ресурс] – Режим доступу до ресурсу: https://arstechnica.com/information-technology/2018/01/now-even-youtube-serves-ads-with-cpu-draining-cryptocurrency-miners/
24. Google tag manager exploited [Електронний ресурс] – Режим доступу до ресурсу: https://www.theregister.com/2017/11/22/cryptojackers_google_tag_manager_coin_hive/
25. Google bans all cryptomining extensions from the chrome store [Електронний ресурс] – Режим доступу до ресурсу: https://www.wired.com/story/google-bansall-cryptomining-extensions-from-the-chrome-store/

26. IDA PRO [Електронний ресурс] – Режим доступу до ресурсу: https://hex-rays.com/ida-pro/
27. A 'fortnite' cheat maker duped players into downloading a bitcoin miner [Електронний ресурс] – Режим доступу до ресурсу: https://www.vice.com/en/article/8x598p/a-fortnite-cheat-maker-duped-players-into-downloading-a-bitcoin-miner-epic-games-sued
28. Brilliant but evil: Gaming company fined $1 million for secretly using players' computers to mine bitcoin [Електронний ресурс] – Режим доступу до ресурсу: https://www.forbes.com/sites/kashmirhill/2013/11/19/brilliant-but-evil-gaming-company-turned-players-computers-into-unwitting-bitcoin-mining-slaves/?sh=11f7e958570b
29. Bijmans H. Just the tip of the iceberg: Internet-scale exploitation of routers for cryptojacking / Bijmans. – 2019
30. New research: Crypto-mining drives almost 90% of all remote code execution attacks [Електронний ресурс] – Режим доступу до ресурсу: https://www.imperva.com/blog/new-research-crypto-mining-drives-almost-90-remote-code-execution-attacks/
31. Social engineering attacks for cryptojacking [Електронний ресурс] – Режим доступу до ресурсу: https://www.bankinfosecurity.com/cryptocurrency-theft-hackersrepurpose-old-tricks-a-10685
32. A vulnerability used to deliver cryptojacking malware [Електронний ресурс] – Режим доступу до ресурсу: https://www.fireeye.com/blog/threat-research/2018/02/cve-2017-10271-used-to-deliver-cryptominers.html
33. Breaking down a two-year run of vivin's cryptominers [Електронний ресурс] – Режим доступу до ресурсу: https://blog.talosintelligence.com/2020/01/vivin-cryptomining-campaigns.html
34. An italian bank's server was hijacked to mine bitcoin [Електронний ресурс] – Режим доступу до ресурсу: https://qz.com/1024930/bitcoin-malware-an-italianbanks-server-was-hijacked-to-mine-bitcoin-says-darktrace/

35. Uk government plugin based mining [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitaltrends.com/computing/government-websites-plugin-coinhive-monero-miner/
36. Tesla hackers hacked aws cloud services to mine monero [Електронний ресурс] – Режим доступу до ресурсу: https://fortune.com/2018/02/20/tesla-hack-amazon-cloud-cryptocurrency-mining/
37. Coinminer, ddos bot attack docker daemon ports [Електронний ресурс] – Режим доступу до ресурсу: https://www.trendmicro.com/vinfo/hk-en/security/news/virtualization-and-cloud/coinminer-ddos-bot-attack-docker-daemon-ports
38. Cve-2019-2725 exploited, used to deliver monero miner [Електронний ресурс] – Режим доступу до ресурсу: https://www.trendmicro.com/en_ca/research/19/f/cve-2019-2725-exploited-and-certificate-files-used-for-obfuscation-to-deliver-monero-miner.html
39. Cloud-based cryptojacking article [Електронний ресурс] – Режим доступу до ресурсу: https://www.trendmicro.com/en_ca/research/19/f/cve-2019-2725-exploited-and-certificate-files-used-for-obfuscation-to-deliver-monero-miner.html
40. Watchdog: Exposing a cryptojacking campaign that's operated for two years [Електронний ресурс] – Режим доступу до ресурсу: https://unit42.paloaltonetworks.com/watchdog-cryptojacking/
41. Detect large-scale cryptocurrency mining attack against kubernetes clusters [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/blog/detect-largescale-cryptocurrency-mining-attack-against-kubernetes-clusters/
42. Jayasinghe K. A survey of attack instances of cryptojacking targeting cloud infrastructure / Jayasinghe. – 2020
43. Estimated iot device count by 2025 [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/1101442/iot-number-of-connecteddevices-worldwide/
44. What is the mirai botnet? [Електронний ресурс] – Режим доступу до ресурсу: https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/

45. Bertino E. Botnets and internet of things security / Bertino. - 2017
46. Ahmad A. A new cryptojacking malware classifier model based on dendritic cell algorithm / Ahmad. – 2019
47. Dashevskyi S. Dissecting android cryptocurrency miners / Dashevskyi. – 2020
48. Nocoin: Block lists to prevent javascript miners [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/hoshsadiq/adblock-nocoin-list
49. Minerblock: An efficient browser extension to block browserbased cryptocurrency miners all over the web [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/xd4rker/MinerBlock/blob/master/assets/filters.txt
50. Acar A. An analysis of malware trends in enterprise networks / Acar. – 2019
51. Willems C. Toward automated dynamic malware analysis using cwsandbox / Williems. – 2007
52. Javascript library callig instructions from the official documents [Електронний ресурс] – Режим доступу до ресурсу: https://javascript.info/modules-intro
53. Pastrana S. A first look at the cryptomining malware ecosystem: A decade of unrestricted wealth / Pastrana. – 2019
54. Posegga J. Rapid: Resource and api-based detection against in-browser miners / Posegga. – 2018
55. Kharraz A. Outguard: Detecting in-browser covert cryptocurrency mining in the wild / Kharraz. – 2019
56. Browser-based deep behavioral detection of web cryptomining with coinspy / 2020
57. Hong G. How you get shot in the back: A systematical study about cryptojacking in the real world / Hong. – 2018
58. Musch M. Thieves in the browser: Web-based cryptojacking in the wild / Munsch. – 2019
59. Wang W. Seismic:Secure in-lined script monitors for interrupting cryptojacks / Wang. – 2018
60. Coinblockerlists [Електронний ресурс] – Режим доступу до ресурсу: https://zerodot1.gitlab.io/CoinBlockerListsWeb/

61. Darabian H. Detecting cryptomining malware: a deep learning approach for static and dynamic analysis / Darabian. – 2020
62. The cuckoo sandbox [Електронний ресурс] – Режим доступу до ресурсу: https://www.cuckoosandbox.org
63. Barlet-Ros P. Detecting cryptocurrency miners with netflow/ipfix network measurements / Barlet-Ros. – 2019
64. Ning R. Capjack: Capture in-browser crypto-jacking by deep capsule network through behavioral analysis / Ning. – 2019
65. Lopez M. Minecap: super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking / Lopez. 2020
66. Gangwal A. Detecting covert cryptomining using hpc / Gangwal. – 2020
67. Tahir R. The browsers strike back: countering cryptojacking and parasitic miners on the web / Tahir. – 2019
68. VirusTotal [Електронний ресурс] – Режим доступу до ресурсу: https://www.virustotal.com/gui/home/upload
69. Caprolu M. Crypto mining makes noise / Caprolu. – 2019
70. Base64 [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Base64
71. Intel SDE [Електронний ресурс] – Режим доступу до ресурсу: https://www.intel.com/content/www/us/en/developer/articles/tool/software-development-emulator.html
72. Vigna G. Minesweeper: An in-depth look into driveby cryptocurrency mining and its defense / Vigna. – 2018
73. Rossberg A. Bringing the web up to speed with webassembly / Rossberg. – 2018
74. Webassembly [Електронний ресурс] – Режим доступу до ресурсу: https://webassembly.org//
75. Bian W. Minethrottle: Defending against wasm in-browser cryptojacking / Bian. – 2020
76. Petrov I. Coinpolice: Detecting hidden cryptojacking attacks with neural networks / Petrov. – 2020

77. Sukarno P. Mitigation of cryptojacking attacks using taint analysis / Sukarno. – 2019
78. Lachtar N. A crossstack approach towards defending against cryptojacking / Lachtar. – 2020
79. Tanana D. Behavior-based detection of cryptojacking malware / Tanana. – 2020
80. Mani G. Decrypto pro: Deep learning based cryptomining malware detection using performance counters / Mani. – 2020
81. Iframe [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe
82. Das S. Sok: The challenges, pitfalls, and perils of using hardware performance counters for security / Sok. 2019
83. Razali M. Cmblock: In-browser detection and prevention cryptojacking tool using blacklist and behaviorbased detection method / Razali. – 2019
84. Ramanathan S. Blag: Improving the accuracy of blacklists / Ramanathan. – 2020
85. Yadav S. Detecting algorithmically generated domain-flux attacks with dns traffic analysis / Yadav. – 2012
86. Dharmdasani H. Botcoin: Monetizing stolen cycles / Dharmdasani. – 2014
87. Eskandari S. “A first look at browser-based cryptojacking. – 2018
88. Holz R. A retrospective analysis of user exposure to (illicit) cryptocurrency mining on the web / Holz. – 2020
89. Sindre G. An experimental analysis of cryptojacking attacks / Sindre. – 2019
90. Cryptographic Hash Function [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Cryptographic_hash_function
91. Hunter D. Matplotlib: A 2D Graphics Environment / Hunter. – 2007
92. Audacity [Електронний ресурс] – Режим доступу до ресурсу: https://www.audacityteam.org/
93. Stratum [Електронний ресурс] – Режим доступу до ресурсу: https://braiins.com/stratum-v1/docs
94. Wireshark [Електронний ресурс] – Режим доступу до ресурсу: https://www.wireshark.org/

95. GPU Mining [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/tech/gpu-cryptocurrency-mining/
96. X86 instruction listing [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/X86_instruction_listings
97. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/
98. Combined crypto market capitalization races past $800 bln [Електронний ресурс] – Режим доступу до ресурсу: https://cointelegraph.com/news/combined-crypto-market-capitalization-races-past-800-bln

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ МАГІСТЕРСЬКОЇ

Статті в іноземних виданнях

1. Landovskyy Y. Solving Adaptive Security Architecture with SOAR / Y. Landovskyy, B. Tetiana. // 38th IBIMA Conference. – 2021.

Тези наукових доповідей:

1. Міжнародна науково-практична конференція “Проблеми кібербезпеки інформаційно-телекомунікаційних систем” (PCSITS)” 14 - 15 КВІТНЯ 2022, КИЇВ, Україна