НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри
_____ **Дмитро ЛАНДЕ**
(підпис)
«_____»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему: Технології протидії виявленню віртуалізації шкідливим ПЗ
Виконала: здобувачка вищої освіти **IV** курсу, групи ФБ-83_____
(шифр групи)

Григор'єва Ольга Олександрівна_____ 
(прізвище, ім'я, по батькові) (підпис)

Керівниця к.т.н., доцент кафедри ІБ Стьопочкіна Ірина Валеріївна 
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент д.т.н., с.н.с професор кафедри ММЗІ
Кудін Антон Михайлович 
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувачка вищої освіти 
(підпис)

Київ – 2022 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність– 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ ДмитроЛАНДЕ
(підпис)
«___»_____2022 р.

ЗАВДАННЯ
на дипломну роботу здобувачці вищої освіти

Григор'єва Ольга Олександрівна_____
(прізвище, ім'я, по батькові)

1. Тема роботи: Технології протидії виявленню віртуалізації шкідливим ПЗ,

керівниця роботи к.т.н., доцент кафедри ІБ Стьопочкіна Ірина Валеріївна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « »_____2022 р. №

2. Термін подання здобувачкою вищої освіти роботи 10 червня 2022 р.

3. Вихідні дані до роботи: наукова література та відкриті джерела за темою; попередні дослідження за темою; база даних технік ухилення ATT&CK від MITRE.

4. Зміст роботи: опис існуючих технік ухилення від виявлення віртуалізації; теорія щодо способів продії ним; практичні приклади заходів маскування; класифікація технік ухилення; статистичне дослідження технік ухилення, які використовують ШПЗ; таблиці у форматі "техніка – протидія техніці" за кожною окремою категорією.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація на тему «Технології протидії виявленню віртуалізації шкідливим ПЗ».

6. Дата видачі завдання18 жовтня 2021 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Вибір теми роботи	20.09.2021 – 15.10.2021	
2	Отримання завдання	18.10.2021	
3	Вивчення літератури за темою, вибір літературних джерел	19.10.2021 – 12.01.2022	
4	Вивчення технік ухилення, написання першого розділу	13.01.2022 – 20.02.2022	
5	Вивчення протидії технікам, написання другого розділу	21.02.2022 – 15.04.2022	
6	Аналіз отриманих знань та результатів	16.04.2022 – 01.05.2022	
7	Проходження переддипломної практики, написання третього розділу	02.05.2022 – 30.05.2022	
8	Отримання допуску до захисту	01.06.2022	
9	Створення презентації для захисту дипломної роботи	02.06.2022 – 12.06.2022	
10	Передзахист дипломної роботи	13.06.2022	
11	Доопрацювання дипломної роботи та презентації	14.06.2022 – 19.06.2022	
12	Захист дипломної роботи	20.06.2022	

Здобувачка вищої освіти  Ольга ГРИГОР'ЄВА

(підпис) (**Власне ім'я, ПРІЗВИЩЕ**)

Керівниця роботи Ірина СТЬОПОЧКІНА

(підпис) (**Власне ім'я, ПРІЗВИЩЕ**)

РЕФЕРАТ

Робота складається з 3 розділів, містить 16 іллюстрацій, 10 таблиць, 1 додаток та 33 літературні посилання, обсяг роботи – 72 сторінки.

Завданням роботи є аналіз технік ухилення від виявлення віртуалізації, вивчення методів продії ним, розробка класифікації та проведення статистичного дослідження технік ухилення, які використовують ШПЗ, та зведення результатів роботи у таблиці у форматі “техніка – протидія техніці” за кожною окремою категорією.

Метою роботи є створення методики щодо протидії виявленню шкідливим програмним забезпеченням, чутливим до середовища виконання, віртуальних машин на базі десктопної версії ОС Windows.

Предметом дослідження є: технології виявлення та протидії виявленню віртуалізації. Об’єктом дослідження є: шкідливе програмне забезпечення, чутливе до середовища виконання.

Наукова новизна полягає у тому, що не існує єдиної бази технік ухилення у форматі «техніка – протидія її», а статистичні дослідження на базі даних з ATT&CK від MITRE не проводились.

Ключові слова: шкідливе програмне забезпечення, чутливе до середовища виконання; анти-віртуалізація; ухилення від пісочниць; шкідливе програмне забезпечення, що ухиляється від віртуалізації.

ABSTACT

The work consists of 3 sections, contains 16 illustrations, 10 tables, 1 appendix and 33 references, the volume of work – 72 pages.

The task of the work is to analyze the techniques of virtualization evasion, to study the methods of its implementation, to develop a classification and statistical research on evasion techniques, that are used by malware, and to summarize the results in a table in the format " technique - its countermeasure" for each category.

The aim of the work is to create a methodology for counteracting the detection of virtual machines based on the desktop version of Windows by context-aware malware.

The subject of research are: virtualization detection technologies and their counteractions. The object of research is: context-aware malware.

The scientific novelty is that there is no full databases of evasion techniques in the format of "technique - its countermeasure", and statistical studies on the database of ATT & CK from MITER have not been conducted.

Keywords: context-aware malware; anti-virtualization; sandbox evasion; evasive malware.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система

ВМ – віртуальна машина

ПЗ – програмне забезпечення

ШПЗ – шкідливе програмне забезпечення

ІБ – інформаційна безпека

ІВ – індикатор віртуалізації

ВСТУП

Технології віртуалізації – надзвичайно важлива частина інформаційних систем, не тільки через усі умови для зручного використання та тестування різноманітних ОС або застосунків у різних середовищах, конфігураціях, при інакших потужностях обладнання, і все це на одному комп'ютері без необхідності додатково їх налаштовувати або навіть виділяти окремі машини, а ще й через можливість надання відокремленого від комп'ютера-хоста середовища, в якому можна виконувати та аналізувати будь-який зразок підозрілого ПЗ, не наражаючи на небезпеку реальну систему. Звісно, розробники ШПЗ знають про такий метод досліджень і роблять усе можливе для того, аби замаскувати свої програми від виявлення, адже програма, яка залишає сліди свого виконання або відкрито демонструє небезпечні дії, легко підлягає докладному вивченню та у майбутньому – створенню протидії. Для ШПЗ не так важливо не виказати свою присутність у реальній системі, як у віртуальній лабораторії, адже останнє буде означати повну поразку, тому для них вміння відрізнити дійсну ОС від пісочниці – життєва необхідність.

Актуальність роботи визначається широким застосуванням віртуалізації в технологіях захисту від ШПЗ з одного боку, та технік ухилення ШПЗ від запуску в віртуальному середовищі - з іншого, адже донині ця задача не має рішення. За статистикою «понад 98 відсотків шкідливих програм … використовують принаймні одну тактику ухилення, і 32 відсотки зразків ШПЗ … містять шість або більше методів ухилення від виявлення»[1].

Перший розділ даної роботи буде описувати досягнення порушників ІБ у мистецтві виявлення середовища віртуалізації, другий – техніки вдосконалення маскування віртуальних машин, що їх використовують фахівці з безпеки. Завершальний – третій розділ – представить узагальнену для ВМ на базі Windows методику з ціллю завадити визначенню чи є система пісочницею, чи ні, а також статистику щодо використання різних технік ухилення.

Мета роботи: дослідження існуючих технік ШПЗ по виявленню віртуалізації, виявлення тенденцій у галузі, узагальнення відповідних їм засобів протидії та розробка методики щодо їх застосування, тестування методики на практиці.

Мета роботи реалізується за допомогою наступних **задач дослідження**:

1. Дослідити способи ухилення від віртуалізації на базі доступної літератури;
2. Провести аналіз та розділити на категорії виділені техніки;
3. Визначити та описати існуючі методи протидії виявленню віртуалізації для загального використання, навести приклади на основі віртуального середовища VMware Workstation та на платформі ОС Microsoft Windows 10;
4. Провести статистичний аналіз технік ухилень на базі проекту ATT&CK від MITRE;
5. Скласти методики застосування рішень по запобіганню виявленню віртуалізації для кожної з категорій ухилень, що використовують сучасні ШПЗ, у вигляді відповідних настанов для дослідника ШПЗ, для ВМ на базі десктопної версії Windows;
6. Розробити комплексний підхід до виявлення та маскування віртуалізації на основі створеної методики та продемонструвати його ефективність.

Предметом дослідження є: технології виявлення та протидії виявленню віртуалізації. **Об’єктом дослідження** є: шкідливе програмне забезпечення, чутливе до середовища виконання.

Методом дослідження є статистичний метод аналізу відкритих джерел інформації щодо ШПЗ, чутливого до середовища виконання, та наукових публікацій, присвячених темі, з подальшим узагальненням отриманих знань у єдину базу методик щодо посилення захисту ВМ від такого ШПЗ, створення програми, яка реалізуватиме комплексний підхід до виявлення ВМ на базі десктопної версії Windows.

Наукова новизна полягає у тому, що не існує єдиної бази технік ухилення у форматі «техніка – протидія її», статистичні дослідження на базі даних з ATT&CK від MITRE не проводились, а комплексний підхід до маскування віртуалізації не застосовувався.

Практичне значення одержаних результатів полягає в тому, що зведені методики, що будуть наведені у Розділі 3 можуть ефективно та зручно використовуватись фахівцями, які розробляють захищене віртуальне середовище або просто посилюють власні системи. Повнота теоретичних методик дозволяє застосовувати тільки даний перелік і звертатися до інших джерел лише за докладними практичними вказівками. Комплексний спобів маскування віртуалізації може бути у майбутньому перетворений на автоматизований потужний інструмент.

Робота була **опублікована** у збірнику матеріалів XX Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» [2].

1 МЕХАНІЗМИ ШПЗ ДЛЯ ВИЯВЛЕННЯ ВІРТУАЛІЗАЦІЇ

Оскільки віртуальне середовище, хоча і симулює роботу ОС, проте все ж відрізняється багатьма параметрами від такої, що її реально використовує користувач, ШПЗ не втрачає зручного шансу перевірити ці показники з метою дізнатись, що його запустили не у цільовій системі, а у деякій лабораторії. Якщо ж буде виявлено свідчення, що вказують на віртуалізацію, зловмисна програма завершить свою роботу та приховає себе від дослідників, а може навіть самознищитися. Параметри ці, як і способи визначення віртуалізації середовища, яких автори навчають своє ШПЗ, дуже різноманітні, починаючи від простої перевірки назв запущених процесів до зчитування температури батареї, яку звісно пісочниця не має. У цьому розділі розглянуто та згруповано відомі техніки та прийоми, які може застосовувати ШПЗ.

Умовно всі техніки ухилення від виявлення можна поділити на 4 великі групи, засновані на:

1. «Взаємодії з людиною – натискання миші та діалогові вікна;
2. Специфічній конфігурації – відкладене виконання, тригери на виконання у певний час, приховування процесів, шкідливі завантажувачі, відмова запуску у разі спроби перейменування, зчитування інформації про унікальний номер розділу жорсткого диску, виконання після перезавантаження системи;
3. Специфічному середовищу – перевірка версії ОС або програм, приховування шкідливого коду у не виконуваних файлах (jpg,gif), у динамічних бібліотеках (DLL);
4. Специфічній ВМ – списки системних служб, унікальні файли та порти, які використовуються для комунікації з хостом (vmx port)» [3, ст.3].

Інший варіант класифікації пропонує більш стислий поділ на техніки, засновані на взаємодії з користувачем (1), наявність певних артефактів (2) та технологію, засновану на затримці часу виконання (3) [4, ст.5]. Незалежно від

способу поділу на види, усі техніки будуть розглянуті в роботи, але розбиті на більш специфічні групи, аби приділити достатньо уваги кожній.

Також у роботі вводиться термін **«Індикатор Віртуалізації»** як поняття, схоже на Індикатор Компроментації, проте з тією різницею, що воно означає об’єкт або активність у цільовій системі, присутність якого може з великою долею вірогідності свідчити про віртуальну природу середовища. Буде виділено пасивні та активні ІВ, де перші являються по суті артефактами різної природи, які є складовими ВМ, а другі – представляють собою певну подію, яка прододиться користувачем або програмою-користувачем.

1.1 Різниця у термінах

Перед тим як приступити до опису технологій виявлення віртуалізації, потрібно визначити різницю між термінами, бо існують три технології, які можна сплутати: анти-віртуалізація (anti-vm), анти-пісочниця (anti-sandboxing) та анти-налагоджувач (anti-debugging). Анти-віртуалізація – тема, якій по більшій частині присвячена дана робота – це набір технологій, які опираючись на перевірки лише пов’язаних з ВМ показників та маркерів роблять висновок про віртуальність або реальність цільової машини. Анти-пісочниця представляє собою набір перевірок, які дають непрямі докази віртуалізації, такі як присутність на машині засобів для аналізу ШПЗ динамічно чи статично, що саме по собі ще нічого не значить, але може бути використане як додатковий фактор при прийнятті рішення щодо природи середовища. Останній термін – анти-налагоджувач, як зрозуміло з назви, позначає групу методів по виявленню роботи налагоджувача на комп’ютері, часто ці методи називають у більш широкому розумінні другим згаданим терміном.

1.2 Перевірка значень ключів реєстру

Одним із методів, який важко виявити через його удавану легітимність, є зчитування полів значень ключів у реєстрі. Дійсно, таке звернення може виконати будь-який законний користувач або системний процес, багато застосунків взагалі не зможуть працювати без таких запитів, адже реєстр використовується ними як база даних, де зберігаються налаштування, шляхи до необхідних їм ресурсів, але принципова різниця між безпечними та навпаки шкідливими операціями у ключах, значення яких перевіряється: ШПЗ для виявлення пісочниці буде шукати системні змінні, які містять у своїй назві щось, що вкаже на назву віртуального середовища, як наприклад ключі з частинками “Vmware”, “VirtualBox”, “Hyper-V”, або характерними саме для цих засобів віртуалізації іншими значеннями ключів. Перелік рядків, які ШПЗ може шукати у реєстрі, можна знайти наприклад на сайті-енциклопедії the Malware Evasion Encyclopedia, що присвячена різним технікам зловмисних програм для ухилення від виявлення [5]. Не можна виключати ситуації, коли таке звернення може знадобитись одній із служб самої ВМ, але через відносно високу ефективність виявлення віртуалізації та простоту такої техніки, майже кожне ШПЗ використовує її. Ключі реєстру належать до пасивних ІВ, бо являються складовою системи.

Для того, аби продемонструвати як це може працювати, використано віртуальну машину Windows 10, запущену у моніторі віртуальних машин Vmware Workstation, для пошуку у реєстрі ключів, наведених у вже згаданій енциклопедії, як це видно на Рисунку 1.1:

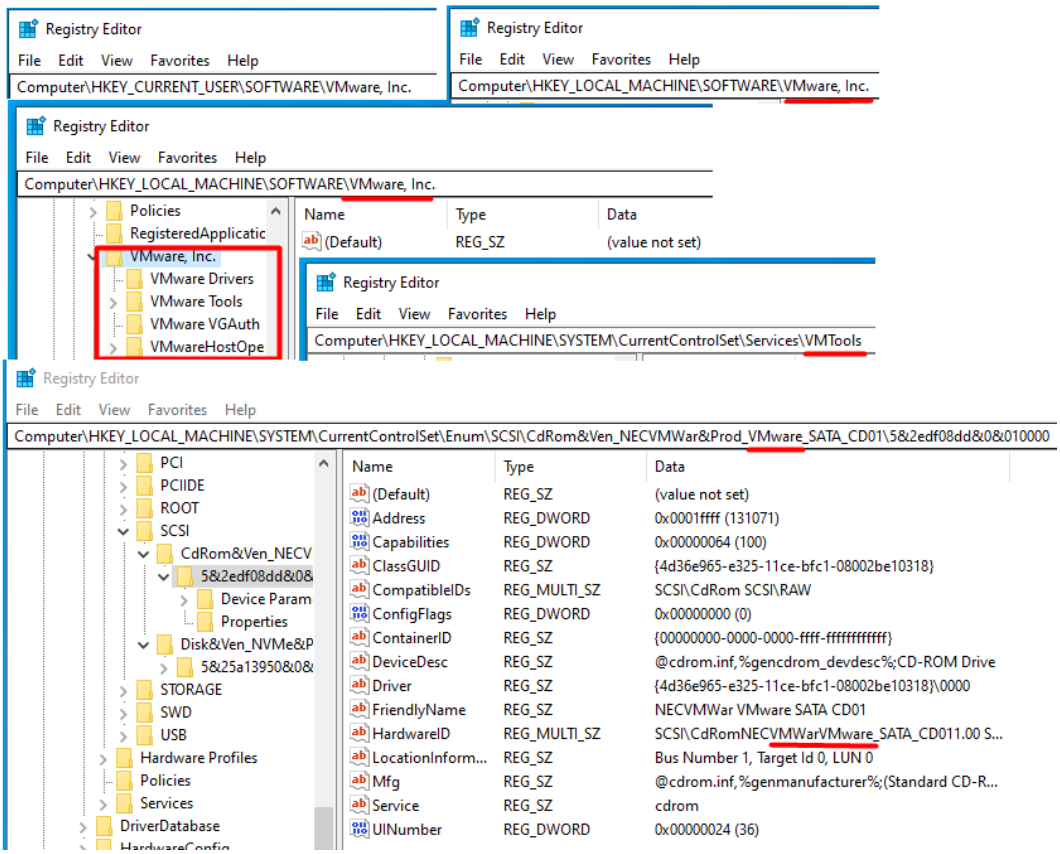

Рисунок 1.1 – Ключи реєстру ОС Windows з рядками “Vmware, Inc.” та “Vmware Tools”

На Рисунку 1.1 показано лише деякі приклади шляхів ключів реєстру, де може бути знайдена інформація про приналежність даної системи до віртуальних. Можливе також хибне спрацювання механізму перевірки ШПЗ: якщо на комп’ютері встановлена віртуальна машина, то пошук не у специфічних місцях реєстру, як було наведено на Рисунку 1.1 (як Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\SCSI або Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SystemInformation і т.п.), а наприклад загальний пошук може призвести до виявлення бажаного рядка у переліку програм. Насправді, не завжди у назві чи значенні ключа обов’язково має бути рядок, що містить саме назву програми для віртуалізації, аби ШПЗ зрозуміло, що його запуск відбувся у пісочниці, є також інші докази: кожен реально існуючий комп’ютер має фізичні параметри, які відрізняються від віртуальної машини; або ж наявність технології віртуалізації

може бути також виявлена за допомогою стандартних значень, які використовуються певним її розробником, цьому будуть присвячені наступні пункти.

Окрім прямого звернення до реєстру, ШПЗ можуть викликати системні утиліти, наприклад systeminfo, аби витягнути потрібну інформацію, по суті теж з реєстру. Зрозуміло, що вивід такої команди буде відрізнятись у різних середовищах. Так на Рисунку 1.2 показано вивід утиліти у існуючому фізично комп'ютері, на Рисунку 1.3 – у віртуальному середовищі:

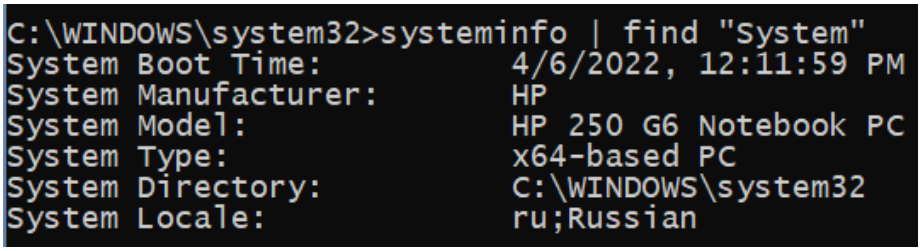

Рисунок 1.2 – Результат роботи команди у реальній машині

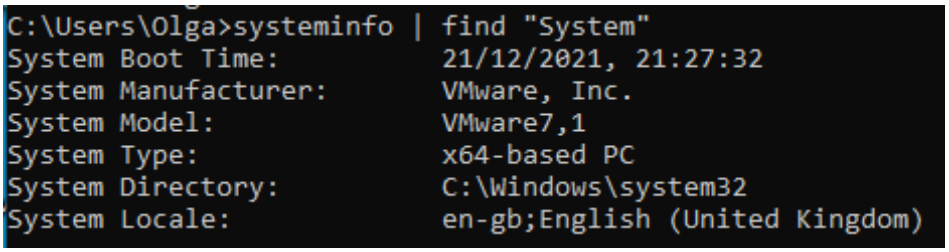

Рисунок 1.3 – Результат роботи команди у віртуальній машині

1.3 Перевірка на активність користувача

Кожен комп'ютер, яким реально користуються, людина з часом налаштовує під себе: завантажує якісь файли, робить пошукові запити у браузері, обирає зручне розширення екрану… При використанні машини також працює певна кількість процесів, натискаються кнопки миші та клавіатури, відкриваються вікна – все це ШПЗ може використати, аби дізнатися, що вона не "на свободі", а під пильним наглядом фахівця, бо мало хто буде встановлювати на віртуальну машину наприклад пакет програм, без яких не можна уявити

реальний комп’ютер: офісні редактори, браузери, месенджери, не кажучи вже про розширення екрану, яке у віртуальній машини сильно відрізняється від звичного. Варто зазначити, що ситуація, коли ШПЗ потрапляє на новий та повністю чистий комп’ютер, що може за деякими із згаданих параметрів нагадувати віртуальне середовище, вважається розробниками зловмисного коду менш ймовірною, ніж перебування під дослідом.

Всі ці перевірки дійсно доволі легко обходяться, але це не заважає ШПЗ раз за разом успішно їх використовувати. За запропонованою класифікацією, перевірки активності експлуатують активні індикатори віртуалізації.

1.3.1 Файлова активність

Деякі ШПЗ перевіряють список останніх відкритих файлів, аби впевнитись що їх кількість більша за деяку прийнятну – ця цифра залежить виключно від власної думки автора ШПЗ, переважно від 3 до 10. Подивитися список нещодавніх документів для кожної окремої програми можна у реєстрі, наприклад саме у ключі Computer\HKEY_CURRENT_USER\ SOFTWARE\Microsoft\Office\14.0\Word\File MRU, вміст якого показано на Рисунку 1.4, там будуть зберігатися останні відкриті у Microsoft Office Word документи:

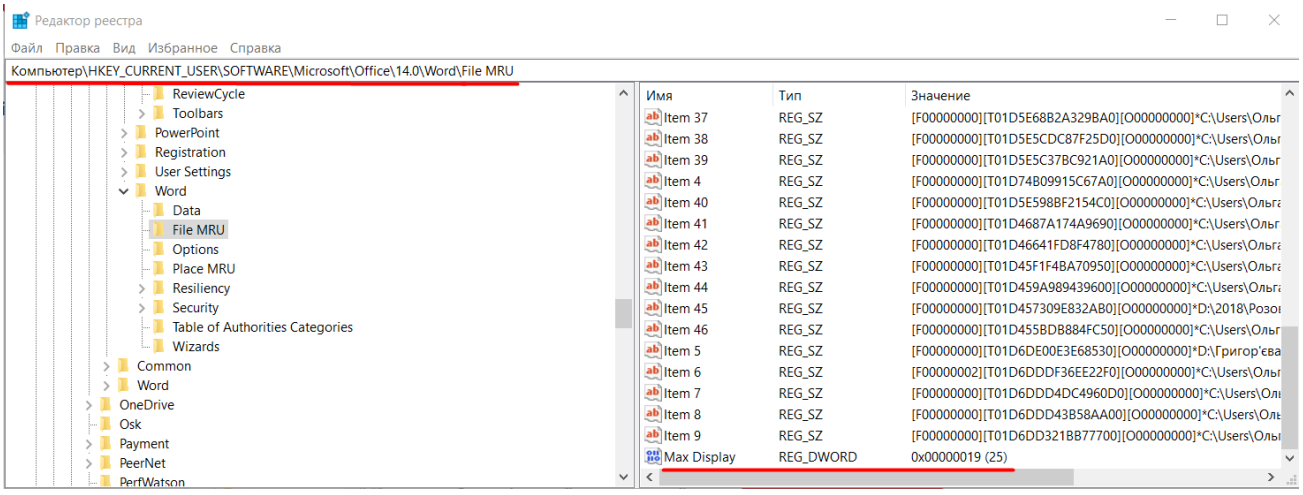

Рисунок 1.4 – Останні відкриті файли у реєстрі

У проілюстрованому місці реєстру можна також знайти кількість файлів, що відображається у переліку – 25, проте для ШПЗ важливо щоб їх було не менше вказаної у його коді кількості.

1.3.2 Активні процеси

Подібно до того, як перевіряється присутність файлів у історії нещодавно відкритих, піддаються аналізу активні процеси – не тільки їх назва (яка може свідчити про наявність пов’язаних з пісочницею програм, або таких, що можуть бути використані для дослідження ШПЗ, як дебагери чи інструменти захоплення мережного трафіка, про що буде згадано пізніше), а радше їх кількість, адекватна звичайному використанню комп’ютера. Користувач вірогідніше за все буде мати принаймні 2-3 активні процеси, тоді як віртуальна машина може бути “неактивною”. Інша варіація – контроль кількості відкритих вікон, які звісно теж представлені процесами деякого застосунку, але їх численність може бути вищою, бо один додаток часто генерує декілька вікон для своєї роботи. Вже згадана the Malware Evasion Encyclopedia стверджує, що машина, з менш ніж з 10 відкритими вікнами, скоріш за все є віртуальною, бо через обмеженість ресурсів у пісочниці недоцільно тримати активними таку кількість процесів [6].

1.3.3 Історія браузеру

Як одна з простих, але дуже ефективних технік виявлення віртуалізації, може використовуватися перевірка історії браузера, бо важко уявити комп’ютер, з якого б не йшли пошукові запити, при цьому синтаксис перевірки буде різним для різних браузерів. Кількість потрібних записів в історії теж обирається автором ШПЗ за власним розсудом.

1.3.4 Активність миші та клавіатури

Віртуальні середовища, що призначені для аналізу зловмисного ПЗ часто працюють в автоматичному режимі через велику кількість зразків, які потрібно перевірити, існують також спеціальні пісочниці на кшталт Cuckoo Sandbox чи JoeSandbox, що мають онлайн-інтерфейс для завантаження підозрілих файлів на перевірку, вони мають конфігурації, визначені авторами, тому по-перше, розробникам ШПЗ легко виявити маркери віртуалізації, бо пісочниці загально відомі та їх алгоритм дії частково можливо зрозуміти з відкритих джерел або звітів досліджень вірусів, а по-друге, вдало використовувати такий спосіб виявлення як контроль активності вводу користувача. Через те, що аналіз відбувається самою системою, втручання реальної людини виключено. Тоді як натискання миші або клавіші клавіатури може бути виконане від імені будь-якої програми з достатніми правами, деякі дії може провести лише людина, такі як CAPCHA, вибір малюнку з запропонованих, інші способи графічної перевірки «на робота». ШПЗ може застосувати такі нехитрі механізми, аби виявити, що має діло з автоматизованим дослідником-програмою, з чого зрозуміло що такий варіант не буде працювати з фахівцем-людиною чи звичайним користувачем, яка легко пройде цей тест.

Проте такі онлайн-пісочниці не є цільовою аудиторією ШПЗ: воно переважно діє з позиції «дізнатися, що я під дослідом», а не «дізнатися, що я НЕ під дослідом», тому шукає такі докази, які однозначно або хоча б з високою вірогідністю кажуть про наявність віртуалізації, і не важливо стоїть за нею людина чи програма-аналізатор. Саме тому, перевіряючи кількість подій – натисків миші чи клавіатури – ШПЗ робить припущення, що наявність таких взаємодій викликана присутністю користувача у зараженій системі.

Таке припущення просто необхідне, адже для операційної системи, а отже і для будь-якої програми, у тому числі шкідливої, немає різниці між натиском кнопки, виконаним людиною, та натиском, що є лише результатом роботи відповідного рядку коду. Тому якщо припустимо деяка технологія пісочниці

буде настільки добре запрограмована, аби симулювати деяку, бажано хаотичну (щоб унеможливити подальше викриття через заздалегідь відомі дії), активність користувача, ніяке просте ШПЗ не зможе, принаймні за цим маркером, зробити позитивний висновок щодо віртуальності зараженого середовища. З часом, напевно, з'явиться ШПЗ, яка зможе аналізувати рухи курсору мишки та робити висновок щодо їх «людського походження»: таке є можливим через недосконаліть рухів, які людина робить курсором, тоді як рухи, які буде робити програма, аби змінити положення вказівника миші, будуть однозначно мати вигляд прямих ліній. У свою чергу, розробники середовищ для аналізу можуть навчити свій застосунок виконувати подібні до людських дії, проте обидві ці розробки принаймні на даний момент є просто надлишковими у сфері зловмисного ПЗ через свою складність та низьку ефективність, бо все ще існує безліч більш простих способів виявити віруталізацію.

Тому зараз ШПЗ перевіряє переважно просто чи змінюється положення курсору за визначений час, чи відбуваються кліки чи натискання клавіш. До речі, вже згадана пісочниця Cuckoo Sandbox «має обмежені можливості для запобігання ухиленням(evasions), наприклад, моделювання випадкових рухів миші» [7, ст.2]. Наприклад, знайдений у 2017 році банківський троян Ursnif оснащений «алгоритмом, що використовує різницю між поточними та попередніми записаними координатами миші, щоб виявити рух миші та уникнути середовищ пісочниці, де миша зазвичай не рухається»[8]. Після отримання результату перевірки, який застосовується як ключ для дешифровки, троян або продовжує роботу, розпаковуючи архіви з зловмисним кодом, або завершається через неправильне значення ключа.

1.3.5 Розширення екрану

Розширення екрану залежить не лише від побажань власника, але і від розміру монітора, тому переважна більшість користувачів все рівно

використовує стандартні значення. Список розширень та відсоток комп'ютерів у всьому світі [9], які їх використовують, наведено у Таблиці 1.1:

Таблиця 1.1 – Поширеність розширень екрану

Розширення	% комп'ютерів у світі
1920×1080	23,1
1366×768	18,54
1536×864	10,27
1440×900	6,09
1280×720	5,91
1600×900	3,4
2560×1440	2,68
1280×1024	2,35
768×1024	1,99

У сумі всі наведені розширення складають близько 75% від налаштувань усіх комп'ютерів, що дає можливість з великим шансом стверджувати, що якщо розширення досліджуваного комп'ютера інакше, то це віртуальна машина. Можливо вірогідність і здається порівняно малою для точного визначення, але треба пам'ятати, що більшість маркерів для визначення віртуалізації не використовується по одному, а у комбінації з іншими, що дає можливість за загальними результатами зробити більш точний висновок.

Інше діло, що змінити розширення екрану – елементарна дія, але дія, яка може потребувати встановлення додаткових інструментів, наприклад VMware Tools, які допомагають зробити роботу з віртуальною машиною приємнішою для користувача. Очевидно, встановлення такої програми неодмінно приверне увагу ШПЗ. Без таких сторонніх засобів розширення буде мати значення за замовчуванням для обраної технології віртуалізації: 1024×768 (чи навіть 800×600) для наприклад VirtualBox, Cuckoo Sandbox та JoeSandbox. Тому хоча такі перевірки і здаються застарілими та безрезультатними, на практиці вони все ще використовуються. Так, ШПЗ TrickBot – троян, ціллю якого є викрадення банківських даних, має таку перевірку у своєму арсеналі [10].

1.4 Перевірка фізичних характеристик

Окрім перевірки програмних характеристик, ШПЗ може звертати увагу на технічні параметри фізичного комп'ютера: відомо, що для підтримки більш-менш комфортної роботи з ним потрібне виконання хоч мінімальних вимог до потужності процесора, об'єму оперативної та внутрішньої пам'яті, наявність звукової карти і т.п. Якщо ж з програмним забезпеченням все в цілому зрозуміле: віртуальна машина майже завжди оснащується новими версіями ОС та програм з причин ефективності досліджуваної роботи, то з технічною стороною все навпаки. Для ВМ виділяють мінімальний об'єм внутрішньої пам'яті, який буде достатнім для встановлення необхідних застосунків, бо більше ні для чого вона використовуватись і не буде; для неї не виділять потужність всього процесора, а часто обмежаться один ядром; щодо звукової карти, то вона може бути взагалі відсутньою, бо не часто на ВМ потрібно щось прослуховувати, на відміну від звичайного комп'ютера.

Таким чином, перевірка кількості ядер процесора системи може дати ШПЗ достатньо точну відповідь на питання чи знаходиться вона під дослідом, особливо якщо результат виявляється таким, якого бути в реальності не може: не існує процесора з 5 ядрами, мало хто вже використовує процесор з 1 або 2 ядрами, а от виділити таку кількість для ВМ, коли маєш потужний процесор на хостовій машині, дуже легко. Комп'ютери з процесором, що має одне ядро звісно теж ще десь існують, але навіть для комфортного використання браузера такої потужності недостатньо, тому якщо після, наприклад після виконання команди для отримання такої інформації у командній строчці, ШПЗ отримує інакший від можливого результат, з великим шансом цього буде достатньо, аби воно припинило свої зловмисні дії. Результат роботи команди, яка виводить дані про кількість ядер процесора, у реальній машині видно на Рисунку 1.5:

```
C:\WINDOWS\system32>WMIC CPU Get DeviceID,NumberOfCores,NumberOfLogicalProcessors
DeviceID  NumberOfCores  NumberOfLogicalProcessors
CPU0      2              4
```

Рисунок 1.5 – Команда для перегляду кількості ядер процесора

ШПЗ може використовувати інші способи отримання бажаної інформації: зчитувати відповідне поле зі структури даних PEB (Process Environment Block), яка містить в собі якраз дані про процесор, у тому числі кількість його ядер, або більш креативним способом: через маску споріденості процесу (process affinity mask), яка являє собою рядок бітів, де кожен біт «представляє процесор, і, отже, просто підрахувавши встановлені біти, можна визначити загальну кількість ядер в системі» [11]. Для отримання цієї маски треба викликати winapi-функцію GetProcessAffinityMask.

З кількістю оперативної та внутрішньої пам’яті все подібно: автор ШПЗ додає перевірку аби пам’яті не було менше за визначене ним самим число, типово це 80GB для внутрішньої та 1GB для оперативної пам’ятей.

Дослідники з Cisco, аналізуючи новий APT-вірус GravityRAT у 2018 році [12] виявили цікавий спосіб перевірки – зчитування температури процесора: віртуальною машиною може бути повернути або некоректне значення, або взагалі нічого, що зрозуміло, адже фізично ніякого процесору вона не має. Вони також зазначають «деякі гіпервізори (VMWare, VirtualBox і Hyper-V) не підтримують перевірку температури. На запит WMI просто видає повідомлення "не підтримується". Цю поведінку можна використовувати, щоб визначити, чи є цільова система справжньою машиною». На Рисунку 1.6 зображена спроба виконання команди, яка має показати температуру термальних зон комп’ютера, у віртуальній машині, запущеній в VMware:

```
C:\Users\Olga>wmic /namespace:\\root\WMI path MSAcpi_ThermalZoneTemperature get CurrentTemperature
Node - DESKTOP-71VI00M
ERROR:
Description = Not supported
```

Рисунок 1.6 – Результат роботи команди з виводу температури процесора у ВМ

До перевірок, націлених на технічні характеристики можна також віднести зчитування виробника жорсткого диску: деякі системи для віртуалізації зазначають там свою назву.

1.5 Перевірка артефактів

1.5.1 Артефакти-процеси та файли

Рядки, що пов’язані з певним віртуальним середовищем, можуть бути виявлені не лише в реєстрі, а і у списку активних процесів, встановлених програм, динамічних бібліотек, у переліку файлів; «перераховуючи назви процесів і служб, зловмисне програмне забезпечення може порівняти їх із жорстко закодованим списком, наданим автором ШПЗ. Цей жорстко закодований список міститиме відомі значення імен процесів або служб, які можуть працювати в системі, такі як vmsrvc.exe, vmusrvc.exe, vboxtray.exe, vmtoolsd.exe, df5serv.exe, vboxservice.exe і т.д.» [13, ст.6]. На додачу може бути проведений пошук серед об’єктів файлової системи по ключовим рядкам, що містять назву пісочниці, у тому числі серед встановлених динамічних бібліотек, наприклад як на Рисунку 1.7 [13, ст.8] для VMware:

- Vmware:
  - *C:\windows\System32\Drivers\Vmmouse.sys*
  - *C:\windows\System32\Drivers\vm3dgl.dll*
  - *C:\windows\System32\Drivers\vmdum.dll*
  - *C:\windows\System32\Drivers\vm3dver.dll*
  - *C:\windows\System32\Drivers\vmtray.dll*
  - *C:\windows\System32\Drivers\VMToolsHook.dll*
  - *C:\windows\System32\Drivers\vmmousever.dll*
  - *C:\windows\System32\Drivers\vmhgfs.dll*
  - *C:\windows\System32\Drivers\vmGuestLib.dll*
  - *C:\windows\System32\Drivers\VmGuestLibJava.dll*
  - *C:\windows\System32\Driversvmhgfs.dll*

Рисунок 1.7 – Приклад динамічних бібліотек з системи VMware

Серед сервісів, які можуть перевірятися, також є VMTools, vmvss, VGAuthService, VMware Physical Disk Helper Service для VMware [14].

1.5.2 Мережні артефакти

До системних артефактів можна також віднести загальновідомі номера MAC-адрес, номера портів для комунікацій з хостовою машиною, ім’я мережевого адаптера також може містити назву ВМ. Перші три байта MAC-адреси вказують на постачальника мережевого адаптера, тому враховуючи що для кожного віртуального середовища вони добре відомі, ШПЗ може легко ухилитись від виявлення, якщо опиниться у системі з такими MAC-адресами. Значення для відомих ВМ наведені у Таблиці 1.2 [13, ст.7], [14]:

Таблиця 1.2 – Стандартні MAC-адреси для ВМ

Віртуальне середовище	MAC-адреси
VMware	00:50:56; 00:1C:14; 00:0C:29; 00:05:69;
Hyper-V	00:03:FF; 00:15:5D
VirtualBox	08:00:27

Щодо можливості виявлення спілкування між гостьовою та хостовою машинами, то наприклад платформа для кібершпигунства Attor «намагається отримати версію VMware, зв’язуючись із портами вводу/виводу, використовуючи [асемблерну] in інструкцію. Якщо запуск відбувся в середовищі VMware як результат повертається магічне значення VMXh». [15, ст.8]

Іншим можливим пасивним індикатором віртуалізації для ШПЗ може виступати час останнього перезавантаження мережевого адаптера: у реальних системах він часто змінюється, на відміну від віртуальних.

1.6 Спрацювання тригеру-умови

Деякі ШПЗ після потрапляння у систему залишаються неактивними і щоб почати свої зловмисні дії чи просто перевірки середовища, у яке вони потрапили, на відповідність бажаним умовам (окрім спроб виявити, що вони під дослідом, ШПЗ може також шукати і заражати системи певного класу,

наприклад банківські чи державні установи, системи, що належать певній компанії, людині або ті, що просто знаходяться у потрібному регіоні), вони можуть чекати настання певних сприятливих заздалегідь визначених умов. Такі умови бувають двох типів: Time Bomb та Logic Bomb, і оскільки настання цих умов є по суті подіями – вони є активними ІВ:

1.6.1 Очікування певного часу

Найпростішим, але все ще актуальним способом уникнути виявлення для ШПЗ є затримка виконання (Time-delay based evasion) через виклик функцій «Sleep» або «NtDelayExecution» у Windows з довільним інтервалом часу очікування. Такий метод виправдовує себе через те, що автоматизовані системи для аналізу зловмисного коду працюють з великою кількістю файлів і просто вимушені визначати найбільший допустимий час для однієї перевірки, щоб уникнути ситуації припинення функціонування через один «завислий» процес. Сторінка на сайті Mitre, присвячена техніці Time Based Evasion містить приклади ШПЗ, які нею володіють: ця інформація показує по-перше, що донині такий спосіб використовується (найновіший приклад – троянізована DLL SUNBURST – датована жовтнем 2021 року) , а по-друге, час такого очікування може складати від 3 хвилин до 15 днів, при чому він може бути не тільки чітко визначеним, але і випадковим [16]. Варіацією такого підходу може бути запланованe виконання у певний час, день, свято і т.д.

Розробники ВМ розуміють, що в такий спосіб файли уникають дослідження, тому деякі системи виявляють «сплячі» програми та модифікують системний час так, щоб ШПЗ перевірило що він дійсно пройшов та активізувалося. У свою чергу, ШПЗ навчилось «виявляло ці виправлення як додатковий ІВ середовища аналізу/пісочниці. Це робиться шляхом отримання позначки часу, переходу до сну та перевірки часової позначки після пробудження. Якщо різниця в часі від попередньо зробленої позначки часу

істотно відрізняється від часу, коли зловмисне ПЗ було запрограмовано на сплячий режим, зловмисне ПЗ уникає або коригує своє виконання» [17].

Одним із дещо ефективніших та менш підозрілих способів відкласти виконання шкідливого коду є додавання перед ним безневинного гальмуючого коду (так званий stalling code), який потребує багато обчислювальних ресурсів або просто дублює себе. Таким чином, наприклад складні криптографічні операції можуть затягнутися на час, більший за той, який стандартно виділений під аналіз одного досліджуваного файлу. Аби не дати ВМ чи фахівцю зрозуміти що всі зроблені дії були лише для подовження часу виконання програми, результат операції «більш складне ШПЗ може використовувати на наступних частинах виконання (наприклад, отримання криптографічного ключа в результаті багатьох трудомістких операцій), щоб змусити системи аналізу дійсно виконувати гальмуючий код» [18, ст.16].

1.6.2 Очікування певної дії

Також умовою для початку активних дій для ШПЗ окрім часу може виступати будь-яка визначена діяльність користувача або системи – так звана Logic Bomb. Подібно до перевірки на активність користувача, коли ШПЗ чекало зміни положення курсора чи натиску клавіш, ця техніка переводить зловмисну програму у стан очікування наприклад доки буде проведене подвійне натискання курсором на зображенні у документі з зараженим макросом (так діє ШПЗ від групи FIN7) [19] або звичайна дія на кшталт створення чи модифікації файлу. До інших дій-тригерів можуть належати також запити до реєстру, створення процесів, інтернет-активність, прокручування відкритих документів та перезавантаження системи.

1.7 Час виконання операцій

Аби виконати деякі операції, що потребують залучення процесору чи інших частин хостової машини, ВМ потрібен додатковий час. ШПЗ навчилось використовувати такі операції для того, аби виконавши їх та проаналізувавши час виконання робити висновок про природу середовища. Для таких операцій у англомовній літературі існує термін “red pill”.

Однією з найпопулярніших технік перевірки є виклик інструкцій, які для свого виконання потребують виходу з ВМ, а отже на такий вихід та подальше повернення відповіді буде витрачено більше часу, ніж потрібно було б у реальній машині. До інструкцій, що потребує відповіді від реального фізичного процесора належить RDTSC (Read Time Stamp Counter), яка повертає час у тактах, що пройшли з останнього запуску процесора. Для певності ШПЗ часто викликає дану інструкцію двічі і порівнює отримані дані з тими узагальненими, які можна отримати на фізичному комп’ютері. Якщо різниця велика, це може бути індикатором віртуалізації. До небезпечних з точки зору виявлення також належать інструкції CPUID, RDMSR, WRMSR, RDPMC, RDRAND і т.д. Такі інструкції «неправильно встановлюють прапори стану, помилково змінюють пам’ять, неправильно генерують винятки або неправильно приймаються чи відхиляються емулятором центрального процесору [18, ст.5].

Затримка в часі виконання може бути викликана не лише вище описаними причинами, процес налагоджування також «супроводжується штрафами по часу виконання, які можуть бути виміряні зловмисним програмним забезпеченням для виявлення наявності пісочниці. Пісочниці намагаються запобігти цьому, підробляючи час. Однак зловмисне програмне забезпечення може обійти це, включивши зовнішні джерела часу, такі як NTP або позначки часу, включені в запити HTTP» [20, ст.4].

1.8 Непрямі докази (anti-debugging)

За винятком прямих та у більшості своїй однозначних індикаторів наявності віртуалізації, які у англомовній літературі прийнято називати anti-vm, існують також anti-debugging техніки ухилення від виявлення. Іншою можливою назвою-синонімом може буде термін anti-sandboxing, через його більшу орієнтованість на динамічний аналіз, ніж терміну anti-virtualization. Ці показники можуть слугувати для ШПЗ як додатковий голос на користь прийняття рішення про віртуаліність цільового середовища, адже самі по собі вони не є чимось таким беззаперечним, як наприклад, у комбінації з позитивно спрацьованими іншими техніками. Ідеться мова про присутність у системі встановлених додаткових програм, що можуть бути використані для аналізу зловмисного коду чи активності: дебаггери, застосунки для захвату та аналізу мережевого трафіку, логів окремих служб чи системи. Деякі приклади таких систем включають в себе: «Wireshark.exe, Fiddler.exe, Procmon.exe/ Procmon64.exe, Procexp.exe/ Procexp64.exe, Sysmon.exe/ Sysmon64.exe, ProcessHacker.exe, OllyDbg.exe, ImmunityDebugger.exe, Windbgx86.exe/ Windbgx64.exe, x32dbg.exe/ x64dbg.exe, Python.exe» [17].

Перевіряти наявність цих та подібних служб ШПЗ може подібно до того, як перевіряє список процесів на ті, що пов’язані безпосередньо з ВМ, присутність дебаггера також простіше за все виявити, додавши до коду ШПЗ виклик функції IsDebuggerPresent(), хоча вона і легко обходиться фахівцями. Більш витончений підхід включає в себе обфускацію зловмисного коду, заплутану структуру програми, пошук точок зупинки, які неодмінно встановлюються під час налагодження тощо. «Щоб додати програмну точку зупину, налагоджувач замінює байт коду операції за адресою точки зупину на байт 0xcc… Щоб додати апаратну точку зупину, налагоджувач зберігає адресу точки зупину в регістрі налагодження, а не змінює код налагоджувача» [21, ст.2]. Обидва типа точок можуть бути виявлені ШПЗ.

1.9 Дії у разі виявлення

Всі маркери потрапляння у віртуальне середовище додаються у ШПЗ не просто так, тобто не лише для того, щоб просто не виконуватись під потенційним наглядом. Якби після виконання своїх перевірок і отримання позитивного результату ШПЗ просто завершувало роботу, це майже б не заважало у його дослідженні, тому зловмисники вигадують все більш і більш витонченні реакції. У загальному є 3 типи таких реакцій [13, ст.3]:

1. Програма раптово завершується, коли виявляє, що її досліджують. Однак цей варіант не рекомендується, оскільки він може викликати підозру.
2. Програма раптово завершується і відображається повідомлення про помилку, пов'язане з відсутністю модуля або пошкодженим виконуваним файлом, щоб уникнути підозр.
3. Програма виконує лише доброякісні операції, щоб бути класифікованою як нешкідливий файл.

Деякі ШПЗ діють ще обережніше: на цільову машину потрапляє лише маленький файл, який виконує всі необхідні перевірки і тільки після того, як впевниться що запуск є безпечним, завантажує інші потрібні собі зловмисні частини. Щодо ситуації, коли ШПЗ визначає, що його дослідження йде у дебаггері, а отже є небезпека потрапити під лупу фахівця зі зворотної розробки, то «як тільки зловмисне програмне забезпечення усвідомлює, що воно виконується всередині налагоджувача, воно намагатиметься відхилитися від свого звичайного шляху виконання коду або перервати/збити налагоджувач» [22, ст.11].

Висновки до розділу 1

У першому розділі були розглянуті та згруповані за видами техніки ухилення та наведені деякі їх докладні приклади, з посиланнями на реально існуючі зразки ШПЗ. Індикатором віртуалізації може стати будь-який компонент ВМ: ключі реєстру, список процесів, назви файлів, налаштування мережі, структури пам’яті, апаратні засоби, активності користувача, певні програми аналізу тощо. Автори ШПЗ створюють його таким чином, аби увесь зловмисний функціонал розпочинав активні дії лише у разі успішних перевірок на справжність цільового середовища, інакше зупиняють зараження, маскуються під безпечні додатки, змінюють логіку своїх проектів таким чином, щоб шлях виконання був дуже заплутаним та складним для статичного аналізу, впроваджують затримки в активізації, додають такі умови для початку роботи, які можуть справдитись лише у реальній машині. Розвиток технологій ШПЗ протистоїть технікам маскування віртуалізації, оскільки віртуалізація є складовою частиною багатьох методів захисту, отже потрібне дослідження і методів маскування.

Індикатори ж віртуалізації, яким був присвячений перший розділ, умовно можна поділити на 2 типи: активні та пасивні. До пасивних належать усі ті індикатори, які є частиною системи, до активних – ті, які з’являються через активність користувача.

2 ТЕХНОЛОГІЇ МАСКУВАННЯ ВІРТУАЛІЗАЦІЇ

Усі техніки, описані в Розділі 1, добре відомі фахівцям з безпеки, тому зрозуміло, що під час створення дослідного середовища для ШПЗ вони враховують їх та придумують протидії. Розробники віртуальних машин теж не стоять осторонь та впроваджують в свої продукти деякі рішення, що допомагають маскуватись від виявлення віртуалізації, але оскільки цільовою задачею популярних засобів віртуалізації не є аналіз ШПЗ, більшість їх спроб легко обходяться нетиповим зловмисним ПЗ.

Тим не менш, існує безліч прийомів та дій, які можуть бути виконані під час налаштування дослідного середовища вручну або з застосуванням спеціальних утиліт для того, аби спробувати «заховати» усі індикатори, що можуть вказати на віртуальність середовища. Звісно, складнощів у таких запобіжних засобах не уникнути: такі дії з ВМ можуть наштовхнутися на велику кількість перешкод, які накладає сама реалізація ОС або ВМ.

Наприклад, відомо що можливо перехопити запит зловмисного характеру та підмінити його результат з такого, що однозначно вказує на віртуалізацію, на такий, що є характерним для реальної машини. Для цього, дані, що повертаються генеруються випадковим чином у певних розумних межах (наприклад температура процесора має бути реально можливою (не більше 90˚), назви файлів та вміст ключів реєстру не мають містити у собі інформацію, що викриває пісочницю) або замінюються на прописані вручну користувачем значення. При цьому проблема виникає у розумінні того, що не всі запити, які виглядають підозріло, виконуються ШПЗ, такі самі дії може робити і система сама, тому ніяк не вийде просто заблокувати усі дії, які здаються зловмисними, їх ще треба якось фільтрувати.

Інша проблема полягає у існуванні можливості, яка виникає через деякі системні вразливості, провести певну операцію від імені іншого процесу (користувацького чи системного), тобто виконати так званий Process Injection. Якби не це, то не було би ніяких складностей у тому, щоб відрізнити шкідливий

процес від легітимного. Згадана атака Process Injection являє собою як визначає цей термін Mitre ATT&CK Framework: «метод виконання довільного коду в адресному просторі окремого активного процесу» [23]. Окрім таких вражаючих результатів як наприклад підвищення привілеїв і виконання шкідливого коду від імені адміністратора, атака ця несе куди більшу, хоча й не зовсім очевидну загрозу: навіть при виконанні процесу від імені користувача з найменш можливими правами усе одно відбувається надійне маскування дій, направлених на виявлення віртуалізації. Таким чином, захист від атаки Process Injection є також хорошим способом завадити ШПЗ заховати себе від фахівця з безпеки. Інші способи протидії розглянуті далі.

2.1 Запобігання зчитуванню викриваючих рядків

Найпростіший спосіб захиститись від виявлення у системі компрометуючого рядка – тобто пасивного ІВ– знайти всі такі докази та видалити або змінити їх, проте якщо деякі файли з підозрілими іменами можна перейменувати, то у більшості випадків зміни у ключах реєстру призведуть до непередбачуваних наслідків для системи, тому неможливо просто виключити з реєстру всі строки, які містять назву середовища віртуалізації чи іншу загальновідому як маркер віртуалізації інформацію. Здається, не лишається інакшого виходу, як перехоплювати усі звернення до реєстру, що повертає до проблеми з атакою Process Injection, коли таке звернення може надходити від легітимного, але «зараженого» процесу та до проблеми фільтрації запитів. Проте, фахівці з безпеки знайшли простий, проте не зовсім очевидний, вихід з цієї ситуації: якщо ШПЗ має шукати назви певних ВМ, то десь в його коді ці назви мають зберігатись, що можна легко перевірити при статичному аналізі підозрілого файлу. Складніша процедура потрібна якщо ШПЗ завантажує необхідні йому частини коду вже після запуску на комп’ютері, тобто у файлі, що потрапляє до дослідників може й не бути той частини, що містить перевірки та виклики з потрібним рядком. На щастя, сама ідея перевірки на відсутність

віртуалізації як первинний крок у алгоритмі дії ШПЗ виключає таку можливість у більшості випадків. Проте аби виключити і такі можливості, можна виконувати своєрідний динамічний аналіз підозрілої програми, перевіряючи аргументи функцій, що вона викликає на присутність в них підозрілих рядків.

Тобто методом, який може допомогти фахівцю з безпеки не дати ШПЗ зчитати з реєстру рядок, що містить назву відомого засобу віртуалізації, є перехоплення функцій (термін hooking у англомовній літературі) з перевіркою їх аргументів та повернення «безпечної» відповіді. Перехоплення є технікою «за допомогою якої потік керування виконанням відхиляється до функції-трампліна з довільним кодом» [24, ст.28]. Під функцією-трампліном розуміють допоміжну функцію, яка виконає фрагмент коду, який було перезаписано при стрибку з головної функції до тої, де відбувається підміна параметрів, а також забезпечить продовження виконання перехопленої функції з того самого місця, після якого був здійснений «стрибок» (асемблерна інструкція JMP) [25]. Важливим є те, що ШПЗ не тільки може виявити такі дії, а й успішно використовувати ті самі методи для виявлення технік по його аналізу, проте якщо зловмисний код не має таких складних механізмів, для фахівця з безпеки він не стане проблемою. Проте треба розуміти, що ШПЗ також може побачити ознаки перехоплення, тому краще ухилятися від перехоплення стандартних функцій Віндовс, натомість зосередившись на перехопленні низькорівневих дій, які передбачає їх реалізація.

Так, наприклад, ШПЗ робить запит-зчитування ключів у директорії HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ і очікує знайти там значення «vmware», у той час цей крок легко виявити під час динамічного аналізу (хоча наявність аналізуючих програм може виявитись ще одним індикатором для ШПЗ), якщо виконати підозрілу програму покроково за допомогою дебагеру чи додати у систему власний скріпт, який буде чекати таке звернення. Тобто на запит ШПЗ отримає значення, яке не визве підозри або відповідь «не знайдено».

Такий підхід можливий не лише для сутностей реєстру, а також для:

1. Файлів або директорій, що мають компрометуючі назви;
2. Процесів/програм з назвами, що пов’язані з ВМ;
3. Назв пристроїв вводу/виводу, що підключаються до комп’ютера або апаратних засобів, які емалюються ВМ.

Протидія виявленню усіх цих категорій не має принципової різниці у реалізації та проводиться вище описаним способом.

Треба зауважити, що такий спосіб все рівно ніяк не вбереже від ШПЗ, яке використало вразливість системи та виконало свою перевірку від імені деякого користувацького або системного процесу, адже ми не можемо на всі звернення до реєстру, що наприклад містять рядок “VMware” відповідати «не знайдено», адже таку операцію може виконувати сама ВМ для забезпечення власних потреб, як для прикладу – перевірки факту встановлення потрібних модулей.

2.2 Протидія аналізу активності та наповненості середовища

ШПЗ, яке перевіряє присутність реального користувача у системі діє на двох напрямках: пасивному та активному. Під активним маються на увазі дії, які можуть проводитись користувачем у режимі реального часу (натискання миші чи клавіатури), під пасивними – сліди використання комп’ютера протягом деякого часу (кількість запитів у браузері, кількість файлів тощо).

Проти пасивного напрямку перевірки активності користувача треба:

- Створити не менше 10 файлів з назвами, що не будуть містити жодних натяків на ВМ;
- Зімітувати браузерну активність провівши не менше 10 запитів;
- Завантажити на комп’ютер стандартні не підозрілі програми, такі як пакет програм Microsoft Office, один із популярних браузерів, додатки соціальних мереж, Notepad++ тощо;
- Впевнитись, що розширення екрану є стандартним для комп’ютера, а не ВМ (встановити розширення 1920×1080 або 1366×768 або 1536×864);

- НЕ встановлювати VMware Tools.

Проти активного напрямку:

- Тримати відкритими не менше 3 вікон, причому їх назви не повинні мати натяків на ВМ;
- Мати не менше 3-5 активних процесів;
- Імітувати рухи мишки та натиски клавіш;
- Проводити під час аналізу популярні події, які вкладаються в умови логічних бомб, наприклад створювати та видаляти файли, відкривати\згортати вікна тощо.

Більшість пунктів є простими для виконання, проте з процесами та активністю миші пов’язані певні складнощі. При перевірці списку процесів можливе виявлення процесів, пов’язаних з ВМ чи з програмами аналізу, такі небажані результати можливо виключити з пошуку ШПЗ за допомогою того ж самого перехвату функції або просто завершити їх, якщо їх значимість для системи не є критично. Також можливо видалити інформацію про процеси «підозрілих» динамічних бібліотек з Блоку Оточення Процесів (PEB), але в окремих випадках це може зламати роботу системи.

Рухи миші можуть перехоплюватись ШПЗ, вивчатись на кількість натисків за одиницю часу, на адекватну швидкість руху, іноді навіть на поведінку, схильну для людини. На щастя, існує багато програм, які здатні емулювати діяльність миші, наприклад: AutoIt (емуляція автоматичної роботи з програмами, підтримка подій миші та клавіатури), AutoHotkey (моделювання повторювальних дій з програмами), LoadRunner (програма для тестування, але містить модуль для програмування натискань миші). Всі ці інструменти первинно не розраховані на введення ШПЗ в оману, але є достаньо гнучкими, аби їх можна було запрограмувати і на рандомізацію натисків, і на реалістичні траєкторії та швидкості.

Не можна забувати, що ШПЗ має в арсеналі способи, які подолати ВМ надзвичайно важко: використання CAPCHA та несправжніх кнопок, на які людина спробувала би натиснути, на відміну від пісочниці, та кнопок з

нетиповою назвою (інші варіанти замість “продовжити”, “встановити”). Аби протидіяти таким технікам, потрібно: для фальшивих кнопок «якщо текст “Встановити” пов’язаний із “статичним” елементом керування (а не з “кнопкою”), це може вказувати на те, що застосовується техніка ухилення» [26].

Найкращим методом уникнути більшості проблем з наповненням середовища є використання у якості дослідного середовища копії системи, яка реально експлуатується, адже це дасть не лише необхідне для уникнення підозр наповнення, але й розуміння недоліків налаштувань, які роблять атаки успішними.

2.3 Протидія виявленню через технічні характеристики

Аби зловмисна програма не запідозрила віртуальність зараженої системи, потрібно при налаштуваннях дослідного середовища враховувати певні пункти:

1. Виділяти для ВМ достатню кількість ядер процесора: 2, але краще 4 або більше, це не тільки допоможе успішно пройти перевірку на підозріло малу кількість ядер, а й дасть гарні обчислювані можливості для аналізу ШПЗ;
2. Виділяти достатню кількість оперативної (більше 1Гб) та внутрішньої пам’яті (більше 80Гб);
3. Підключати до ВМ аудіо-девайси, камеру.

При перевірці інших фізичних характеристик – пасивних ІВ, які можуть бути присутні лише на реальній машині, таких як наприклад температура термальних зон процесора, можливе використання все ж того механізму перехоплення функції та підміни даних, що повертаються. Деякі дані можливо просто змінити у системі без подальших проблем для функціоналу, як наприклад модифікація визначених бітів в певному регістрі CPUID може завадити ШПЗ визначити факт віртуалізації. Так, виклик інструкції CPUID зі значенням регістру EAX=1 показує інформацію про процесор, а саме «31-й біт

ECX на фізичній машині буде дорівнювати 0. На гостьовій віртуальній машині він дорівнюватиме 1». [27]. На прикладі VMware Workstation протидія надзвичайно проста – потрібно лише додати наступний рядок до конфігураційного файлу ВМ з розширенням “.vmx”: cpuid.1.ecx=”0---:----:----:----:----:----:----:----“. Як видно, 31 біт тут встановлено значенням “0” – саме це потрібно, аби перевірка ШПЗ провалилася. Інші біти довільні.

Друга приклад перевірки CPUID для ухилення передбачає виклик зі значенням EAX=40000000 та зчитування результату, яке у випадку ВМ поверне визначений гіпервізором рядок, для VMware Workstation це буде “VMwareVMware”. Перевірити це можна виконавши наведений на електронному ресурсі [28] код, спочатку на фізичній машині, потім на ВМ, сам код та два таких виклики продемонстровано на Рисунку 2.1:

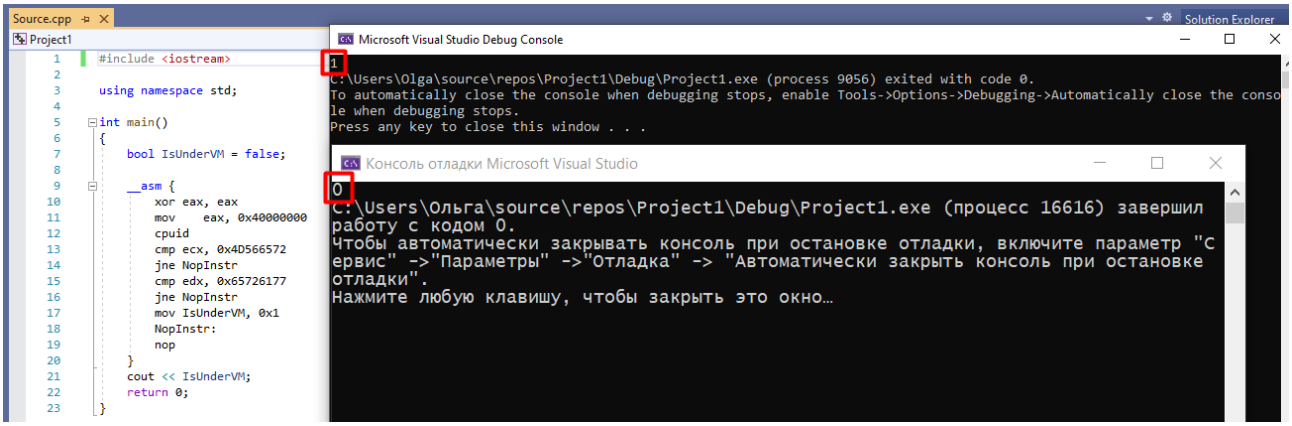

Рисунок 2.1 – Код перевірки, його виконання на ВМ (вікно терміналу зверху) та на фізичній машині (знизу)

Сам код робить надзвичайно прості дії: визначає EAX=0x40000000 та порівнює значення у регістрах EDX та ECX з частинками рядку “VMwareVMware”, а саме перший це «eraw», другий «MVer», причому рядок записаний у регістри молодшим бітом вперед. Як видно, перевірка на ВМ дає позитивний результат – рядок знайдено, на фізичній машині – негативний.

Протидія цій перевірці теж потребує змін у конфігураційному файлі ВМ з розширенням “.vmx”, у даному випадку приклад виконано на ВМ Windows 10 x64, тому файл, який потрібно змінити – “Windows 10 x64.vmx”. Все, що

потрібно змінити – значення згаданих регістрів, на довільні біти, як показано на Рисунку 2.2:

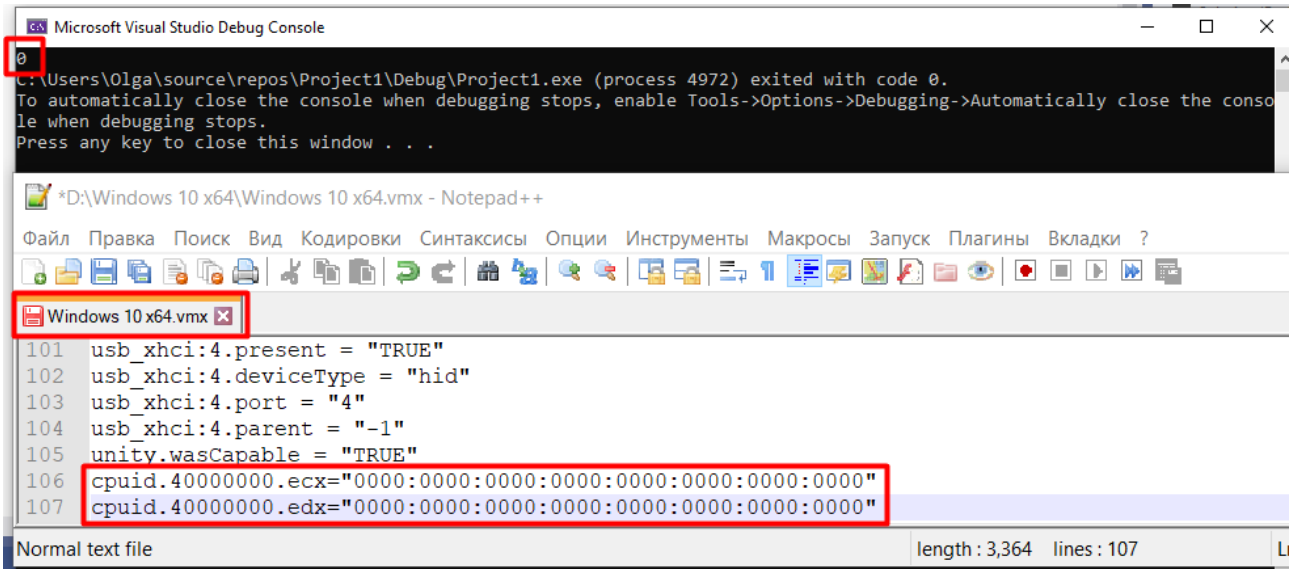

Рисунок 2.2 – Зміни у файлу конфігурації ВМ та результат

Після змін у конфігураційному файлі треба перезавантажити ВМ та знову запустити код, тепер він має видавати негативний результат – віртуалізацію не визначено.

2.4 Запобігання виявленню різниці у часі виконання

Затримки в часі виконання виникають через специфіку деяких гіпервізорів ВМ: вони виконують звернення до процесора, пам'яті тощо не напряму, а з використанням функцій хостової ОС. «Основна відмінність гіпервізорів типу 1 від типу 2 полягає в тому, що тип 1 працює на фізичному обладнанні, а тип 2 — поверх операційної системи» [29]. Таким чином, найпростіший спосіб уникнути затримок – використовувати ВМ, гіпервізори яких належать до 1 типу, хоча вони і більш вибагливі до ресурсів та не такі популярні, як гіпервізори 2 типу, які ще називають хостовими, через їх «надбудову» над хостовою ОС, до яких належать і такі поширені рішення віртуалізації як VMware Workstation та VirtualBox. До першого ж типу відносяться Hyper-V, vSphere та Xen hypervisor.

Проте існують і рішення цієї проблеми, які не потребують дорогих апаратних технологій віртуалізації. Насамперед, проти згаданої у першому розділі інструкції RDTSC (як і проти будь-якої іншої) можна «реалізувати перехоплення та зробити RDTSC привілейованою інструкцією, яку можна викликати лише в режимі ядра. Виклик “перехопленого” RDTSC в режимі користувача призводить до виконання обробника, який може повернути будь-яке потрібне значення» [30]. Можливо також змінювати значення Time Stamp Counter (TSC) вручну, аби ШПЗ думало що пройшло не так багато часу з запуску операції, або навпаки збільшувати, у випадку коли є необхідність уникнути технологій відкладеного виконання. Сучасні ВМ іноді вміють підроблювати таке значення самостійно, без звернень до хостової ОС.

2.5 Запобігання виявленню інструментів аналізу

За можливості треба або просто відмовитись від використання відомих програм для аналізу ШПЗ у ВМ, використовуючи власні реалізації або ж замаскувати існуючі. Для аналізаторів трафіку на кшталт Wireshark достатньо просто запустити “підслуховуючий агент” на хостовій машині або на іншій ВМ в одній мережі з цільовою. Саме через таку легкість обходу розробники ШПЗ і не витрачають час на розробку анти-мережевих аналізаторів. Зовсім інакша справа з дебагерами: зловмисний код має життєвий інтерес їх викривати, тому фахівець з безпеки має врахувати особливості роботи налагоджувачів та протидіяти їх виявленню. Так, програмні точки зупину можуть бути визначені ШПЗ пошуком байту 0xcc у своєму коді, отже досліднику потрібно перед установкою байту скопіювати оригінальний байт зловмисного коду та його адресу і коли ШПЗ перевірить, надіслати у відповідь копію з незміненим значенням.

Більшість ШПЗ не будуть просто використовувати вбудовані функції Віндовс через легкість їх перехоплення, натомість вони можуть «замість цього виконувати прямі виклики…Внутрішні структури, такі як PEB набагато важче

підробити — деякі зміни можуть навіть зламати частини системи» [24, ст.5]. Проте так само як зловмисники можуть перевірити флаг структури PEB, який відповідає для налагоджування, фахівці з безпеки можуть перевірити час доступу до цього флагу і таким чином зрозуміти, що ШПЗ виконувало перевірку. Процитована вище стаття 2017 року розглядає механізм згаданої у роботі winapi-функції IsDebuggerPresent(), зауважує що її реалізація в ОС Віндовс представляє собою по суті той самий прямий доступ до флагу налагоджування, отже розуміючи яку ділянку пам'яті потрібно прочитати, можна передбачити дії ШПЗ і виявивши таке звернення у досліджуваному коді, фахівець з безпеки може виявити цю техніку. Схожим методом долаються і інші, більш специфічні перевірки.

2.6 Зміна характеристик, які підлягають модифікації

Кожна ВМ встановлюється з налаштуваннями по замовчуванню. Після встановлення можливо змінити деякі з параметрів, причому зміна одних передбачена системою і не потребує особливих зусиль, а з іншими не все так просто. Ім'я комп'ютера – найпростіше, що можна і обов'язково треба змінити, з автоматично згенерованого на щось більш «людське», але ні в якому разу не використовувати пов'язані з аналізом або ВМ назви. Зробити це можна у налаштуваннях як показано на Рисунку 2.3:

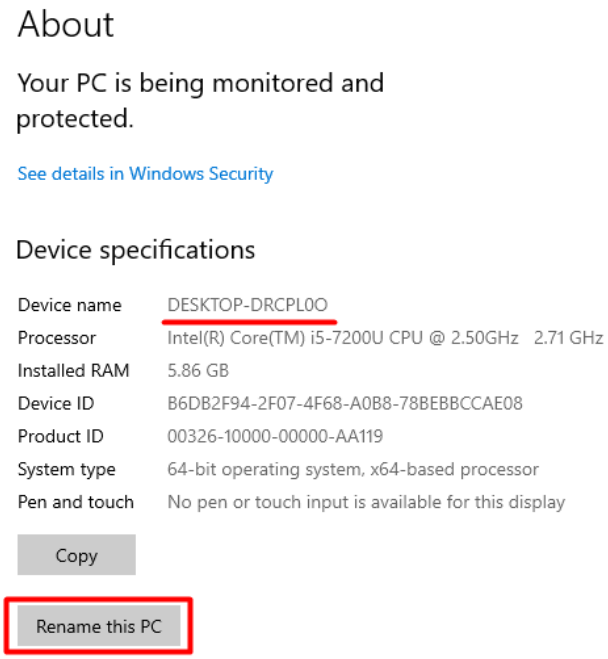

Рисунок 2.3 – Зміна імені комп'ютера

Ім'я користувача змінюється так само, як і на звичайній машині, потрібно уникати типових імен для аналізаторів, як "admin", "sandbox", "vmware", "malware". Також легко встановити свої MAC-адреси на мережеві адаптери, замінивши дефолтні адреси, так наприклад у вже згаданому "Windows 10 x64.vmx"міститься рядок: «ethernet0.generatedAddress = "00:0c:29:58:7f:cd"», перші 3 біти якого є стандартними для VMware Workstation, можливо провести модифікацію прямо у файлі.

Можливо також модифікувати деякі значення в реєстрі, насамперед такі, які особливим чином ні на що не впливають, проте доволі часто перевіряються ШПЗ: назви апаратного забезпечення, імена виробників, назви програм для полегшення роботи з ВМ тощо. Так, у VMware Workstation існують VMwareTools, які можна і не встановлювати, проте за їх присутності вони легко викривають віртуалізацію. Зміна деяких значень реєстру може бути не тільки забороненою, але й небезпечною для системи, адже деякі з ключів критично необхідні для роботи ВМ. Дізнатись, які зміни не призведуть до блокування нормальної функціональності системи можна у офіційній документації або на власному досвіді, враховуючи що механізм снапшотів це дозволяє. Необов'язково міняти всі значення, можна змінити назву папки, прописати

важливим програмам новий шлях, якщо це потрібно, і це не дасть ШПЗ прочитати реєстр, адже у його коді буде шлях, який тепер не існує. На Рисунку 2.4 наведено деякі приклади модифікацій в реєстрі, де початкова версія ліворуч, а змінена – праворуч:

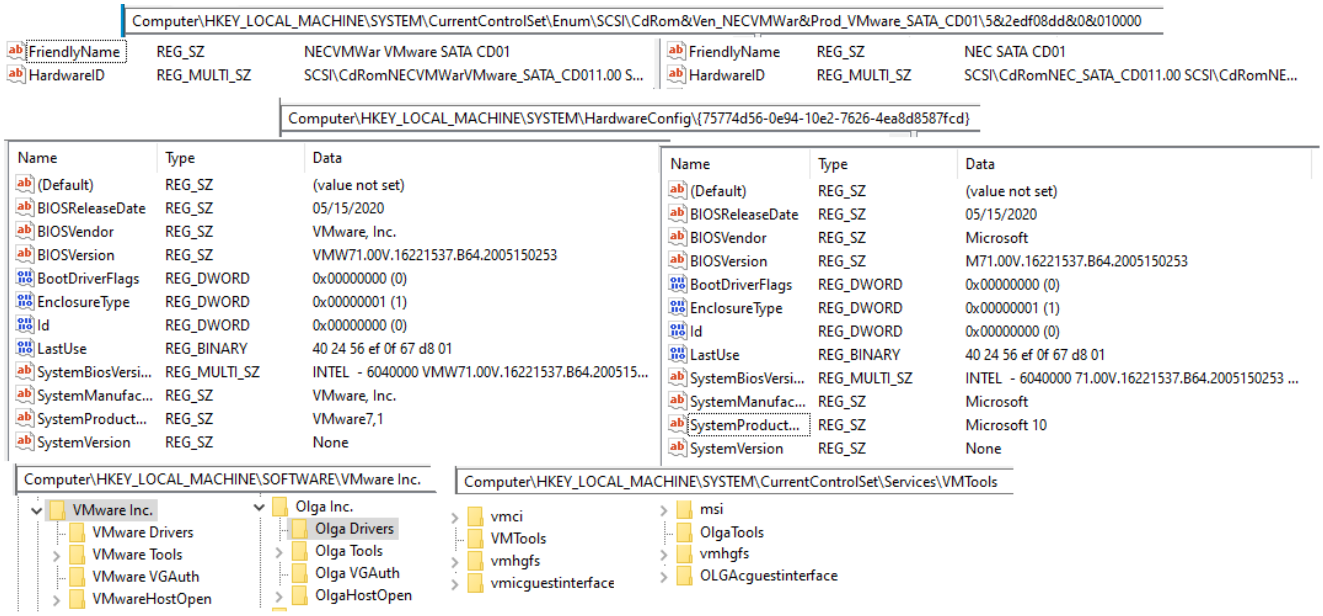

Рисунок 2.4 – Зміни в реєстрі

Висновки до розділу 2

В розділі 2 проведений порівняльний аналіз методів захисту від виявлення віртуалізації. За допомогою введеного в першому розділу роботи поняттю ІВ можна визначити єдиний підхід до аналізу методів уникнення віртуалізації. Практично він полягає в моніторингу всіх ІВ системи на досить низькому рівні та комплексний підхід з увагою до артефактів, що можуть викривати віртуалізацію. Це надзвичайне навантаження на ресурси системи, тому потрібно зробити захист від технік ухилення адаптивним до них, що означає необхідність чіткого розуміння тенденцій та способів розробки ШПЗ. Розділ 3 займається проблемами статистики та розробкою методології приховування ІВ. У більшості випадків потрібне створення власних інструментів і для аналізу, і для приховування ВМ.

3 МЕТОДИКА ПРОТИДІЇ ВИЯВЛЕННЮ ВІРТУАЛІЗАЦІЇ НА БАЗІ ДЕСКПОТНОЇ ВЕРСІЇ ОС WINDOWS

Техніки ухилення від виявлення віртуалізаці популярні серед розробників ШПЗ з самого виникнення ідеї зловмисної програми, але ці техніки включають в себе не тільки техніки анти-віртуалізації, анти-аналізу чи анти-налагодження, це також усі види заплутування аналізаторів, ускладнення програм для реверс-досліджень, ін’єкції у легітимні процеси та ще багато чого. Не дарма у проекті ATT&CK від MITRE розділу тактики «Ухилень з метою захисту» (Defense Evasion) присвячена найбільша кількість сторінок з техніками – для ШПЗ ефективне уміння приховувати себе продовжує його життя та покращує результати зловмисної роботи. Так, однією з технік у досягненні тактики приховування є Virtualization/Sandbox Evasion (T1497) [31], що містить три під-техніки: Системні Перевірки (System Checks), Перевірки, засновані на активності Юзера (User Activity Based Checks), Перевірки, засновані на Часі (Time Based Evasion). Для кожної з під-технік на сайті є деяка кількість прикладів їх використання реальними ШПЗ. Статистика використання різних технік ухилення досліджується автором у першій частині Розділу 3.

Щодо питання популярності саме технік анти-віртуалізації, у дослідників думки різняться: так, існують два ґрунтовні статистичні аналізи кінця 2021 року, де за результатами першого [32] виявлено після перевірки 180 тисяч зловмисних зразків спадання тенденції (майже у двічі) на використання саме технік анти-віртуалізації, порівняно з іншими техніками ухилення. Друге ж дослідження [33] на 45 тисяч зразків, навпаки показує, що кількість технік проти віртуалізації залишилась плюс-мінус стабільною протягом останньої декади, навіть трохи збільшилась. Така різниця може звісно пояснюватися і різними виборками дослідних ШПЗ, але варто й зауважити, що чіткого переліку технік, що вважаються техніками анти-віртуалізації, не має, що може давати таке розходження у висновкам щодо тенденцій. Яке б з досліджень не було ближче до правди, технік проти віртуалізації використовується достатня

кількість, аби виникала потреба їх докладно вивчати. Більше того, необхідною є розробка чіткої та повної класифікації груп технік ухилення, аби не допускати таких непорозумінь.

3.1 Статистика на базі ATT&CK від MITRE

Отже, у техніки Ухилень від Віртуалізації T1497 існує три підтехніки, проте вони не дають повного і чіткого розподілу кожної можливої техніки лише на 3 ці групи, також на сайті є приклади ШПЗ, які використовують так звані «Загальні Ухилення від Віртуалізації» (General Virtualization/Sandbox Evasion), тобто до цієї категорії і підпадають ті ухилення, що не можна класифікувати за трьома основними. У Розділі 1 даної роботи автором по суті вже було запропоновано класифікацію технік анти-віртуалізації, оскільки кожен вид описувався в окремому підрозділі. Таким чином у роботі виділено 7 видів [2]:

1. Перевірка ключів реєстру
2. Перевірка фізичних характеристик
3. Перевірка артефактів
4. Час виконання операцій
5. Спрацювання тригеру-умови
6. Перевірка на активність користувача
7. Анти-аналіз (але не анти-дебаг)

Окремого чіткого визначення потребують три з них. Для початку техніки «Спрацювання тригеру-умови» та «Перевірка на активність користувача» можуть у деяких випадках відрізнятись лише ідейно, а не фактично. Так, наприклад існує ШПЗ, яке розповсюджується через офісний документ з підтримкою макросів. Зловмисні модулі до нього підвантажуються лише за умови, що на картинку у документі було натиснуто двічі, інакше нічого не відбувається. Постає питання: Чи є це перевіркою на активність користувача чи очікуванням певної події (подвійного натиску лівої кнопки миші). Відповідь на це питання, як видно, залежить від того, як його сприймати, тому аби не

плутати види технік, потрібно зазначити, що різниця між ними у тому, що у випадку «Перевірки на активність користувача» очікується просто активність, можливо хаотична – підтвердження наявності людини-користувача, а у випадку «Спрацювання тригеру-умови» – певна визначена подія як от натискання на картинку, вікно тощо. Також питання може викликати техніка «Анти-аналіз (але не анти-дебаг)» через те, що вона не була наведена у Розділі 1, проте являє собою набір технік, які не мають певних спільних рис, крім тих, що їх не можна чітко віднести до жодної іншої категорії, у тому числі до анти-налагоджування, адже цей процес майже завжди є складовою аналізу. Натомість, до сьомого типу технік ухилення можуть належати перевірки на наявність інших механізмів аналізу, крім дебагерів, хитрощі, до яких ШПЗ вдається аби зрозуміти, що воно під наглядом як наприклад перевірка імені, під яким воно збережено у цільовій системі, адже часто фахівці називають файл його хеш-значенням, а також спроби виявити перехоплення системних функцій, які часто впроваджуються для більшого контролю над ВМ.

Таким чином, для зручності представлення, результати подані у Таблиці А.1, яка знаходиться у Додатку А. У ній підтехніки з сайту ATT&CK від MITRE позначені своїми абревіатурами: SC, UABC, TBE, G (для general атак) а виділені категорії відповідно до наведеного вище порядку скорочено так: реєстр, фіз., артеф., час вик., тригер, актив., анти-ан. Для кожного ШПЗ у таблиці Додатку А окрім під-технік та технік наведено також його номер у базі ATT&CK, назву, цільову групу, рік виявлення та короткий опис суті ухилення. Щодо цільових груп, то класи наведного ШПЗ позначено цифрами у відповідності до скорочень, наведених на початку Додатка А.

Щодо статистики, яку виражають зведені дані технік ухилення, то більше половини (53%) з них становлять такі, що стосуються підтехніки Системних перевірок, тоді як найменша група – перевірки на активність користувача. Друге місце за популярністю займають перевірки, що базуються на часі, як це можна побачити на Рисунку 3.1:

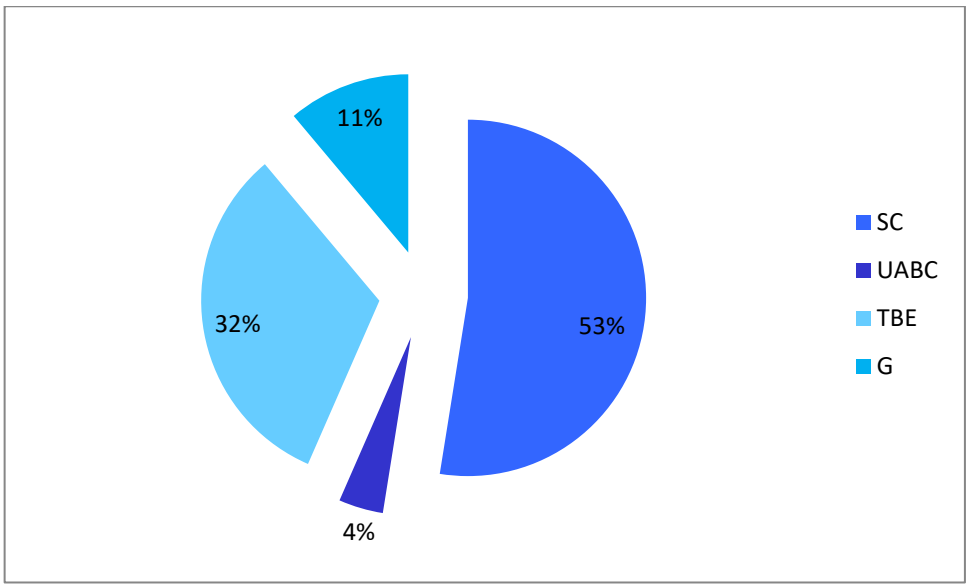

Рисунок 3.1 – Розподіл між підтехніками за даними з ATT&CK від MITRE

Якщо проводити аналіз саме по видам, сім яких було виділено вище у даній роботі, то різко виділяються 3 домінантні напрямки: Спрацювання тригеру-умови, Перевірка артефактів та Перевірка фізичних характеристик, як це видно з Рисунку 3.2:

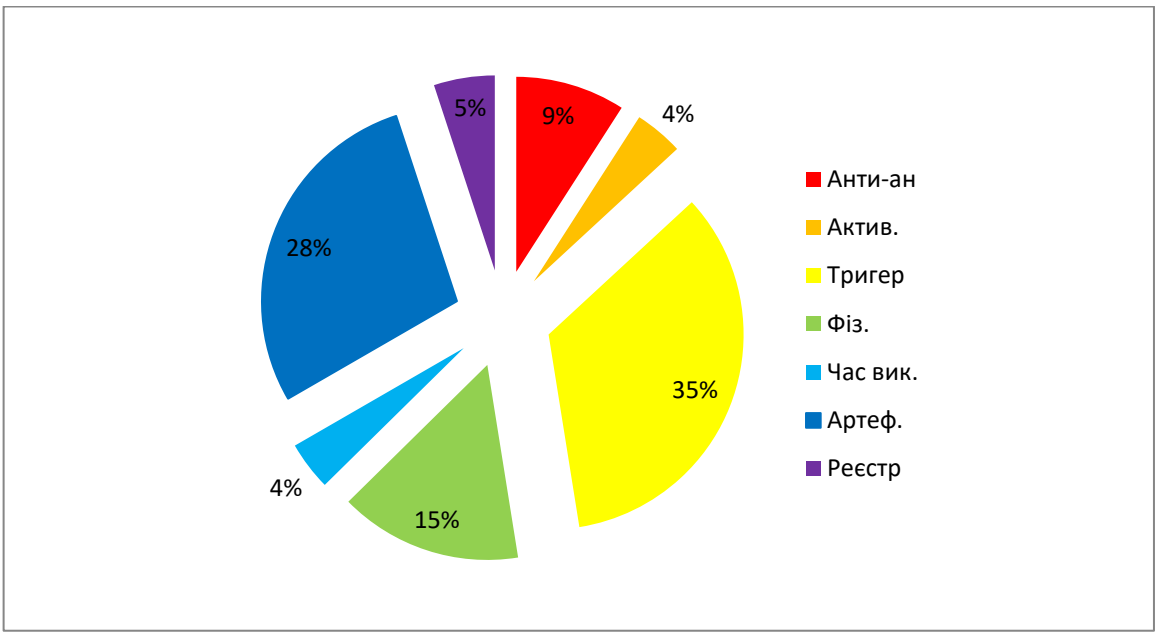

Рисунок 3.2 – Розподіл між типами ухилень за даними з ATT&CK від MITRE

Ці домінуючі напрямки прямо передбачуються результатами узагальненої статистики на Рисунку 3.1, адже всі вони належать якраз до Системних перевірок, якщо дотримуватися класифікації MITRE. Їх частка у загальній

кількості становить 78%, і хоча інші техніки становлять доволі вагомі 22%, їх кількість та різноманітність набагато більша, тому може бути недоцільним захищатися від них, адже це буде потребувати більших ресурсів за меншої ефективності, порівняно з першими трьома.

Цікавим є також проаналізувати, які саме дії проводить ШПЗ, аби виявити віртуалізацію. Так, для підтехніки «Перевірки, засновані на Часі» такою дією є майже завжди (у 23 випадках з 32) є певна затримка виконання. Більш оригінальний та елегантний спосіб – перевірка чи відбувалося прискорення системного часу – виявлено лише у 4 ШПЗ. Для категорії «Перевірки, засновані на активності Користувача» типовим є перевірка кількості натисків кнопок миші або реєстрування положення курсора, що є зовсім не дивним. Підтехніка «Системні Перевірки» представлена дуже різними техніками, проте спільна риса у них всіх одна: шукається рядок, що вказує на назву ВМ. Цей рядок шукається у реєстрі (Перевірка ключів реєстру), у специфікаціях апаратного обладнання (Перевірка фізичних характеристик), у назвах папок, файлів, програм, процесів, доменів, версії чи назві ОС (Перевірка артефактів). У перевірці артефактів теж були наявні перевірки MAC адрес, магічного значення VMXh. Фізичні ж характеристики зчитувалися для: материнських плат, bios-a, драйверів, жорстких дисків та звісно для процесора. Дійсно у двох з вивчених ШПЗ була наявна перевірка на температуру процесора та на можливість програвання звуків аудіо-картою, тому і такі функції не слід ігнорувати. До Анти-аналізу потрапили всього 9 технік, але вони заслуговують на увагу. Йдеться мова про перевірки імені, під яким запущено у системі ШПЗ, причому якщо ця назва є хешем, це однозначно вказує на певний процес аналізу, а також один зразок змінював своє розширення на щось не підозріле. Сюди ж відносяться ухилення щодо вищого рівня – виявлення та видалення перехоплень, які могли виконати дослідники та виклик системних функцій з неправильними параметрами, знову ж таки аби завадити перехопленню своїх зловмисних дій.

Оскільки у Додатку А у Таблиці А.1 також вказаний тип ШПЗ, не зайвим буде навести і статистику щодо цього аспекту: так, 27 з усіх зразків мали ознаки троянів різних видів, 19 виказали себе як програми-бекдори, а 10 – як крадіжники інформації. Як така, ця інформація показує деяку кореляцію між цілями, для яких було створене ШПЗ і тим, чи використовує воно техніки ухилення. Зрозуміло, що і для трояну, і для бекдору дійсно життєво необхідно знати, що він знаходиться у реальному середовищі. Що ж до того, у якому році було виявлено зразки, то дві третини з них «не старіші» п’яти років, що свідчить про те, що техніки ухилення активно використовуються і досі.

У цілому у базі ATT&CK від MITRE у розділі техніки «Virtualization/Sandbox Evasion» немає прикладу такого ухилення, яке б не було розглянуте у даній роботі, що говорить про повноту теоретичного огляду та свідчить про доцільність використання саме наведеної у Додатку А бази технік.

3.2 Техніки та протидії їм

Оскільки метою даної роботи є у тому числі створення єдиної повної методики щодо дій, які потрібно провести фахівцю з безпеки, який хоче збільшити кількість викритих зловмисних дій з боку підозрілих дослідних зразків шляхом посилення та модифікації власного віртуального середовища для аналізу на базі десктопної версії Windows, такі методики буде надано далі за сімома категоріями, визначеними раніше. Методика буде складена у вигляді семи окремих таблиць, по одній для кожної категорії, причому це буде зручно для швидкого та зручного користування. Хоча за результатами з минулого підрозділу 3.1 стало зрозуміло що принаймні серед загально відомих ШПЗ переважають перевірки націлені на саму цільову систему, не можна зовсім відкидати інші напрямки, адже якщо ресурси, часові, грошові та персоналу це дозволяють, ВМ для дослідів має бути підготована якнайкраще.

Не всі категорії технік ухилення є об’ємним, наприклад Перевірка ключів реєстру, наведена у Таблиці 3.1, складається лише з однієї техніки:

Таблиця 3.1 – Перевірка ключів реєстру

Частина системи	Техніка ухилення	Протидія
Реєстр	Пошук значень, що вказують на ВМ у ключах та їх значеннях	Зміна значень, які можливо; моніторинг доступу до реєстру запитів з рядками, пов'язаними з ВМ, повернення безпечних значень; пошук у коді зразку імен ключів реєстру

Така стисліть пояснюється тим, з реєстром насправді можна провести дуже обмежену кількість дій – як правило, ШПЗ просто веде у ньому пошук рядку; основна складність у даній категорії – не всі ключі реєстру можуть бути змінені або видалені без порушення нормального функціонування системи, а моніторинг їх зчитування ускладнено тим, що через можливості ін’єкцій у легітимні процеси, деякі ШПЗ можуть виконувати їх непомітно.

Категорія, посвячена перевірці різноманітних апаратних властивостей, наведена у Таблиці 3.2 та містить доволі багато технік, насамперед найцікавішими є ті, що стосуються процесору:

Таблиця 3.2 – Перевірка фізичних характеристик

Частина системи	Техніка ухилення	Протидія
Апаратне забезпечення у цілому	Зчитування його параметрів з ключів у реєстрі	Змінювати значення виробника, назви девайса на такі, які притаманні реальним машинам
Пам'ять	Перевірка об'єму	Виділяти не менше 1Гб оперативної та 80Гб внутрішньої пам'яті
Айудіо-девайси	Чи можливо програти звуки, чи під'єднані вони	Підключати аудіо-девайси або перехоплювати перевірки і підтверджувати наявність
Камера	Чи під'єднана	Підключати камеру до ВМ або перехоплювати перевірки і підтверджувати наявність

Продовження Таблиці 3.2

Частина системи	Техніка ухилення	Протидія
Процесор	Температура термальних зон	Перехоплювати перевірки та повертати припустиме значення (від 30˚ до 90˚)
	Кількість ядер	Виділяти не менше 2 ядер процесора під ВМ, в ідеалі 4 і більше
	Перевірка 31 біту ecx регістру інструкції cpuid	Змінити значення біту з 1 на 0
	Виклик cpuid зі значенням eax=40000000	Заміна значень регістрів ecx, edx довільними байтами (окрім 31 біту)

Наступна – Таблиця 3.3 – являє собою перелік протидій перевіркам, націленим на пошук програмних артефактів, що свідчать про ВМ:

Таблиця 3.3 – Перевірка артефактів

Частина системи	Техніка ухилення	Протидія
Програми	Пошук таких, що мають відношення до ВМ	Не встановлювати/видаляти такі, без яких система може працювати (напр. VMtools), для інших перехоплювати такі перевірки
Динамічні бібліотеки, драйвери		
Ім'я комп'ютера, користувача, домену	Пошук стандартних значень або пов'язаних з ВМ	Змінити на довільне, яке не буде викривати ВМ
Мережні адаптери	Пошук стандартних для ВМ адаптерів	Перехоплення і поверення припустимої відповіді
	Пошук усіх адаптерів	Змінити назву тих адаптерів, які можливо
	Час перезавантаження адаптера	Поставити адаптер на автоперезавантаження через невеликий термін

Продовження Таблиці 3.3

Частина системи	Техніка ухилення	Протидія
MAC-адреси	Пошук стандарних значень	Змінити адреси, які згенеровані ВМ автоматично на довільні
Порт I\O	Перевірка магічного значення VMXh	Перехоплення інструкції IN або заміна значення
Файлова система	Перевірка імен папок і файлів на рядки, що вказують на ВМ	Змінити імена таких об'єктів
Інформація про ОС та bios	Пошук відомих значень айді або назв ВМ	Зміна значень де можливо у ключах реєстру, інакше перехоплення і повернення безпечних результатів

Наведені протидії часто включають в себе перехоплення функцій-перевірок, що і є основною складністю у захисті від технік даної категорії, через причини, наведені у пункті 2.2 даної роботи. Як видно, перевірки такого виду теж різноманітні, чого не сказати про перевірки Часу виконання операцій, наведені у Таблиці 3.4:

Таблиця 3.4 – Час виконання операцій

Частина системи	Техніка ухилення	Протидія
Інструкції, які призводять до виходу з ВМ	Виклик інструкції, замір часу виконання та перевірка виходу з ВМ	Перехоплення потоку виконання інструкцій та підміна результатів виконання без фактичної потреби передавати фізичному процесору дії
Інструкція RDTSC		Заборона виконання інструкції у режимі звичайного користувача, або дозвіл, проте з підміною повертаючих значень

Така стислість пояснюється тим, що перевірки цього класу доволі складні за своїм технічним описанням, тому наведена теоретична характеристика. Проблеми з реалізацією протидій ним також стосуються необхідності непомітного перехоплення.

Техніки, що очікують Спрацювання тригеру-умови, перелічені у Таблиці 3.5, в основному протидія ним включає в себе докладний динамічний аналіз зразків та певні модифікації середовища на програмування автоматичного виконання подій-тригерів:

Таблиця 3.5 – Спрацювання тригеру-умови

Частина системи	Техніка ухилення	Протидія
Системний час	Затримка часу виконання	Пошук затримок у коді і або пропуск їх через прискорення усіх системних часових відміток, або якщо час невеликий, переміщення зразку в чергу для аналізу доки час не вийде (прискорення можна виявити у разі використання зовнішнього джерела часу)
	Запуск у певний час\дату	
	Рандомізація часу запуску	
	Перевірка часу через зовнішні джерела	Створення несправжніх зовнішніх сервісів, підміна повертаючих значень через мережу
Вся система	Використання гальмівного коду	Не зупиняти перевірку, доки весь код не виконається; перевіряти чи використовуються далі у коді результати функцій, якщо ні, то пропускати складні обрахунки
Миша або клавіатура	Очікування натиску певних комбінацій клавіш	Кнопки, які містяться у зразках, мають бути натиснені; для документів з макросами усі картинки мають бути натиснуті двічі, документи прогортані до кінця
Вся система	Очікування подій, які вказують на реальність середовища	У повторювані дії мають бути винесені часті події, що відбуваються у реальних середовищах: створення, видалення, відкриття файлів, зміни у реєстрі, вікриття програм, у тому числі браузеру

Складність цієї категорії – у великому спектрі подій, що можуть очікуватись та неможливості передбачити без попереднього аналізу яку подію генерувати.

Найбільша категорія – Перевірка на активність користувача – наведена у Таблиці 3.6 і хоча у більшості випадків перевірка на активність не включає в себе такі складні технології, як аналіз переміщень курсору миші, така техніка існує і тому потребує уваги:

Таблиця 3.6 – Перевірка на активність користувача

Частина системи	Техніка ухилення	Протидія
Екран	Зчитування розширення	Встановлення популярного розширення (1920×1080 або 1366×768 або 1536×864)
Миша	Кількість натисків за проміжок часу	Встановлення скрипту, що генеруватиме активність АБО перехоплення таких перевірок і повеременення адекватних значень
	Зміна положення курсору за проміжок часу	
	Аналіз рухів (швидкість, траєкторії)	Прописувати логіку автоматизованого скрипта таким чином, аби вона імітувала людську поведінку з мишею
Клавітура	Кількість натисків за проміжок часу	Встановлення скрипту, що генеруватиме активність
Файлова система	Перевірка кількості останніх фалів	Створити хоча б 10 файлів
Вікна	Перевірка відкритих вікон	Тримати 3-5 активних вікон
Програми	Чи встановлені типові програми	Встновити офісні програми, браузер тощо
	Перевірка активних процесів	Запускати декілька програм, не пов'язаних з аналізом чи ВМ

Продовження Таблиці 3.6

Частина системи	Техніка ухилення	Протидія
Браузер	Перевірка історії пошуків	Провести хоча б 5 пошукових запитів
Інтерактивна	Потрібно натиснути на кнопку з нетиповим написом	Генерація натискання усіх об'єктів, що можна натиснути адекватного розміру
	Кнопки такого розміру\кольору, що людина їх не помічає	Не натискати кнопки менші за мінімальний допустимий розмір, порівнювати колір фону і самої кнопки на видиму різницю
	Фальшиві кнопки, які лише виглядають так, ніби їх можна натиснути	Пошук у коді назв, типово пов'язаних з кнопками, але які не є об'єктами, що можна натиснути, та їх натискання
	Перевірки на кшталт CAPCHA	Впровадження систем розпізнавання образів
Вся система	Перевірка часу перезавантаження системи	Зміна значення на реально можливе

Основний виклик – висока важкість імітування поведінки, ідентичної людській, проте для більшості зразків ідеальна симуляція і не потрібна – достатньо простих дій по наповненню та установки скриптів, що генеруватимуть активність миші.

Остання категорія – Анти-аналіз знаходиться у Таблиці 3.7:

Таблиця 3.7 – Анти-аналіз (але не анти-дебаг)

Частина системи	Техніка ухилення	Протидія
Програми	Перевірка програм аналізу коду, трафіка тощо	Замаскувати за можливості програми, створити власні реалізації, які не буде так легко виявити, для перехоплення мережевого трафіка використовувати хостові мережні інтерфейси

Продовження Таблиці 3.7

Частина системи	Техніка ухилення	Протидія
Файлова система	Перевірка назви зразка	Не давати дослідним зразкам назв, що містять типові рядки ("шпз", "вірус") або такі, що є хешем
	Використання нетипового розширення	Аналіз вмісту файлу та спроба запустити його, змінивши розришення
Системні виклики	Виклик функцій з помилковими параметрами, аби уникнути перехоплення	Валідація параметрів та пропуск помилкових викликів
	Видалення перехоплень	Встановлення перехоплень від імені суперкористувача
	Заміна стандартних функцій викликом їх функціональних інструкцій	Моніторинг не лише функцій, але й інструкцій, які вони використовують на підозрілі параметри

Протидій, що становлять найбільший виклик для фахівців – ті, що включають у себе механізм перехоплень, проте варто зазначити, що і створення програм моніторингу інструкцій, які виконуються всередині системних викликів за замовчуванням, потребує ґрунтовного розуміння усіх них та також дій, які буде виконувати ШПЗ, аби успішно їх виявляти. Не можна не згадати і про те, що такі заходи будуть вимагати серйозних ресурсів від ВМ для вирішення.

3.3 Практична імплементація індикаторів віртуалізації

Для перевірки наявності ІВ у цільовій системі автори ШПЗ пишуть програми, які це перевіряють. У даній роботі створено програму з **комплексним індикатором віртуалізації**. Під поняттям комплексності мається на увазі використання технік ухилення різних категорій, а саме: Перевірка фізичних характеристик представлена перевіркою об’єму

оперативної пам’яті (як правило, не менше 8ГБ), Перевірка артефактів – перевіркою дефолтної MAC-адреси (порівняння з адресами, що ВМ використовують за замовчуванням), Анти-аналіз представлений пошуком встановлених VMTools (у ключах реєстру локальної машини), а Перевірка активності користувача – очікуванням натиску лівої кнопки миші протягом 5 секунд. Якщо 3 з 4 ІВ показали позитивний результат, приймається рішення про віртуальну природу цільового середовища (вибір порогу 3 з 4-х є довільним, вибір конкретного значення порогу може бути балансом між точністю визначення віртуалізації та компактністю коду ШПЗ, або кількістю звернень до системних параметрів, пов’язаних з непомітністю ШПЗ, тощо).

Описаний функціонал виконується наступним кодом:

```
import winreg
import win32api
import time
import psutil
from getmac import get_mac_address

check = 0
IV = 0
while True:
    a = win32api.GetKeyState(0x01) #очікування натиску ЛКМ
    if a < 0:
        print("Mouse activity found")
        break
    time.sleep(0.5)
    check = check + 1
    if check > 10:
        IV = IV + 1 #ІВ спрацював, додається +1 аргумент на користь ВМ
        print("No mouse activity detected")
        break

path = winreg.HKEY_LOCAL_MACHINE
while True:
    try: #шукається у реєстрі папка VMTools
        key = winreg.OpenKeyEx(path,
r"SYSTEM\\CurrentControlSet\\Services\\VMTools")
        if key:
            IV = IV + 1
            print("VM Tools detected!")
```

```
        winreg.CloseKey(key)
        break
    except EnvironmentError:
        print("No VMTools found")
        break

RAM_TOTAL = (psutil.virtual_memory().used / psutil.virtual_memory()[2]) * 100
#(кількість зайнятої оперативної пам'яті / відсоток зайнятої)*100% = уся доступна
print('RAM memory volume:', RAM_TOTAL / 1000000000, "GB")
RAM_TOTAL = RAM_TOTAL / 1000000000
RAM_TOTAL = round(RAM_TOTAL, 2) #переводитья у ГБ
if int(RAM_TOTAL) < 8: #чи не менше 8ГБ
    IV = IV + 1
    print("RAW volume is small")

MAC = get_mac_address()
MAC =
MAC[0]+MAC[1]+MAC[2]+MAC[3]+MAC[4]+MAC[5]+MAC[6]+MAC[7]
#вибираються перші 3 біти
if MAC == "00:0c:29" or MAC == "00:50:56" or MAC == "00:1C:14" or MAC
== "00:0C:29" or MAC == "00:05:69":
    print("Default VMWare MAC found!", MAC) #порівнюється з дефолтними
    IV = IV + 1
else:
    print("No default MAC found!")

if IV >=3:
    print("THIS IS VIRTUAL MACHINE!")
else:
    print("THIS IS REAL MACHINE!")
```

Запуск наведеного коду на ВМ, маскування якої не було посилена, призводить до результатів, наведених на Рисунку 3.3:

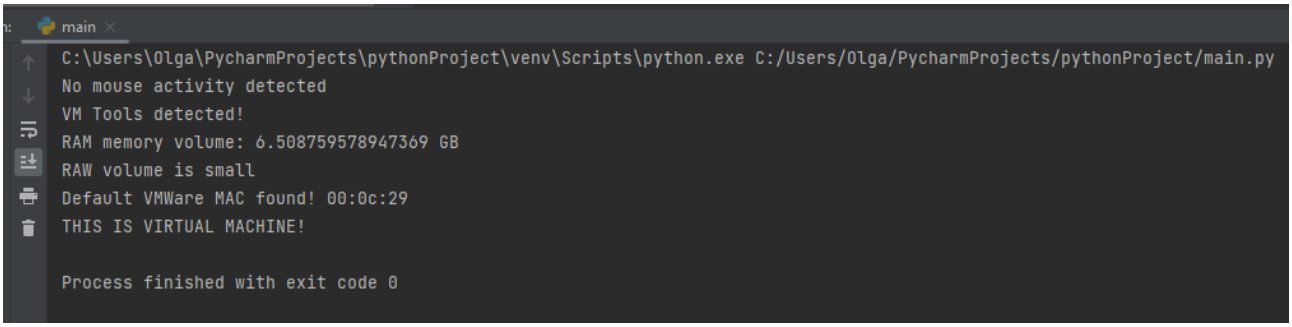

Рисунок 3.3 – Результат виявлення ІВ у ВМ без модифікацій

Як видно, всі чотири перевірки виявили присутність ІВ у цільовій системі, тому програма зробила висновок, що це – ВМ. У випадку ШПЗ, за отримання такого результату, далі б застосовувався прописаний у коді план приховування чи видалення себе з системи.

Застосувавши методику, розроблену у пункті 3.2 можливо замаскувати викриваючи ІВ, у даному випадку за допомогою їх модифікації. Зміни, які за методикою, представленою у пункті 3.2, потрібно внести у ВМ, аби ІВ не були виявлені, включають:

1. Заміну ключа реєстру Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VMTools на …\Services\SomeTools;
2. Збільшення об’єму оперативної пам’яті ВМ до 8ГБ;
3. Проведення натиску лівої кнопки миші під час роботи програми;
4. Заміну значення MAC-адреси за замовчуванням у налаштуваннях ВМ на “3C:C5:04:09:C3:95” – головне замінити 3 перші біти.

Заміна MAC-адреси, як і об’єму оперативної пам’яті, може бути виконана у налаштуваннях ВМ (для VMware Workstation) у місці, зображеному на Рисунку 3.4. Важливо робити це не за допомогою вбудованої функції ВМ «Generate», а з застосуванням стороннього генератора адрес, адже VMware Workstation завжди використовує перші 3 біти за замовчуванням, що призведе до виявлення ІВ.

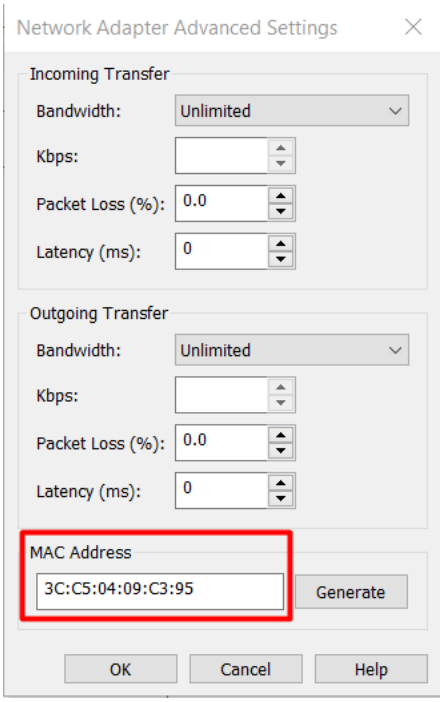

Рисунок 3.4 – Зміна MAC-адреси

Наведені зміни призводять до того, що програма не виявляє жодного індикатора віртуалізації під час роботи та робить висновок про реальність цільового середовища, як видно на Рисунку 3.5:

Рисунок 3.5 – Результат виконання програми після маскування ІВ

Таким чином, продемонстровано роботу з практичного впровадження розробленої методики та її ефективності. Шляхом використання більш складних технік виявлення можливе тестування спроможності ВМ маскувати себе та адаптуватися до перевірок. Створення засобів, що автоматизуватимуть дані дії є дуже перспективним напрямком роботи.

Висновки до розділу 3

В третьому розділі роботи представлені результати статистичного аналізу використання методів виявлення віртуального середовища. Аналіз проводився на базі даних проекту ATT&CK від MITRE. Отримані наступні результати:

- виявлено широке застосування різноманітних технік «анти-віртуалізації» сучасними ШПЗ;
- визначені методи, які застосовуються найчастіше: Спрацювання тригер-умови, Перевірка артефактів та Перевірка фізичних характеристик;
- показана недосконалість класифікації, що застосовується MITRE, запропонована власна класифікація методів виявлення віртуалізації, на основі якій виділені найбільш «небезпечні» індикатори віртуалізації;
- розроблена методика протидії виявленню віртуалізації від найбільш поширених методів виявлення віртуалізації для ВМ на базі десктопної версії Windows, оцінена складність їх реалізації;
- запропоновано створення нового індикатора віртуалізації на основі комплексного застосування найбільш поширених методів виявлення віртуалізації; проведено експериментальні дослідження запропонованої методики та нового комплексного ІВ.

Повертаючись до термінології, запровадженої у Розділі 1, дві з названих найпопулярніших груп категорій ухилення належать до пасивних індикаторів віртуалізації, а саме Перевірка артефактів та Перевірка фізичних характеристик, адже ці техніки експлуатують частини системи (параметри апаратних чи програмних засобів), що є невід'ємною частиною самої системи, але з категорією Спрацювання тригеру-умови все інакше: її техніки перевіряють активні ІВ, адже що логічні, що часові події, ініціюються користувачем. Отже, зважаючи на їх статистичну перевагу над іншими, неможливо виділити і захищатися лише від активних чи лише від пасивних ІВ – потрібен комплексний підхід.

ВИСНОВКИ

В даній роботі представлені результати досліджень актуальної на даний час наукової задачі – виявлення чи відбувається функціонування програми у віртуальному середовищі. Актуальність задачі пояснюється широким використанням технік ухилення, які використовуються ШПЗ при виявленні їх функціонування в віртуальному середовищі. В ході вирішення задачі було проаналізовано існуючі методи виявлення віртуалізації, технології протидії цим методам – маскування віртуалізації, проведений статистичний аналіз використання методів виявлення віртуального середовища. В результаті проведених досліджень з’ясовано:

- не існує такого показника (складової системи), яка не може потенційно викрити природу цільового середовища та стати ІВ;
- повне маскування віртуалізації (повне покриття всіх перевірок, що їх виконує ШПЗ) майже неможливе, не тільки через те, що постійно з’являються нові способи, а й через надзвичайну вартість даних заходів (причому мова йде про новаторські та складні технічні, програмні та людські ресурси) – тому необхідним є визначення балансу між складністю методу маскування віртуалізації та його ефективністю;
- існують певні недоліки класифікації існуючих методів «уникнення від віртуалізації», що призводить до неоднозначності тлумачень термінів та технік методів маскування віртуалізації;
- найбільш часто використовуються методи трьох категорій уникнення від віртуалізації: Спрацювання тригер-умови, Перевірка артефактів та Перевірка фізичних характеристик.

В роботі запропоновано:

- авторську класифікацію методів ухилення від віртуалізації, яка заснована на проведеному статистичному аналізі бази ATT&CK

від MITRE, яка відрізняється від існуючих відсутністю неоднозначності тлумачень методів;

- термін «Індикатор Віртуалізації», причому з поділом на активний та пасивний.

В роботі також:

- виділені найбільш «небезпечні» індикатори віртуалізації;
- розроблена методика протидії виявленню віртуалізації від найбільш поширених методів виявлення віртуалізації для ВМ на базі десктопної версії Windows, оцінена складність їх реалізації;
- запропоновано створення нового індикатора віртуалізації на основі комплексного застосування найбільш поширених методів виявлення віртуалізації;
- проведено експериментальні дослідження запропонованої методики та нового індикатора віртуалізації, показано його ефективність.

Способи протидій різняться від простих модифікацій у системі до написання власних застосунків моніторингу, але після проведення аналізу усіх матеріалів, було виявлено, що найкращим методом уникнути більшості проблем принаймні з наповненням середовища є використання у якості дослідної машини копії системи, яка реально експлуатується, адже це дасть не лише необхідне для уникнення підозр наповнення, але й розуміння недоліків налаштувань, які роблять атаки успішними.

За результатами роботи було виділено також два актуальні супутні напрямки досліджень та розробок, які потенційно можуть мати позитивний вплив і на досліджуване питання, а саме: складність проведення непомітних перехоплень та можливість ін’єкцій у системні процеси, чи процеси легітимного користувача, з подальшим виконанням перевірок від їх імені. Безперечно, розробка рішень, що удосконалюватимуть захист від ін’єкцій у адресний простір процесів або застосунків, що зможуть проводити моніторинг

системних викликів чи інструкцій непомітно, будуть мати високу ефективність і проти технік ухилення.

Потрібно наголосити і на тому, що вивчення технік ухилення допомагає не лише комфортніше та повніше вивчати ШПЗ, а й певніше робити висновок про природу коду, адже навіть без очевидних зловмисних дій використання певної кількості технік має насторожити (хоча не можна забувати, що деякий клас технік може бути використаний і легітимним ПЗ для, наприклад, функціонування ВМ чи процесу налагодження). Також це залежить від того, які середовища компанія використовує для роботи працівників, бо якщо більшість робочих місць існують на віртуальних машинах, то при атаці на дану організацію точно не будуть використовувати техніки виявлення ВМ, бо і так зрозуміло що вони є, але атакувати їх все рівно треба, а от якщо ШПЗ масове, то такі перевірки доречні, проте націлені не так на анти-віртуалізацію, як на анти-аналіз та анти-налагодження. Останнє є напрямком подальших досліджень.

Загалом, розроблена методика та наведена теоретична інформація може ефективно та плідно використовуватись для розроби нових або покращення існуючих ВМ, здатних маскувати себе від ШПЗ, чутливого до середовища виконання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Evasive Malware Now a Commodity [Електронний ресурс] // SecurityWeek – 04.05.2018 – Режим доступу до ресурсу: https://www.securityweek.com/evasive-malware-now-commodity.
2. Григор'єва О.О. Огляд та статистика технологій виявлення віртуалізації шкідливим ПЗ [Текст] / Григор'єва О.О., Стьопочкіна І.В. // XX Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики», 15−16 червня 2022 р., м. Київ. – 2022. – С. 219-222.
3. Hot Knives Through Butter: Evading File-based Sandboxes [Текст] / Singh A., Bu Z. // FireEye. – 2014. – 27 с.
4. Advanced or Not? A Comparative Study of the Use of Anti-debugging and Anti-VM Techniques in Generic and Targeted Malware [Текст] / P. Chen, C. Huygens, L. Desmet, W. Joosen. – Springer International Publishing Switzerland, 2016. – 14 с.
5. Evasions: Registry [Електронний ресурс] // Evasion techniques – Режим доступу до ресурсу: https://evasions.checkpoint.com/techniques/registry.html (дата звернення: 30.05.2022).
6. Evasions: UI artifacts [Електронний ресурс] // Evasion techniques – Режим доступу до ресурсу: https://evasions.checkpoint.com/techniques/ui-artifacts.html#check-number-of-top-level-windows (дата звернення: 30.05.2022).
7. Investigating Anti-Evasion Malware Triggers Using Automated Sandbox Reconfiguration Techniques [Текст] /A. Mills, P. Legg // Journal of Cybersecurity and Privacy. – 2020. – 21 с.
8. Ursnif variant found using mouse movement for decryption and evasion [Електронний ресурс] // Forcepoint – Режим доступу до ресурсу:

https://www.forcepoint.com/fr/blog/x-labs/ursnif-variant-found-using-mouse-movement-decryption-and-evasion (дата звернення: 30.05.2022).

9. Desktop Screen Resolution Stats Worldwide [Електронний ресурс] // GlobalStats – Режим доступу до ресурсу: https://gs.statcounter.com/screen-resolution-stats/desktop/worldwide/#monthly-202103-202203 (дата звернення: 30.05.2022).
10. TrickBot malware now checks screen resolution to evade analysis [Електронний ресурс] // BleepingComputer – Режим доступу до ресурсу: https://www.bleepingcomputer.com/news/security/trickbot-malware-now-checks-screen-resolution-to-evade-analysis/ (дата звернення: 30.05.2022).
11. Malware Evasion Techniques: Same Wolf – Different Clothing [Електронний ресурс] // LastLine – Режим доступу до ресурсу: https://www.lastline.com/labsblog/malware-evasion-techniques/ (дата звернення: 30.05.2022).
12. GravityRAT – The Two-Year Evolution Of An APT Targeting India [Електронний ресурс] // TalosIntelligence – Режим доступу до ресурсу: https://blog.talosintelligence.com/2018/04/gravityrat-two-year-evolution-of-apt.html (дата звернення: 30.05.2022).
13. Environment Awareness [Текст] / A. Torello, F. Guibernau // Defense Evasion Technique Research. – 2020. – 17 с.
14. Malware Evasion Techniques Part 2: Anti-VM Blog [Електронний ресурс] // DeepInstinct – Режим доступу до ресурсу: https://www.deepinstinct.com/blog/malware-evasion-techniques-part-2-anti-vm-blog (дата звернення: 30.05.2022).
15. At Commands, Tor-based Communications: Meet Attor, a Fantasy Creature and Also a Spy Platform [Текст] / Z. Hromcová // ESET Research white papers. – 2019. – 32 с.
16. Virtualization/Sandbox Evasion: Time Based Evasion [Електронний ресурс] // ATT&CK від MITRE – Режим доступу до ресурсу:

https://attack.mitre.org/techniques/T1497/003/ (дата звернення: 30.05.2022).

17. Malware Evasion Techniques Part 3: Anti-Sandboxing [Електронний ресурс] // DeepInstinct – Режим доступу до ресурсу: https://www.deepinstinct.com/blog/malware-evasion-techniques-part-3-anti-sandboxing (дата звернення: 30.05.2022).
18. A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion [Текст] / A. Bulazel, B. Yener // ResearchGate. – 2017. – 22 с.
19. FIN7 Evolution and the Phishing LNK [Електронний ресурс] // Mandiant – Режим доступу до ресурсу: https://www.mandiant.com/resources/fin7-phishing-lnk (дата звернення: 30.05.2022).
20. Defeating Evasive Malware [Текст] // VMRay – 2020. – 7 с.
21. Hiding Debuggers from Malware with Apate [Текст] / H. Shi, J. Mirkovic // USC/Information Sciences Institut – 2017. – 8 с.
22. Resurrecting Anti-virtualization and Anti-debugging: Unhooking your Hooks [Текст] / T. Apostolopoulosa, V. Katosb, K.-K. R. Chooc , C. Patsakis // ResearchGate. – 2021. – 36 с.
23. Process Injection [Електронний ресурс] // ATT&CK від MITRE – Режим доступу до ресурсу: https://attack.mitre.org/techniques/T1055/ (дата звернення: 30.05.2022).
24. Analysis, Anti-Analysis, Anti-Anti-Analysis: An Overview of the Evasive Malware Scenario [Текст] / M. Botacin , V. F. da Rocha , P. L. de Geus , A. Grégio // ResearchGate. – 2017. – 39 с.
25. X64 Function Hooking by Example [Електронний ресурс] – Режим доступу до ресурсу: http://kylehalladay.com/blog/2020/11/13/Hooking-By-Example.html (дата звернення: 30.05.2022).
26. Evasions: Human-like behavior [Електронний ресурс] // Evasion techniques – Режим доступу до ресурсу: https://evasions.checkpoint.com/techniques/human-like-behavior.html#countermeasures (дата звернення: 30.05.2022).

27. Anti-VM and Anti-Sandbox Explained [Електронний ресурс] // Cyberbit – Режим доступу до ресурсу: https://www.cyberbit.com/blog/endpoint-security/anti-vm-and-anti-sandbox-explained/ (дата звернення: 30.05.2022).
28. Defeating malware's Anti-VM techniques (CPUID-Based Instructions) [Електронний ресурс] – Режим доступу до ресурсу: https://rayanfam.com/topics/defeating-malware-anti-vm-techniques-cpuid-based-instructions/ (дата звернення: 30.05.2022).
29. What's the difference between Type 1 vs. Type 2 hypervisor? [Електронний ресурс] // TechTarget – Режим доступу до ресурсу: https://searchservervirtualization.techtarget.com/feature/Whats-the-difference-between-Type-1-and-Type-2-hypervisors (дата звернення: 30.05.2022).
30. Evasions: Timing [Електронний ресурс] // Evasion techniques – Режим доступу до ресурсу: https://evasions.checkpoint.com/techniques/timing.html#difference-vm-hosts (дата звернення: 30.05.2022).
31. Virtualization/Sandbox Evasion [Електронний ресурс] // ATT&CK від MITRE – Режим доступу до ресурсу: https://attack.mitre.org/techniques/T1497/ (дата звернення: 30.05.2022).
32. Longitudinal Study of the Prevalence of Malware Evasive Techniques [Текст] / L. Maffia , D. Nisi, P. Kotzias, G. Lagorio, S. Aonzo, D. Balzarotti // – 2021. – 18 с.
33. A Systematical and Longitudinal Study of Evasive Behaviors in Windows Malware [Текст] / N. Galloro , M. Polino , M. Carminati , A. Continella, S. Zanero // – 2021. – 50 с.

Додаток А Таблиця-статистика технік ухилення

Умовні позначення типів ШПЗ: 1 – крадіжка даних, 2 – бекдор, 3 – троян, 4 – очисник даних, 5 – без файлове шпз, 6 – вимагач, 7 – завантажувач, 8 – інше.

Таблиця А.1 – Техніки ухилення за даними ATT&CK від MITRE

№	Ім'я ШПЗ	Тип ШПЗ	Техніка	Тип	Рік	Що перевіряється
1	Darkhotel	1	SC	Анти-ан	2015	чи є ім'я файла хешем
2	Darkhotel	1	UABC	Актив.	2015	активність положення курсору
3	FIN7	1	UABC	Тригер	2015	подвійне натискання на картинці у документі
4	Okrum	2	SC	Фіз.	2017	кількість внутрішньої пам'яті
5	Okrum	2	TBE	Час вик.	2017	чи прискорений системний час
6	Okrum	2	UABC	Актив.	2017	натиск ЛКМ тричі
7	Spark	2	UABC	Тригер	2017	чи натисне юзер на заставку
8	AppleJeus	7	TBE	Тригер	2018	затримка виконання
9	BADFLICK	2	TBE	Тригер	2018	затримка виконання
10	Bazar	2&7	G	Тригер	2020	велика кількість однакових викликів
11	Bazar	2&7	TBE	Тригер	2020	затримка виконання
12	BendyBear	8	TBE	Тригер	2020	системний час процесора
13	Bisonal	3	G	Артеф.	2018	зчитування значення VMXh
14	Bisonal	3	TBE	Час вик.	2018	чи прискорений системний час
15	Clambling	2	TBE	Тригер	2017	затримка виконання
16	Clop	6	TBE	Тригер	2019	затримка виконання
17	Crimson	3	TBE	Тригер	2016	затримка виконання
18	DRATzarus	3	TBE	Тригер	2020	системний час процесора
19	Egregor	6	TBE	Тригер	2020	затримка виконання
20	EvilBunny	8	SC	Анти-ан	2011	чи є ім'я файла хешем
21	EvilBunny	8	TBE	Тригер	2011	чи прискорений системний час
22	FatDuke	2	TBE	Тригер	2016	рандомізація запуску
23	GoldenSpy	2	TBE	Тригер	2021	затримка виконання

Продовження Таблиці А.1

№	Ім'я ШПЗ	Тип ШПЗ	Технік а	Тип	Рік	Що перевіряється
24	GoldMax	6	SC	Артеф.	2021	mac адреси
25	GoldMax	6	TBE	Тригер	2021	затримка виконання
26	GrimAgent	2	TBE	Тригер	2020	затримка виконання
27	GuLoader	7	SC	Актив.	2019	кількість вікон
28	GuLoader	7	TBE	Час вик.	2019	перевірки часу
29	HermeticWi per	4	TBE	Тригер	2022	затримка виконання
30	LiteDuke	2	TBE	Тригер	2014	затримка виконання
31	Lokibot	1	TBE	Тригер	2015	затримка виконання
32	P8RAT	5	TBE	Тригер	2020	затримка виконання
33	P8RAT	5	SC	Артеф.	2020	процеси ВМ
34	Pony	1	TBE	Тригер	2015	затримка виконання
35	QakBot	3	TBE	Тригер	2007	затримка виконання
36	QakBot	3	SC	Артеф.	2007	виконавчі файли аналізу
37	Raindrop	7	TBE	Тригер	2020	затримка виконання
38	SodaMaster	5	TBE	Тригер	2020	затримка виконання
39	SodaMaster	5	SC	Реєстр	2020	ключ реєстру ВМ
40	SUNBURST	3	TBE	Тригер	2020	затримка виконання
41	SUNBURST	3	SC	Артеф.	2020	ім'я домену
42	ThiefQuest	6	TBE	Тригер	2020	чи прискорений системний час
43	Tomiris	2	TBE	Тригер	2021	затримка виконання
44	TrickBot	3	TBE	Тригер	2016	велика кількість однакових викликів
45	Ursnif	3	TBE	Тригер	2016	затримка виконання
46	WhisperGate	4	TBE	Тригер	2022	затримка виконання
47	WhisperGate	4	G	Анти-ан	2022	програми аналізу
48	XCSSET	2	TBE	Тригер	2020	затримка виконання
49	Astaroth	1	SC	Артеф.	2017	айді Віндовс
50	Astaroth	1	SC	Фіз.	2017	номер диску
51	Astaroth	1	SC	Артеф.	2017	юзернейми
52	Attor	1	SC	Артеф.	2013	зчитування значення VMXh
53	BadPatch	3	SC	Фіз.	2017	ім'я диску, материнскої плати, bios

Продовження Таблиці А.1

№	Ім'я ШПЗ	Тип ШПЗ	Техніка	Тип	Рік	Що перевіряється
54	BLUELIGHT	3	SC	Артеф.	2021	vm tools
55	CSPY Downloader	8	SC	Артеф.	2020	файлові шляхи, структури пам'яті
56	CSPY Downloader	8	SC	Реєстр	2020	значення ключів реєстру
57	Denis	2&3	SC	Реєстр	2017	значення ключів реєстру
58	Denis	2&3	SC	Фіз.	2017	дані процесора
59	Dyre	3	SC	Артеф.	2015	список процесів
60	Dyre	3	SC	Реєстр	2015	значення ключів реєстру
61	Evilnum	6	SC	Артеф.	2018	назви файлів
62	Evilnum	6	SC	Фіз.	2018	апаратна інф.
63	Ferocious	7	SC	Артеф.	2021	версія ОС
64	Ferocious	7	SC	Фіз.	2021	чи можливо програванн звуків
65	Ferocious	7	SC	Актив.	2021	чи активна миша
66	FinFisher	8	SC	Фіз.	2018	айді виробника девайса
67	Frankenstein	8	SC	Артеф.	2019	програми аналізу ВМ
68	Grandoreiro	3	SC	Артеф.	2016	зчитування значення VMXh
69	GravityRAT	3	SC	Фіз.	2016	температура процесора
70	GravityRAT	3	SC	Фіз.	2016	назви ВМ у апаратній інф.
71	InvisiMole	8	SC	Артеф.	2013	артефакти ВМ
72	Lazarus Group	4	SC	Анти-ан	2014	програми аналізу ВМ
73	Lucifer	8	SC	Артеф.	2020	артефакти ВМ
74	MegaCortex	6	SC	Фіз.	2019	кількість ядер процесора
75	NativeZone	7	SC	Артеф.	2021	артефакти ВМ
76	ObliqueRAT	3	SC	Анти-ан	2020	програми аналізу ВМ
77	OilRig	8	SC	Фіз.	2014	наявність миші
78	OopsIE	3	SC	Фіз.	2018	температура процесора
79	OSX_OCEANLOTUS.D	2	SC	Анти-ан	2014	пошук інструкцій
80	PlugX	3	SC	Артеф.	2017	vm tools

Продовження Таблиці А.1

№	Ім'я ШПЗ	Тип ШПЗ	Техніка	Тип	Рік	Що перевіряється
81	PoetRAT	3	SC	Фіз.	2020	об'єм пам'яті диску
82	Remcos	8	SC	Артеф.	2018	пошук ВМ
83	RogueRobin	8	SC	Фіз.	2018	версія bios
84	ROKRAT	8	SC	Артеф.	2016	артефакти ВМ
85	Smoke Loader	8	SC	Артеф.	2011	список процесів
86	SynAck	6	SC	Артеф.	2017	назва папки запуску
87	Trojan.Karagany	3	SC	Фіз.	2010	драйвери
88	Trojan.Karagany	3	SC	Артеф.	2010	файлові шляхи
89	UBoatRAT	8	SC	Артеф.	2017	артефакти ВМ
90	WastedLocker	6	SC	Реєстр	2020	значення ключів реєстру
91	yty	8	SC	Артеф.	2018	артефакти ВМ
92	Carberp	1	G	Анти-ан	2009	видалення перехоплень
93	CHOPSTICK	2	G	Час вик.	2012	час виконання
94	Gelsemium	8	G	Тригер	2014	велика кількість однакових викликів
95	Hancitor	7	G	Тригер	2020	активність у файлі з макросом
96	Metamorfo	3	G	Артеф.	2018	артефакти ВМ
97	Pteranodon	2	G	Анти-ан	2017	заміна розширень файлів шпз
98	RTM	3	G	Артеф.	2019	назви папок
99	StoneDrill	4	G	Анти-ан	2016	виклик системних функцій з помилками