Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Методи виявлення атак шифрувальників на основі аналізу характеристик зашифрованих файлів
(тема)

Виконав: студент 2 курсу, групи СКСм-19-1
Лободенко Г. Я.
(прізвище, ініціали)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

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

Освітня програма Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)

Керівник: ст. викл. Адамов О.С.
(посада, прізвище, ініціали)

Допускається до захисту

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

2020 р.
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп’ютерна інженерія
(код і повна назва)

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

ЗАТВЕРДЖУЮ:
Зав.кафедр
и _____
(підпис)
2
0
“___ ”_____ - __ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Лободенко Георгію Ярославовичу
(прізвищ

1. Тема роботи «Методи виявлення атак шифрувальників на основі аналізу характеристик зашифрованих

затверджена наказом“ 30 ” _____

2. Термін подання 15

3. Вихідні дані до OracleVM VirtualBox
Програми-вимагачі CryptoLockerта DeriaLocker
Методи детектування зашифрованих файлів

4. Перелік питань, що
Аналіз предметної області
Підготовка тестових даних
Методи виявлення активності шифрувальника
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 15 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Видача теми роботи, її узгодження та затвердження.	01.09.2020 – 07.09.2020	
2.	Аналіз проблемної області, постановка задачі, вибір засобів реалізації.	07.09.2020 – 21.09.2020	
3.	Придбання необхідних компонентів.	21.09.2020 – 05.10.2020	
4.	Збірка пристрою.	05.10.2020 – 26.10.2020	
5.	Створення додатка на смартфон.	26.10.2020 – 02.11.2020	
6.	Написання коду для мікроконтролера.	02.11.2020 – 09.11.2020	
7.	Оформлення пояснювальної записки.	09.11.2020 – 14.12.2020	
8.	Перевірка виконаного проекту, допуск до захисту.	14.12.2020 – 21.12.2020	
9.	Захист проекту.	23.12.2020	

Дата видачі завдання 01 вересня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Адамов О. С.

РЕФЕРАТ

Пояснювальна записка містить 75 сторінок, 30 рисунків, 19 таблиць, 7 лістингів, 32 джерел посилання.

ШИФРУВАННЯ, ПРОГРАМИ-ВИМАГАЧІ, ЕНТРОПІЯ, .MAGIC, СИГНАТУРА, ВІРТУАЛЬНА МАШИНА

Метою атестаційної роботи є аналіз та дослідження характеристик зашифрованих файлів для виявлення атак шифрувальників, збір даних та формування з них тестових наборів для подальшого використання в засобах боротьби з програмами-вимагачами.

Аналіз зашифрованих даних та їх характеристик є відправною точкою для формування подальших тестових наборів, які будуть використовуватись для емуляції програм-вимагачів, що, в свою чергу, демонструють інструменти, способи та методи, котрими користуються розробники шкідливих програм.

ABSTRACT

The explanatory note contains 75 pages, 30 figures, 19 tables, 7 listings, 32 sources of references.

ENCRYPTION, DEMANDING PROGRAMS, ENTROPY, .MAGIC, SIGNATURE, VIPTUAL MACHINE

The purpose of the certification work is to analyze and study the characteristics of encrypted files to detect encryption attacks, data collection and formation of test sets from them for further use in the fight against extortionist programs.

Analysis of encrypted data and their characteristics is the starting point for the formation of further test sets that will be used to emulate extortionist programs, which, in turn, demonstrate the tools, methods and techniques used by malware developers.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПК – Персональный комп’ютер;

ПЗ – Програмне забезпечення;

ОС – Операційна система

API – Application programming interface (інтерфейс програмування);

CPU – Central processing unit (центральний процесор);

GPU – Graphics processing unit (відеокарта);

ASCII – American standard code for information interchange (Американський стандартний код для обміну інформацією);

VM – virtual machine (віртуальна машина);

URL – Uniform Resource Locator (уніфікований показник ресурса);

FTP – File Transfer Protocol (протокол передачі файлів по мережі);

TBE – The Behavior Educations (модуль поведінкового аналізу)

HIPS – Host-based intrusion prevention system (система запобігання вторгнень)

WMI – Windows Management Instrumentation (інструментарій управління Windows)

RAT – Інструмент віддаленого пошуку

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

VPN – Virtual Private Network (віртуальна приватна мережа)

VME – Virtual Machine Environment (середа віртуальних машин/пісочниця)

BIOS – Basic Input/Output System (базова система вводу/виводу)

UEFI – Unified Extensible Firmware Interface (Уніфікований розширюваний інтерфейс мікропрограми)

ВСТУП

На сьогоднішній день питання захисту інформації є одним із найважливіших, так як зі збільшенням потоку даних у всьому світі, росте попит на незаконне оволодіння цінною інформацією заради фінансової вигоди, шантажу, тощо. Навіть компанії гіганти страждають від наслідків проникнення шкідливого програмного забезпечення до їх систем.

Найчастіше розробників шкідливого програмного забезпечення цікавить фінансова складова. Таким чином можна відзначити такий тип незаконного ПЗ, як програми-вимагачі.

Програми-вимагачі представляють проблему для підприємств, освітніх установ і систем охорони здоров'я. Дослідження в області кібербезпеки продемонстрували, що це сімейство шкідливого ПЗ здатне без труднощів вивести з ладу базову інфраструктуру, необхідну для функціонування міст, регіонів або, навіть всієї країни.

Згідно з даними звіту американського постачальника кібербезпеки FireEye, який привертає увагу експертів з кібербезпеки до цієї проблеми, 68% атак залишаються непоміченими [1].

Статичний аналіз файлів на основі їх характеристик являється першим ступенем детектування будь-якого типу зашифрованих файлів та запобігає подальшому їх запуску, який може привести до захвату системи стороннім шкідливим програмним забезпеченням. Навіть не запускаючи файли, можна зробити висновки щодо, з першого погляду, безпечних файлів в системі, та виділити декілька характеристик, на які варто звернути увагу.

Дослідженням методів та технік якими користуються програми-здирники та аналізу їх самих дасть представлення про методи, котрими можна протидіяти хакерським нападам такого типу.

Знову ж таки актуальність даної теми обумовлена кількістю випадків захвату систем шкідливим програмним забезпеченням, особливо програмами-здирниками. Дані програми несуть загрозу для всіх типів користувачів: від цілих систем корпорацій, які містять цінні файли, але знаходяться під відмінним захистом до так званих легких цілей – звичайних наївних користувачів. Детектування подібних файлів є основою для запобігання подібним вторгненням.

Статичний аналіз з подібними методами застосовується в більшості антивірусних системах, але в зв’язку з прогресом технологій захисту інформації, дані методи покриваються машинним навчанням.

Цілі та завдання дослідження:

1) розібрати поняття шифрування;

2) описати методи, якими користуються розробники шкідливих програм-шифрувальників;

3) розглянути техніки та методи, котрими користуються програми-вимагачі;

4) привести приклади програм-вимагачів;

5) виділити методи для детектування зашифрованих файлів;

6) програмно реалізувати кожний метод окремо один від одного;

7) проаналізувати кожний метод і на основі цих даних об’єднати таким чином щоб досягнути найліпшого результату.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Шифрування. Визначення, найбільш відомі методи.

Шифрування. Метод, що підвищує безпеку файлу або повідомлення, через кодування вмісту, щоб його могла прочитати тільки людина, у якого є відповідний ключ шифрування для його декодування. Наприклад, якщо робити покупку через Інтернет, інформація про транзакції (така як номер телефону, e-mail і кредитна карта) зазвичай зашифрована для забезпечення безпеки.

Зазвичай метою шифрування є збереження конфіденційності як при передачі інформації, так і під час її зберігання. Крім того, цілісність інформації та її незмінність, що включає підтвердження того, що ніхто не міняв підписаний документ і підписав є тим, ким він себе називає.

Можна виділити декілька основних методів шифрування, кожен з яких є по своєму індивідуальним, та за котрими стоїть клопітка робота.

Симетричне шифрування. Також відоме як шифрування з секретним ключем або приватним ключем, воно складається з використання одного і того ж ключа для шифрування і дешифрування. Це шифрування складається з застосування алгоритму по відношенню даних, які треба зашифрувати, використовуючи закритий ключ, щоб зробити їх нерозбірливими. За допомогою даного алгоритму шифрування можна захистити найпростіші системи, повідомлення та файли. Для більш серйозних систем не рекомендується для використання.

Асиметричне шифрування. Відомий як метод за допомогою відкритого ключа). В асиметричному методі шифрування ключі вказуються парами:

– відкритий ключ для шифрування;

– секретний ключ для розшифрування.

В системі шифрування з відкритим ключем користувачі обирають випадковий ключ, який знають тільки вони, котрий буде виконувати роль закритого ключа. На основі цього ключа автоматично створюється алгоритм (відкритий ключ). Користувачі обмінюються цим відкритим ключем по незахищеному каналу, але розшифрування буде відбуватися виключно по заздалегідь вибраному закритому ключу.

– шифрування транспозиції. Метод шифрування шляхом транспонування полягає в тому, що змінюється порядок даних, щоб зашифрувати їх і зробити незрозумілими. Це може означати, наприклад, зміна геометрії даних, щоб зробити їх візуально непридатними для використання.

– шифрування за допомогою підстановки. Підстановка – це заміна одного або декількох об'єктів (зазвичай букв) одним або декількома різними об'єктами. Існує кілька видів заміни:

– моно-алфавітна підстановка полягає в заміні кожної букви повідомлення іншою буквою алфавіту;

– полі-алфавітна заміна – це періодичне використання серії моно-алфавітних шифрів;

– гомофонічна заміна дозволяє кожній букві простого текстового повідомлення відповідати потенційній групі різних символів.

1.2 Шкідливе програмне забезпечення. Програми-вимагачі.

1.2.2.1 Основі відомості.

Щодня, майже кожна система, незважаючи на її захищеність, може потрапити в пастку розробників шкідливого програмного забезпечення і стати їх заручником. Програми-вимагачі однин із видів шкідливого програмного забезпечення, який вважається одним із самих настирних, так як блокує критичні ресурси шляхом шифрування даних. Щоб проникнути до системи, данні програми використовують стандартні сценарії атаки,

наприклад набори даних в програмах або фішингові атаки на e-mail. Потрапляючи до системи, шкідливі файли закріплюються в обчислювальній машині та встановлюють контроль над системою. Програми-вимагачі змушують жертву заплатити викуп, так як блокують доступ до більшості функцій та утримують ресурси під невідомим для звичайного користувача методом шифрування. Подібні програми віддають перевагу загальноприйнятим способам шифрування на основі відкритого та закритого ключа, так як для розшифрування даних, після подібного нападу, існує два шляхи – заплатити викуп або спробувати повернути файли за допомогою резервних копій. Після оплати, розробник шкідливого ПЗ видає ключ (або ключі) для розшифрування, який (які), в свою чергу, дають можливість відновити файли, але також бувають випадки, коли зловмисник зникає без надання способів для повернення файлів.

Цілями програм вимагачів э не тільки звичайні користувачі. При відмові до важливих ресурсів страждають різні галузі, наслідками вторгнення до котрих є катастрофічними, наприклад:

– медична галузь: повний локдаун медичної установи, неможливість проведення будь-якого типу хірургічних утручань, прийому пацієнтів або взаємодії зі швидкими;

– фінансова галузь: фінансові установи не зможуть виконати будь-які операції без підключення до інтернету;

– торгівельна галузь: взаємопов’язана з фінансовою галуззю, так як платежі не будуть оброблятись.

1.2.2.2 Огляд технік та методів які використовують розробники шкідливого програмного забезпечення

Мистецтво злому з’явилося разом з інтернетом та розвивалось паралельно з ним, так як це невід’ємна його частина. З роками технології змінювались, з’являлись нові методи та техніки. З підвищенням рівню та попиту на захист від шкідливого програмного забезпечення та хакерів, інші в

свою чергу мали можливість для винаходження нових способів для обходу таких систем. Даний симбіоз розвивається в обидві сторони чим штовхає прогрес в обидві сторони. Цей процес породив низку технік та методів, якими хакери користуються і по цей день. Ось декілька із них:

1.2.2.1 Послаблене шифрування. Зловмисники можуть поставити під загрозу можливості шифрування мережевого пристрою, щоб обійти його, яке в іншому випадку могло б захистити передачу даних [3].

Шифрування може використовуватися для захисту переданого мережевого трафіку з метою збереження його конфіденційності (захист від несанкціонованого розкриття) і цілісності (захист від несанкціонованих змін). Шифри шифрування використовуються для перетворення відкритого тексту повідомлення в зашифрований текст і можуть зажадати великих обчислювальних ресурсів для дешифрування без пов'язаного ключа дешифрування. Зазвичай довгі ключі збільшують вартість криптоаналізу або дешифрування без ключа.

Хакери взламують пристрої, шифрують мережевий трафік, і маніпулювати ними. Наприклад, за допомогою таких дій, як зміна способу системи, зменшення простору ключів і відключення криптографічного обладнання, розробник може негативно вплинути на пристрій і/або зробити неможливим безпечне шифрування мережевого трафіку. Це створює більший ризик несанкціонованого розкриття і може полегшити маніпулювання даними, вхід до облікового запису або збір даних.

1.2.2.2 Зашифрований канал. Розробники шкідливого ПЗ використовують відомий алгоритм шифрування, щоб приховувати командування та управління трафіком, а не покладатися на будь-який властивий захист, що забезпечується протоколом зв'язку. Незважаючи на використання захищеного алгоритму, ці реалізації можуть бути вразливими до зворотного проектування, якщо секретні ключі кодуються та/або генеруються у зразках шкідливих програм/файлах конфігурації.

1.2.2.3 Шифрування даних для впливу. Хакери зашифровують дані

в цільових системах або у великій кількості систем в мережі, щоб порушити доступність системних і мережевих ресурсів. Вони можуть спробувати зробити збережені дані недоступними, зашифрувавши файли або дані на локальних і віддалених дисках і заборонивши доступ до ключа дешифрування. Це може бути зроблено для того, щоб отримати грошову компенсацію від жертви в обмін на дешифрування або ключ дешифрування (програма-вимагач), або щоб зробити дані назавжди недоступними в випадках, коли ключ не зберігається або не передається [4]. У разі програм-вимагачів звичайні призначені для користувача файли, такі як документи Office, PDF-файли, зображення, відео, аудіо, текст і вихідні файли, зазвичай шифруються. У деяких випадках зловмисники можуть зашифрувати важливі системні файли, розділи диска і MBR.

Щоб максимально вплинути на цільову організацію, шкідливе ПЗ, призначене для шифрування даних, може мати черв'ячні функції для поширення по мережі за допомогою інших методів атаки, таких як справжні облікові записи, скидання облікових даних ОС і загальні ресурси адміністратора SMB / Windows.

1.2.2.4 Зовнішні сервіси. Використання зовнішніх віддалених служб для початкового доступу і збереження в мережі є частим шляхом для взлому. Дистанційні служби, такі як VPN, Citrix і інші механізми доступу, дозволяють користувачам підключатися до внутрішніх мережевих ресурсів підприємства із зовнішніх джерел. Часто існують шлюзи віддалених служб, керуючих підключеннями і аутентифікації облікових даних для цих служб. Такі, як Windows Remote Control, можна використовувати ззовні.

Доступ до діючих облікових даних для використання служби часто є вимогою, яке може бути отримано шляхом збору облікових даних шляхом отримання облікових даних користувачів після злому корпоративної мережі [5]. Віддалений доступ до сервісів місцева якості резервного або постійний механізм доступу під час операції.

Зловмисники можуть впровадити образи хмарних контейнерів з

шкідливим кодом для забезпечення стійкості. Amazon Web Service (AWS) Образи машин Amazon (AMI), образи хмарної платформи Google (GCP) і образи Azure, а також популярні середовища виконання контейнерів, такі як Docker, можуть бути імплантовані або створені для резервного копіювання. Залежно від того, як надається інфраструктура, це може забезпечити постійний доступ, якщо інструмент підготовки інфраструктури дає вказівку завжди використовувати останній образ [6].

Був розроблений інструмент, який спрощує установку бекдор в образи хмарних контейнерів [6]. Якщо у зловмисника є доступ до скомпрометованому екземплярі AWS і дозволу на перерахування доступних образів контейнерів, він може розгорнути бекдор, наприклад Web Shell [7]. Зловмисники також можуть впроваджувати образи Docker, які можуть бути ненавмисно використані в хмарних розгортання, як повідомлялося в деяких випадках ботнетів для криптомайнінг.

1.2.2.5 Помилковий контролер домену. Розробники шуідливого ПЗ можуть зареєструвати неправдивий контролер домену, щоб дозволити маніпулювання даними Active Directory. DCShadow може бути використаний для створення неправдивого контролера домену (DC). DCShadow – це метод маніпулювання даними Active Directory (AD), включаючи об’єкти та схеми, шляхом реєстрації (або повторного використання неактивної реєстрації) та моделювання поведінки DC [7]. Після реєстрації зловмисний DC може мати змогу вводити та тиражувати зміни в інфраструктуру AD для будь-якого об’єкта домену, включаючи облікові дані та ключі.

Реєстрація неправдивого DC включає створення нового сервера та об'єктів nTDSDSA у розділі конфігурації схеми AD, що вимагає привілеїв адміністратора (доменних чи локальних для DC) або хешу KRBTGT [7].

Ця техніка може обійти системні журнали та монітори безпеки, такі як продукти інформації про безпеку та управління подіями (SIEM) (оскільки про дії, здійснені на неправдивому постійному струмі, ці датчики можуть не повідомляти)[7]. Метод може також використовуватися для зміни та

видалення реплікації та інших пов'язаних метаданих для перешкоджання судово-медичному аналізу. Супротивники можуть також використовувати цю техніку для здійснення ін'єкції історії SID та / або маніпулювання об'єктами AD (такими як облікові записи, списки контролю доступу, схеми) для встановлення бекдорів для стійкості [7].

1.2.2.6 Заважати змісту загального характеру. Можна доставляти корисні дані у віддалені системи, додаючи контент в загальні репозиторії, такі як мережеві диски або внутрішні репозиторії коду. Зміст, що зберігається на мережевих дисках або інших загальних папках, можуть бути пошкоджені додаванням шкідливих програм, сценаріїв або коду експлойту до допустимих файлів. Як тільки користувач відкриє спільно використовуваний заражений контент, шкідлива програма може бути запущена для запуску коду зловмисника в віддаленій системі. Зловмисники можуть використовувати пошкоджений загальний контент для бокового переміщення.

Зведення загального доступу до каталогу - це різновид цього методу, в якому використовуються кілька інших методів для поширення шкідливих програм при доступі користувачів до спільного мережного ресурсу. Він використовує зміна ярликів для файлів каталогів .LNK, які використовують маскування, щоб вони виглядали як справжні каталоги, приховані за прихованими файлами і каталогами. Шкідливі каталоги на основі LNK мають вбудовану команду, яка запускає прихований шкідливий файл в каталозі, а потім відкриває реальний передбачуваний каталог, щоб очікуване дію користувача і раніше виконувалося. При використанні з часто використовуваними мережними каталогами цей метод може призвести до частих повторних заражень і широкому доступу до систем і, можливо, новим облікових записів з більш високими привілеями [8].

Зловмисники також можуть поставити під загрозу загальні мережеві ресурси через двійкове зараження, додавши свій код в справний двійковий файл в загальній мережевий папці. Шкідлива програма може змінити вихідну

точку входу (OEP) справного виконуваного файлу, щоб гарантувати виконання легітимного коду. Зараження може продовжити поширення через новий заражений файл, коли він запущений віддаленої системою. Ці зараження можуть бути націлені як на виконавчі, так і на небінарні формати, що закінчуються розширеннями, включаючи, крім іншого, .EXE, .DLL, .SCR, .BAT і / або .VBS.

1.2.2.7 Проксі в браузері. Зловмисники можуть використовувати уразливості системи безпеки і вбудовані функції в програмному забезпеченні браузера для зміни змісту, зміни поведінки і перехоплення інформації за допомогою різних методів використання браузера [9].

Конкретний приклад – це коли зловмисник впроваджує в браузер програмне забезпечення, яке дозволяє йому наслідувати файли cookie користувача, сеанси HTTP і сертифікати клієнтів SSL і використовувати браузер як спосіб підключення до інтрамережі [9].

Ротація браузера вимагає SeDebugPrivilege і процесу високої цілісності. Трафік браузера перенаправляється з браузера зловмисника через браузер користувача шляхом настройки проксі-сервера HTTP, який буде перенаправляти будь-який трафік HTTP і HTTPS. Це ніяк не впливає на призначений для користувача трафік. Проксі-з'єднання розривається, як тільки браузер закривається. У якій би процес браузера не був впроваджений проксі, противник бере на себе контекст безпеки цього процесу. Браузери зазвичай створюють новий процес для кожної відкритої вкладки, і дозволи і сертифікати відповідно діляться. З цими дозволами зловмисник може переглядати будь-який ресурс в інтрамережі, доступний через браузер і має достатні дозволу, наприклад Sharepoint або веб-пошту.

1.2.2.8 Виявлення системних мережевих підключень. Є можливість спробувати отримати список мережевих підключень до або від скомпрометованої системи, до якої вони в даний час отримують доступ, або від віддалених систем, запитуючи інформацію по мережі.

Хакер, який отримав доступ до системи, яка є частиною хмарної

середовища, може зіставити віртуальні приватні хмари або віртуальні мережі, щоб визначити, які системи і служби підключені. Прийняті дії, ймовірно, будуть одного і того ж типу виявлення в залежності від операційної системи, але отримана інформація може включати інформацію про мережевий хмарної середовищі, відноситься до цілей противника. Постачальники хмарних послуг можуть використовувати свої віртуальні мережі по-різному, але вразливі майже однаково [10].

Утиліти і команди, які отримують цю інформацію, включають netstat, "net use" і "net session" з мережі. У Mac і Linux для виведення списку поточних підключень можна використовувати netstat і lsof. who -a і w можна використовувати, щоб показати, які користувачі в даний час увійшли в систему, аналогічно "net session".

1.2.2.9 Багатоступінчасті канали. Існує метод створення кількох ступенів для управління і контролю, які використовуються в різних умовах або для певних функцій. Використання декількох кроків може ускладнити пошук каналу управління і контролю.

Засоби віддаленого доступу зв'яжуться з командним сервером першого рівня для отримання інструкцій. На першому етапі можуть бути автоматизовані можливості для збору базової інформації про вузол, інструментів оновлення і завантаження додаткових файлів. На цьому етапі можна завантажити інший інструмент віддаленого доступу (RAT) для перенаправлення хосту на сервер управління і контролю другого рівня. Друга фаза, ймовірно, буде більш функціональною і дозволить противнику взаємодіяти з системою через зворотний снаряд і додаткові функції RAT.

Різні етапи, швидше за все, будуть розміщені окремо, без дублювання інфраструктури. Завантажувач може також мати зворотні виклики резервного копіювання першого етапу або резервні канали в разі, якщо вихідний канал зв'язку першого етапу виявлений і заблокований.

1.2.2.10 Діючі акаунти. У хакерів є можливість отримати і використовувати облікові дані існуючих облікових записів як засіб

отримання початкового доступу, зберігання, підвищення привілеїв чи обходу безпеки. Скомпрометовані облікові дані можуть використовуватися для обходу засобів управління доступом, розташованих на різних ресурсах в системах в мережі, а також можуть використовуватися для постійного доступу до віддалених систем і доступним ззовні службам, таким як VPN, Outlook Web Access і віддалений робочий стіл. Зламані облікові дані також можуть надати зловмисникові підвищені привілеї в певних системах або доступ до обмежених областей мережі. Розробники шкідливого програмного забезпечення відмовляються від використання шкідливих програм або інструментів в поєднанні з законним доступом, що надаються цими обліковими даними, щоб утруднити виявлення їх присутності.

Перекриваються дозволу для локальних, доменних і хмарних облікових записів в мережевих системах викликають занепокоєння, оскільки зловмисник може перемикатися між обліковими записами і системами для досягнення високих рівнів доступу (наприклад, адміністратора домену або підприємства), щоб обійти засоби контролю доступу в масштабі підприємства [11].

1.2.2.11 Фішинг для отримання інформації. Перш ніж скомпрометувати жертву, хакери можуть відправити фішингові повідомлення для отримання конфіденційної інформації, яка може бути використана при орієнтування. Фішинг з метою отримання інформації – це спроба обманом змусити мети розкрити інформацію, часто облікові дані або іншу корисну інформацію. Фішинг для отримання інформації відрізняється від фішингу тим, що його метою є збір даних від жертви, а не виконання шкідливого коду.

Всі форми фішингу розробляються за допомогою електроніки. Фішинг може бути цільовим, так званим цільовим фішингом. При цільовому фішинг зловмисник буде націлений на конкретну людину, компанії або галузі. У більш загальному плані зловмисники можуть проводити нецільової фішинг, наприклад, в кампаніях масового збору облікових даних.

Зловмисники отримують інформацію безпосередньо, обмінюючись електронними листами, миттєвими повідомленнями або іншими засобами електронного зв'язку [12]. Фішинг для отримання інформації часто включає в себе методи соціальної інженерії, такі як видача себе за джерело із зазначенням причини для збору інформації (наприклад, створення облікових записів або злом облікових записів) і / або відправка кількох, на перший погляд термінові повідомлення.

1.2.2.12 Незаконне отримання можливостей. Перш ніж скомпрометувати жертву, зловмисники можуть купити і / або вкрасти функції, які можуть бути використані при орієнтування. Замість того, щоб розвивати свої власні можливості всередині компанії, зловмисники можуть купувати, завантажувати або вкрасти їх. Дії можуть включати в себе покупку шкідливих програм, програмного забезпечення (включаючи ліцензії), експлойтів, сертифікатів та інформації, пов'язаної з уразливими. Зловмисники можуть отримати можливості для підтримки своїх операцій на багатьох етапах життєвого циклу зловмисника.

Крім завантаження безкоштовних шкідливих програм, програмного забезпечення і експлойтів з Інтернету, також отримати ці можливості від третіх осіб. До стороннім організаціям можуть ставитися технологічні компанії, що спеціалізуються на шкідливі програми і експлойтів, на кримінальних ринках або від приватних осіб [13].

Крім отримання можливостей, вони вкрасти можливості третіх осіб (включаючи інших зловмисників). Це може включати крадіжку ліцензій на програмне забезпечення, шкідливе ПЗ, сертифікати SSL / TLS і підписи коду, а також крадіжку закритих баз даних з уразливими або експлойта.

1.2.2.13 Drive-by компроміс. Неправомірний доступ до системи може бути реалізований в тому випадку, якщо користувач відвідує веб-сайт під час звичайного перегляду. За допомогою цього методу веб-браузер користувача зазвичай є метою для експлуатації, але зловмисники також

можуть використовувати зламані сайти для невикористання, наприклад для отримання токена доступу до додатка.

Є кілька способів доставити код експлойта в браузер, в тому числі:

– законний сайт буде скомпрометований, якщо зловмисники впровадили будь шкідливий код, такий як JavaScript, iFrames і міжсайтовий сценарії;

– шкідлива реклама оплачується і обслуговується законними постачальниками реклами;

– вбудовані інтерфейси веб-додатків використовуються для вставки будь-якого іншого типу об'єкта, який може використовуватися для відображення веб-контенту або містити сценарій, який запускається на клієнтів, які відвідують його (наприклад, повідомлення на форумі, коментарі та інший веб-контент, керований користувачем).

Часто веб-сайт, який використовується зловмисником, відвідує конкретне співтовариство, таке як уряд, конкретна галузь або регіон, де мета полягає в тому, щоб скомпрометувати конкретного користувача або групу користувачів на основі спільних інтересів. Ця цільова атака називається стратегічним зломом мережі або атакою водопою. Є кілька відомих прикладів [11].

Типовий процес компромісу:

1) користувач відвідує веб-сайт, який використовується для розміщення контенту, контрольованого зловмисником;

2) скрипти виконуються автоматично, зазвичай шукаючи потенційно вразливу версію в версіях браузера і плагіну. Від користувача може знадобитися допомога в цьому процесі, включивши сценарії або активні компоненти сайту і ігноруючи діалогові вікна з попередженнями;

3) коли виявляється вразлива версія, в браузер доставляється код експлойту;

4) якщо експлойт буде успішним, він призведе до виконання коду зловмисника в системі користувача, якщо не встановлені інші засоби захисту. У деяких випадках потрібно другі відвідини сайту після початкового сканування перед доставкою коду експлойту.

Можна використати скомпрометовані веб-сайти для доставки користувача до шкідливого додатком, призначеному для крадіжки токенів доступу до додатків, як токени OAuth для доступу до захищених додатків і інформації. Ці шкідливі програми були доставлені через спливаючі вікна на законних сайтах [12].

1.2.2.14 Компроміс ланцюга поставок. У хакерів є можливість маніпулювати продуктами або механізмами доставки продуктів до того, як вони будуть отримані кінцевим споживачем, щоб поставити під загрозу дані або систему.

Компрометація ланцюжка поставок може статися на будь-якому етапі ланцюжка поставок, включаючи:

– маніпуляції з інструментами розробки;

– управління середовищем розробки;

– управління репозиторіями вихідного коду (загальнодоступними або приватними);

– управління вихідним кодом в залежностях з відкритим вихідним кодом;

– управління механізмами поновлення / поширення програмного забезпечення;

– скомпрометовані / заражені образи системи (відомі випадки зараження знімних носіїв на заводі) [8];

– заміна легального ПЗ модифікованими версіями;

– продаж модифікованих / підроблених продуктів законним дистриб'юторам.

Хоча компрометація ланцюжка поставок може вплинути на будь-яку частину обладнання або програмного забезпечення, зловмисники часто

зосереджуються на шкідливих надбудови до законного програмного забезпечення в каналах поширення або оновлення програмного забезпечення [3]. Орієнтування може бути визначений для бажаного набору жертв [8], або шкідливі програми може бути поширене серед широкого кола споживачів, але перехід до додаткової тактиці може бути тільки для конкретних жертв. Популярні проекти з відкритим вихідним кодом, які використовуються в якості залежностей в багатьох додатках, також можуть бути використані як засіб додавання шкідливого коду для залежних користувачів [6].

1.2.2.15 Альтернативні матеріали. Зловмисники можуть використовувати альтернативні матеріали для аутентифікації, такі як хеші паролів, квитки Kerberos і маркери доступу до додатків, для навігації по середовищі і обходу звичайних засобів контролю доступу до системи.

Процеси аутентифікації зазвичай вимагають дійсної ідентифікації (наприклад, імені користувача) разом з одним або декількома факторами аутентифікації (такими як пароль, PIN-код, фізична смарт-карта, генератор токенов і т. д.) [13]. Альтернативний аутентифікаційний матеріал законно генерується системами після того, як користувач або додаток успішно автентичності шляхом надання дійсної ідентичності і необхідних чинників аутентифікації. Альтернативний аутентифікаційний матеріал також може бути створений в процесі створення ідентичності [12].

Кешування альтернативного матеріалу для аутентиікації дозволяє системі підтвердити, що особистість успішно увійшла, не пропонуючи користувачеві повторно ввести фактор аутентифікації. Оскільки альтернативна аутентифікація повинна підтримуватися системою – або в пам'яті, або на диску – її можна вкрасти за допомогою методів входу в аккаунт. Крадіжкою альтернативних аутентифікаційних матеріалів, зловмисники можуть обійти контроль доступу до системи і увійти до системи, не знаючи пароля у відкритому вигляді або будь-яких додаткових чинників аутентифікації.

1.2.2.16 Віртуалізація пісочниці / ухилення. Використання різні інструменти для виявлення і обходу середовищ віртуалізації і аналізу є нерідким випадком взлому. Це може включати зміну поведінки на основі результатів перевірки артефактів, що вказують на середу віртуальних машин (VME) або пісочницю. Якщо зловмисник виявляє VME, він може змінити своє шкідливе ПЗ, щоб відокремитися від жертви або приховати основні функції імплантату. Вони також можуть шукати артефакти VME, перш ніж відкидати вторинні або додаткові корисні дані. Зловмисники можуть використовувати інформацію, отриману в результаті обходу віртуалізації / пісочниці під час автоматичного виявлення, для формування подальшої поведінки.

Вони використовувати різні способи для виконання віртуалізації / обходу в пісочниці, наприклад, перевірка інструментів моніторингу безпеки (таких як Sysinternals, Wireshark і т. д.) Або інших системних артефактів, пов'язаних з аналізом або віртуалізацією. Зловмисники також можуть перевірити законну активність користувачів, щоб визначити, чи знаходяться вони в середовищі аналізу. Додаткові методи включають використання таймерів сну або циклу під шкідливий код, щоб уникнути роботи в тимчасовій пісочниці [14].

1.2.2.17 Сигналізація трафіку. При бажанні можна використовувати сигналізацію трафіку, щоб приховати відкриті порти або інші шкідливі функції, які використовуються для наполегливості або управління і контролю. Сигналізація трафіку включає використання магічного значення або послідовності, які повинні бути відправлені в систему для запуску спеціального відповіді, такого як відкриття закритого порту або виконання шкідливої завдання. Це може приймати форму відправки серії пакетів з певними характеристиками до відкриття порту, зловмисник може використовувати для управління і контролю. Як правило, ця серія пакетів складається з спроб підключення до заздалегідь визначеної послідовності закритих портів (наприклад, блокуючим портам), але може включати

незвичайні прапори, певні рядки або інші унікальні характеристики. Після завершення послідовності відкриття порту може бути виконано брандмауером на основі хоста, але також може бути реалізовано за допомогою спеціального програмного забезпечення.

Зловмисники також можуть підключитися до вже відкритого порту, але служба, яка прослуховує цей порт, буде реагувати на команди або запускати інші шкідливі функції тільки в тому випадку, якщо передані відповідні магічні значення.

Пакети сигналів можна відстежувати для ініціювання зв'язку різними способами. Один із способів, спочатку реалізований Cd00r [15], – використовувати бібліотеки libpcap для прослуховування розглянутих пакетів. Інший метод використовує сирі сокети, що дозволяє шкідливому ПО використовувати порти, які вже відкриті для використання іншими програмами.

На мережевому диску зловмисники можуть використовувати створені пакети для включення аутентифікації мережевого пристрою для стандартних послуг, пропонованих пристроєм, таких як telnet. Ця сигналізація також може використовуватися для відкриття закритого порту служби, такий як telnet, або для запуску модифікації шкідливого модуля імплантату на пристрої, додавання, видалення або зміни шкідливих можливостей [16]. Щоб включити цю сигналізацію трафіку на вбудованих пристроях, зловмисники повинні спочатку отримати і використовувати образ системи виправлень через монолітної природи архітектури.

1.2.2.18 Підрив довірчого управління. Зловмисники можуть скомпрометувати заходи безпеки, які попереджають користувачів про ненадійною активності або запобігають запуск ненадійних програм. Операційні системи і продукти безпеки можуть містити механізми для ідентифікації програм або сайтів як мають певний рівень довіри. Приклади цих функцій можуть включати програму, запуск якої вирішено, оскільки воно підписано чинним сертифікатом підпису коду, програма, яка запитує

користувача, так як у неї є атрибут, встановлений для завантаження з Інтернету, або яка отримує вказівку, що ви збираєтеся для підключення до ненадійного сайту.

Також можна спробувати підірвати ці механізми довіри. Метод, який використовують зловмисники, буде залежати від конкретного механізму, який вони намагаються знищити. Зловмисники можуть змінити дозволу для файлів і каталогів або змінити реєстр, щоб підтримати порушення цих елементів управління [16]. Зловмисники також можуть створювати або красти сертифікати підпису коду, щоб завоювати довіру цільових систем [17].

1.2.2.19 Процес ін'єкції. Хакери впроваджувати код в процеси, щоб обійти безпеку процесу і, можливо, підвищити привілеї. Впровадження процесу – це засіб для досягнення довільного коду в адресному просторі окремого живого процесу. Запуск вашого коду в контексті іншого процесу може надати доступ до пам'яті процесу, системним / мережевих ресурсів і, можливо, до підвищених привілеїв. Виконання за допомогою впровадження процесу також може уникнути виявлення продуктами безпеки, оскільки виконання замасковано під законний процес.

Є багато різних способів впровадження коду в процес, багато з яких порушують законну функціональність. Ці реалізації існують для кожної основної ОС, але зазвичай залежать від платформи.

Більш складні шаблони можуть виконувати множинне впровадження процесів в сегментні модулі і додатково уникати виявлення за рахунок використання іменованих каналів або інших механізмів взаємодії між процесами (IPC) в якості каналу зв'язку.

1.2.2.20 Заплутані файли або інформація. Недобросовісні розробники спробувати ускладнити виявлення або аналіз виконуваного файлу або файлу шляхом шифрування, кодування або іншого обфускації його змісту в системі або при передачі. Це звичайна поведінка, яке можна використовувати на різних платформах і в мережі для обходу безпеки.

Корисні дані можуть бути стиснуті, заархівовані або зашифровані, щоб уникнути виявлення. Ця корисне навантаження може використовуватися під час початкового доступу або пізніше, щоб знизити ймовірність виявлення. Іноді може знадобитися дію користувача для відкриття і деобфускації/ декодування файлів або інформації для виконання користувачем. Від користувача також може знадобитися ввести пароль, щоб відкрити захищений паролем стиснений / зашифрований файл, наданий зловмисником [18]. Зловмисники також можуть використовувати стислі або заархівовані сценарії, такі як JavaScript.

Частини файлів також можуть бути закодовані, щоб приховати текстові рядки, які в іншому випадку допомогли б захисникам у виявленні [19]. Корисне навантаження також може бути розділена на окремі, здавалося б, нешкідливі файли, які виявляють шкідливі функції тільки при повторній збірці [19].

Вони можуть приховати команди, що виконуються з корисних даних або безпосередньо через інтерпретатор команд і сценаріїв. Змінні середовища, псевдоніми, символи і інша специфічна для платформи / мови семантика можуть використовуватися для обходу механізмів виявлення сигнатур і управління додатками [20].

1.2.2.21 Попереднє завантаження ОС. Цікавим випадком є використання механізмів попереднього завантаження ОС для забезпечення стійкості системи. Коли комп'ютер завантажується, прошивка і різні служби запуску завантажуються до операційної системи. Ці програми контролюють потік виконання до того, як операційна система вступить у володіння [21].

Розробники шкідливого ПЗ можуть перезаписувати дані в завантажувальних драйвери або прошивках, як BIOS (базова система введення / виведення) і уніфікований розширюваний інтерфейс мікропрограм (UEFI), щоб вони зберігалися в системах нижче операційної системи. Це може бути особливо складно виявити, так як шкідливе ПЗ на цьому рівні не буде виявлено засобами захисту на основі програмного забезпечення хоста.

1.2.2.22 Мережеве граничне з'єднання. Метод перетинання кордонів мережі, компрометуючи пристрої мережі периметра. Злом цих пристроїв може дозволити зловмиснику обійти обмеження маршрутизації трафіку, які інакше розділяли б довірені і ненадійні мережі.

Такі пристрої, як маршрутизатори і брандмауери, можуть використовуватися для створення кордонів між надійними і ненадійними мережами. Вони досягають цього, обмежуючи типи трафіку, щоб забезпечити дотримання політики організації в спробі знизити ризик, властивий таким з'єднанням. Формування трафіку може бути досягнуто шляхом заборони IP-адрес, портів рівня 4 або глибокої перевірки пакетів для ідентифікації додатків. Для участі в решті частини мережі ці пристрої можуть мати пряму адресацію або бути прозорими, але їх режим роботи не впливає на те, як зловмисник може їх обійти в разі компрометації.

Коли зловмисник отримує контроль над таким прикордонним пристроєм, він може обійти його застосування політики, щоб вільно дозволити зазвичай заборонений трафік через кордон довіри між двома різними мережами. Отримавши достатні права на пристрої, зловмисник може перелаштувати пристрій, щоб дозволити трафік, який він хоче, дозволяючи їм надалі досягати таких цілей, як управління і контроль через проксі-сервер з декількома вузлами або фільтрація даних шляхом дублювання трафіку. У тих випадках, коли прикордонний пристрій розділяє два окремих об'єкта, противник також може полегшити бічне переміщення в нове середовище жертви.

1.2.2.23 Зміна процесу автентифікації. Доволі простим, але не менш ефективним є метод при якому відбувається заміна механізмів і процесів автентифікації для доступу до облікового запису користувача або іншим чином небезпечний доступ до облікових записів. Процес автентифікації обробляється такими механізмами, як процес локального сервера автентифікації безпеки (LSASS) і диспетчер облікових записів безпеки

(SAM) в Windows або плагіни автентифікації (PAM) в системах на основі Unix, що відповідають за збір, зберігання і перевірка облікових даних.

Зловмисники можуть зловмисно змінити частину цього процесу, щоб або розкрити облікові дані, або обійти механізми автентифікації. Зламані облікові дані або доступ можуть використовуватися для обходу засобів контролю доступу, розташованих на різних ресурсах в системах в мережі, а також можуть використовуватися для постійного доступу до віддалених систем і доступним ззовні службам, таким як VPN, Outlook Web Access і віддалений робочий стіл.

1.2.2.24 Потік викрадення. Хакери виконувати свої власні шкідливі корисні навантаження, перехоплюючи спосіб роботи програм в операційних системах. Потік перехоплення може використовуватися для цілей зберігання, оскільки це перехоплений виконання може повторюватися з плином часу. Зловмисники також можуть використовувати ці механізми для підвищення привілеїв чи обходу засобів захисту, таких як засоби управління додатками або інші обмеження виконання.

Є багато способів, якими зловмисник може захопити потік виконання, в тому числі маніпулювати тим, як операційна система знаходить програми для запуску. Ви також можете перехопити, як операційна система знаходить бібліотеки для використання програмою. Місця, де операційна система шукає програми/ресурси, такі як каталоги файлів, а в разі Windows реєстр також може бути отруєний, щоб включити шкідливі корисні дані.

1.2.2.25 Приховування артефактів. Також можна спробувати приховати артефакти, пов'язані з їх поведінкою, щоб уникнути виявлення. Операційні системи можуть мати функції для приховування різних артефактів, таких як важливі системні файли і виконання адміністративних завдань, щоб не порушувати робоче середовище користувача і не дозволяти користувачам змінювати файли або функції в системі. Зловмисники можуть зловживати цими функціями, щоб приховати такі артефакти, як файли,

каталоги, облікові записи користувачів або інші дії системи, щоб уникнути виявлення [22].

Зловмисники також можуть спробувати приховати артефакти, пов'язані зі зловмисним поведінкою, шляхом створення обчислювальних областей, ізольованих від загальних інструментів безпеки, таких як технологія віртуалізації.

1.2.2.26 Маніпулювання токенами доступу. Розробники шкідливого ПЗ заміняють токени доступу для роботи в контексті безпеки іншого користувача або системи, щоб вжити заходів і обійти засоби контролю доступу. Windows використовує маркери доступу для визначення власника запущеного процесу. Користувач може маніпулювати токенами доступу, щоб запущений процес виглядав так, як ніби він є дочірнім по відношенню до іншого процесу або належить комусь іншому, крім користувача, що запустив процес. Коли це відбувається, процес також бере на себе контекст безпеки, пов'язаний з новим токеном.

Зловмисник може використовувати вбудовані API Windows для копіювання токенов доступу існуючих процесів; це відомо як крадіжка токенів. Потім цей токен можна застосувати до існуючого процесу (наприклад, уособлювати / вкрасти токен) або використовувати для створення нового процесу (наприклад, створення процесу з токеном). Щоб вкрасти токен, зловмисник вже мав би бути в контексті користувача root (тобто адміністратора). Однак зловмисники зазвичай використовують крадіжку токенов, щоб підняти свій контекст безпеки з рівня адміністратора до системного рівня. Потім зловмисник може використовувати токен автентифікації в віддаленій системі як обліковий запис цього токену, якщо обліковий запис має відповідні дозволи в віддаленій системі [23].

Будь-який стандартний користувач може використовувати команду runas і функції Windows API для створення токенів уособлення; для цього не потрібно доступ до облікового запису адміністратора. Існують і інші

механізми, такі як поля Active Directory, які можна використовувати для зміни токенів доступу.

1.2.2.27 Виконання огорож. Зловмисники можуть використовувати огорожі виконання, щоб обмежити виконання або дії в залежності від наданих зловмисником і умов навколишнього середовища, які, як очікується, будуть присутні на цілі. Поручні гарантують, що корисне навантаження діє тільки на намічену мету і знижує супутню шкоду від ворожої кампанії [24]. Значення, які зловмисник може надати про цільовій системі або середовищі, яка буде використовуватися в якості огорожі, можуть включати певні імена мережевих ресурсів, підключення фізичні пристрої, файли, домени, приєднані до Active Directory (AD), і локальні / зовнішні IP –адреси [24].

Захисні заходи можуть використовуватися для запобігання розкриття можливостей в середовищах, які не призначені для злому або використання. Таке використання захисних огороджень відрізняється від типового обходу віртуалізації / пісочниці. У той час як використання обходу віртуалізації / пісочниці може включати перевірку відомих значень пісочниці і продовження виконання тільки в тому випадку, якщо збігу немає, у застосуванні обмежень буде включати перевірку очікуваної мети і продовження виконання тільки при наявності такого збігу.

1.2.2.28 Крадіжка або підробка квитків Kerberos. Хакери можуть спробувати зламати автентифікацію Kerberos шляхом крадіжки або підробки квитків Kerberos, щоб дозволити передачу квитків.

Kerberos – це протокол автентифікації, широко використовуваний в сучасних доменних середовищах Windows. У середовищах Kerberos, званих «областями», є три основних учасника: клієнт, служба та Центр поширення ключів (KDC) [25]. Клієнти запитують доступ до служби, і через обмін квитками Kerberos, що виходять від KDC, їм надається доступ після успішної автентифікації. KDC відповідає як за автентифікацію, так і за випуск квитків.

1.2.3 Найвідоміші програми-вимагачі

На сьогоднішній день в області шкідливого програмного забезпечення є низка найбільш популярних програм, які, в свою чергу, наробили багато шкоди компаніям та користувачам. Кожна програма по своєму унікально реалізована, але спільними є як методи так і техніки захвату систем, обходу антивірусів та вимагання коштів з користувачів. Виділивши найбільш популярні із них та проаналізувавши, можна знайти основні відмінності, спільності та звернути увагу на наслідки атак такого шкідливого програмного забезпечення. Як і звичайні програми, данні хакерські творіння оновлюються обростаючи новою функціональністю, знаходячи нові лазівки та прикривають минулі недоліки, що не дали досягти більшого результату.

1.2.3.1 JCry – це програма-вимагач, написана на Go. Він був ідентифікований як учасник кампанії #OpJerusalem 2019. Данні зображені в таблиці 1.1.

Таблиця 1.1 – Данні програми-вимагача JCry

Метод/техніка	Опис
Шифрування даних для впливу	JCry зашифрував файли і вимагав біткоїни для їх розшифровки.
Запобігання відновленню системи	JCry видаляє тіньові копії, для гарантії, що дані не будуть відновлені легким шляхом.
Виконання користувачем: шкідливий файл	JCry пропонує користувачам клацнути на файл, який є начебто обновленням Adobe Flash Player.

Спливаюче вікно та зміст файлу з вимогами програми JCry зображено на рисунку 1.1:

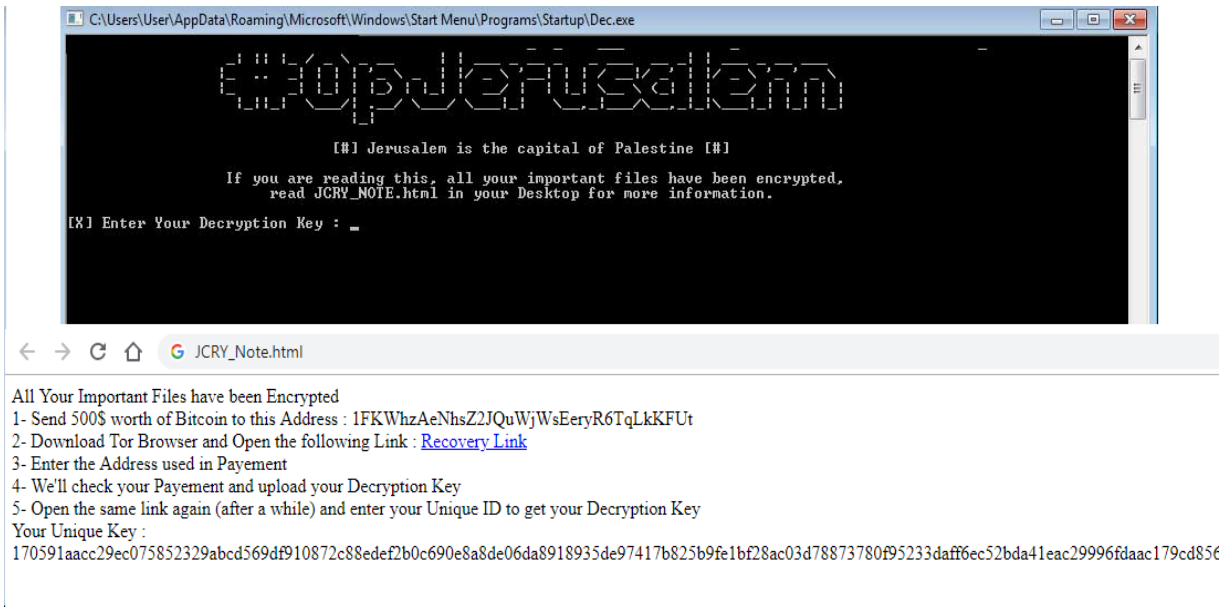

Рисунок 1.1 – Спливаюче вікно та зміст файлу з вимогами програми JCry

1.2.3.2 LockerGoga – це складна програма, що використовується для різних атак на європейські компанії. Вперше про це було оголошено в січні 2019 року. Дана програма завдала великої шкоди компаніям по всьому світу та вважається однією з найсерйозніших за останні роки.

Таблиця 1.2 – Данні програми-вимагача LockerGoga

Метод/техніка	Опис
Віддалений доступ до облікового запису	LockerGoga заміняє паролі облікових записів і виходить з системи поточними користувачами.
Шифрування даних для впливу	LockerGoga шифрує, включаючи файли ядра ОС Windows, за допомогою RSA-OAEP MGF1, а потім вимагає оплати в біткоїнах за ключ дешифрування.
Порушення захисту: відключення або зміна інструментів	Безпосередній установці LockerGoga передувала команда «kill task» для відключення антивірусу.

На рисунку 1.2 зображено вимоги програми-вимагача LockerGoga:

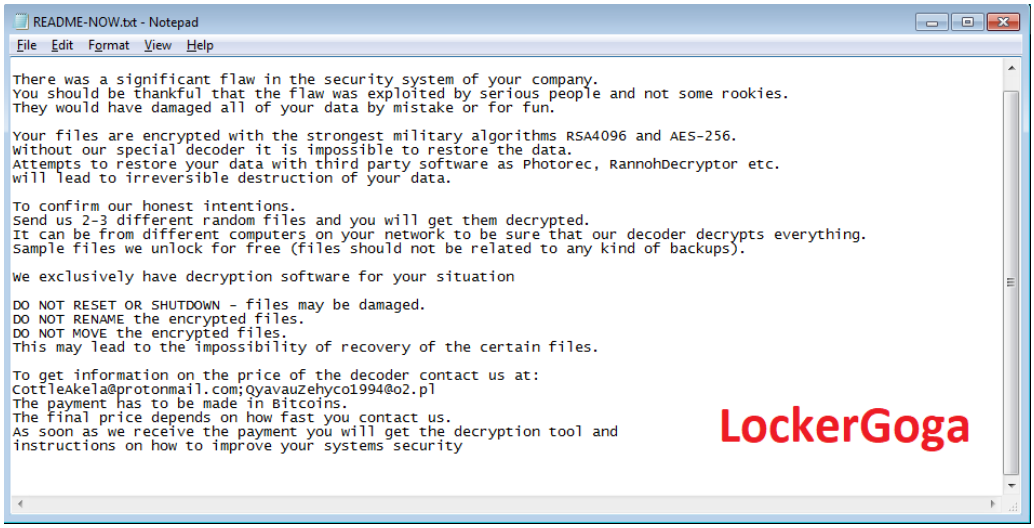

Рисунок 1.2 – README-NOW.txt файл з вимогами LockerGoga

1.2.3.3 Maze. Програма-вимагач Maze, раніше відома як ChaCha, була виявлена в травні 2019 року. Крім шифрування файлів на машинах жертви взлому, оператори Maze проводять кампанії по крадіжці інформації у постраждалих компаній. перед шифруванням і публікації інформації в Інтернеті для шантажу. Спочатку ця програма не виділялась нічим серед інших, маючи звичайне вікно для вимагання (рисунок 1.3), але через декілька місяців у неї уже був веб-сайт з даними користувачів (рисунок 1.4).

Деякі з необхідних викупів досягли мільйонів доларів. Повідомляється, що Maze зажадав 6 млн. доларів від виробника проводів і кабелів в Джорджії і 15 млн. доларів від неназваної організації після того, як група зашифрувала їх мережу. Але після того, як в березні COVID-19 був оголошений пандемією, Maze і інші групи програм-вимагачів пообіцяли не атакувати лікарні та медичні установи.

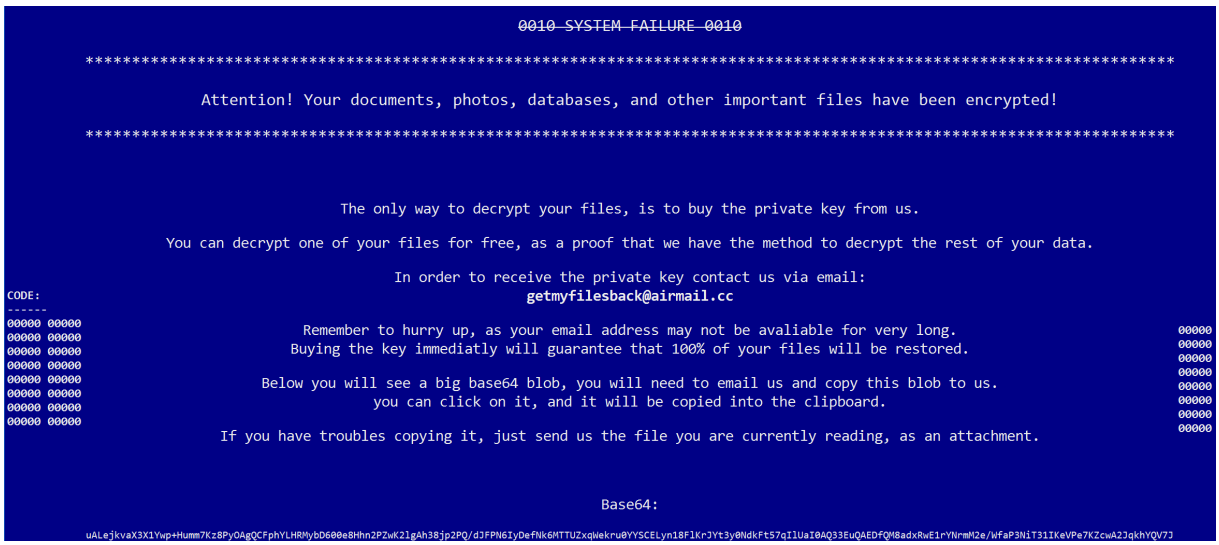

Рисунок 1.3 – Вікно програми-вимагача ChaCha

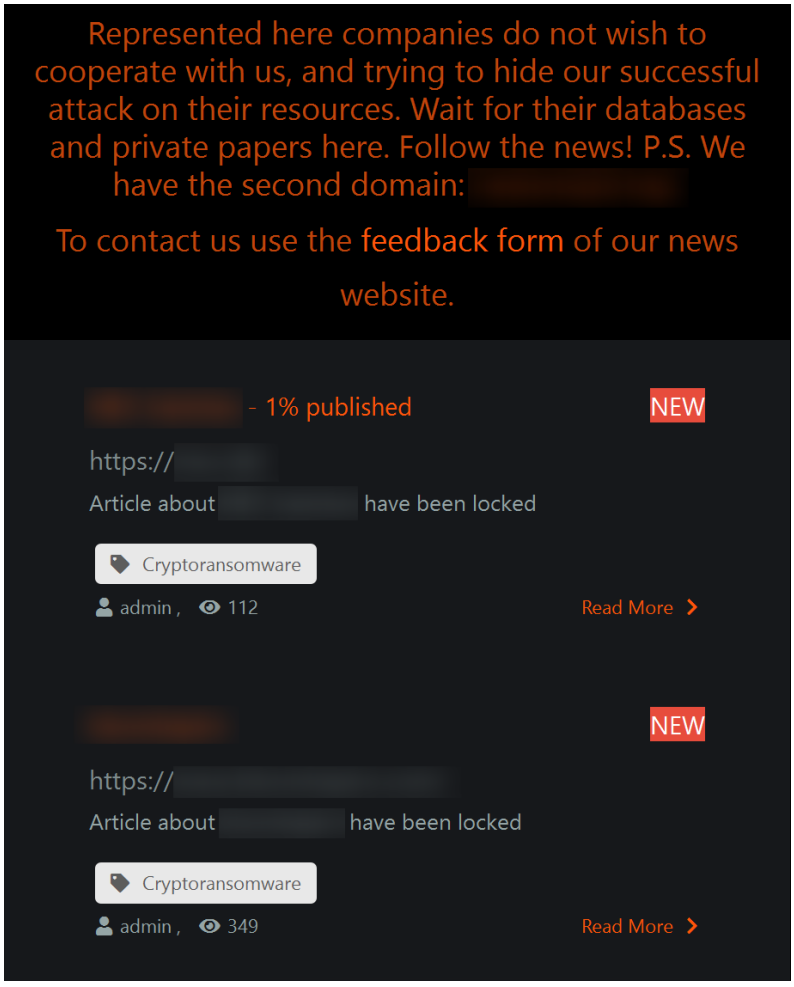

Рисунок 1.4 – Розділ на веб-сайті програми-вимагача Maze

Так як ця хакерська програма завдала стільки шкоди й досі являється однією з найнебезпечніших, то перелік методів та технік матиме великий

обсяг. В таблиці 1.3 зображено методи та техніки, які використовує програма Maze.

Таблиця 1.3 – Данні програми-вимагача Maze

Метод/техніка	Опис
Динамічне розширення	Maze підробляє рядки POST з випадковим вибором зі списку можливостей, включаючи «forum», «php», «view» і т.д., при встановленні з'єднання з C2, що ускладнює виявлення [26].

Продовження таблиці 1.3

Шифрування даних для впливу	Maze порушує роботу систем, зашифровуючи файли на цільових машинах, розшифровуючи файли в разі виплати викупу. Maze використовує алгоритм ChaCha на основі Salsa20 і алгоритм RSA для шифрування файлів.
Власний API	Maze використовує кілька функцій Windows API в процесі шифрування, включаючи IsDebuggerPresent, TerminateProcess, Process32FirstW і інші [26].
Інструментарій управління Windows	Maze використовує WMI, щоб спробувати видалити тіньові томи на машині і підключити віртуальну машину до мережного домену мережі організації-жертви [26].
Заборона відновлення системи	Maze намагається видалити тіньові томи заражених машин один раз до і один раз після процесу шифрування.

1.2.3.4 Netwalker – це безфайлова програма-здирник, написана на PowerShell, яка запускається безпосередньо в пам'яті. В таблиці 1.4 приведені методи та техніки, котрими користується дана хакерська програма.

Таблиця 1.4 – Данні програми-вимагача Netwalker

Метод/техніка	Опис
Передача інструменту входу	Оператори, розгортають Netwalker, використовуючи psexec і certutil для отримання навантаження [27].
Власний API	Netwalker може використовувати функції Windows API для впровадження бібліотек DLL вимагачів [27].

Продовження таблиці 1.4

Інтерпретатор команд і сценаріїв	Netwalker був написаний на PowerShell і запускається безпосередньо в пам'яті, уникаючи виявлення. Оператори, розгортають його, використовуючи пакетні сценарії для отримання корисного навантаження Netwalker [27].

На рисунку 1.5 зображено вимоги програми Netwalker під час нападу на K-Electric (постачальник електроенергії для Карачі (Пакистан)).

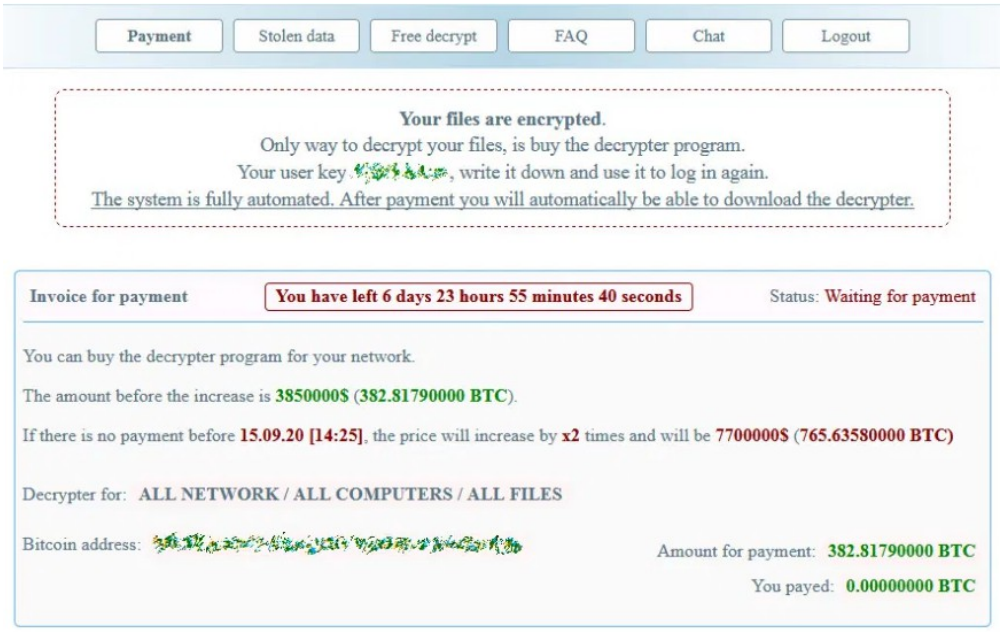

Рисунок 1.5 – Меню програми-вимагача Netwalker після зашифровки файлів

1.2.3.5 NotPetya – це шкідлива програма, яка була вперше виявлена в результаті всесвітньої атаки, що почалася 27 червня 2017. Основна мета шкідливої програми полягала в ефективному руйнуванні даних і дискових структур в скомпрометованих системах. Хоча NotPetya є різновидом програми-здирника, цілком ймовірно, що зловмисники ніколи не збиралися робити зашифровані дані доступними для відновлення. Таким чином, NotPetya більш доречно вважати типом шкідливого ПЗ. Він містить червоподібні функції для поширення по комп'ютерній мережі з використанням експлойтів SMBv1 EternalBlue і EternalRomance [28]. Методи та техніки даного небезпечного ПЗ відображено в таблиці 1.4.

Таблиця 1.5 – Данні програми-вимагача NotPetya

Метод/техніка	Опис
Шифрування даних для впливу	NotPetya шифрує призначені для користувача файли і дискові структури, такі як MBR, за допомогою 2048-бітного RSA [28].
Використання віддалених сервісів	NotPetya може використовувати два експлойта SMBv1, EternalBlue і EternalRomance, для поширення на інші віддалені системи в мережі [28].
Інструментарій управління Windows	NotPetya може використовувати wmic для поширення по мережі [28].
Запланована заздалегідь задача	NotPetya дає завдання перезавантажити систему через годину після зараження.
Стирання журналу Windows	NotPetya використовує wevtutil для очищення журналів подій Windows [28].

На рисунку 1.6 зображено вимоги даного вірусу, який припинив роботу багатьох підприємств у всьому світі (в Україні це були «ПриватБанк», «Азовсталь»).

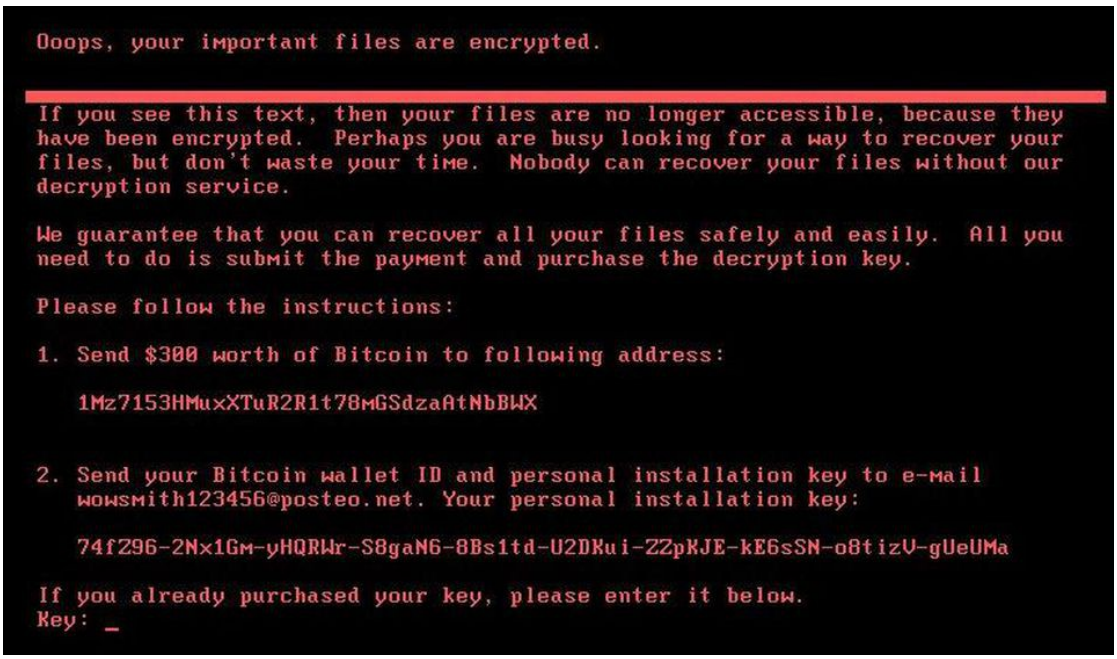

Рисунок 1.6 – Вікно з вимогами вірусу NotPetya

1.2.3.6 Ragnar Locker – це програма-вимагач, яка використовується як мінімум з грудня 2019 року. Вікно з вимогами зображено на рисунку 1.7.

Рисунок 1.7 – Вимоги програми-здирника Ragnar Locker

Ragnar Locker входить до альянсу розробників шкідливого програмного забезпечення, який вони, разом з Maze, заснували в 2020. В таблиці 1.6 описані техніки та методики, які використовує дана програма-вимагач.

Таблиця 1.6 – Данні програми-вимагача Ragnar Locker

Метод/техніка	Опис
Шифрування даних для впливу	Ragnar Locker шифрує файли на локальному комп'ютері і підключених дисках перед відображенням записки про викуп.
Зупинка служб	Ragnar Locker намагється зупинити служби, пов'язані з бізнес-додатками і базами даних, щоб розблокувати файли, для їх зашифровування
Командна оболонка Windows	Ragnar Locker використовує cmd.exe і пакетні скрипти для виконання команд
Служби Windows	Ragnar Locker використовує sc.exe для створення нової служби драйвера VirtualBox.
Запуск віртуальних екземплярів	Ragnar Locker використовує VirtualBox і урізану віртуальну машину Windows XP для запуску. Використання загальної папки, зазначеної в конфігурації, дозволяє Ragnar Locker шифрувати файли в операційній системі хоста, включаючи файли на будь-яких підключених дисках [29].

1.2.3.7 Xbash – це сімейство шкідливих програм, націлених на сервери Linux і Microsoft Windows. Шкідлива програма пов'язана з Iron Group, групою зловмисників, відомої своїми попередніми атаками програм-вимагачів. Xbash був розроблений на Python, а потім перетворений в автономний виконуваний файл ELF для Linux за допомогою PyInstaller [30]. В таблиці 1.7 описані данні XBash програми-здирника.

Таблиця 1.7 – Дані програми-вимагача XBash

Метод/техніка	Опис
Шифрування даних для впливу	Xbash зловмисно шифрує системи баз даних жертви і жадає викуп у криптовалюті.
Знищення даних	Xbash знищує бази даних на базі Linux.
Виконання операції клієнтом	Xbash може спробувати побачити утиліту в Hadoop, Redis або ActiveMQ [30]. Якщо ви бачите, що служба запущена, просто закрийте її.
Веб-протоколи	Xbash використовує HTTP для зв'язку C2.
Сканування мережевої служби	Xbash може виконувати сканування портів TCP і UDP [30].
Передача інструменту входу	Xbash може завантажувати додаткові шкідливі файли зі свого сервера C2 [30].

На рисунку 1.8 зображений приклад вікна програми XBash з вимогами до жертви.

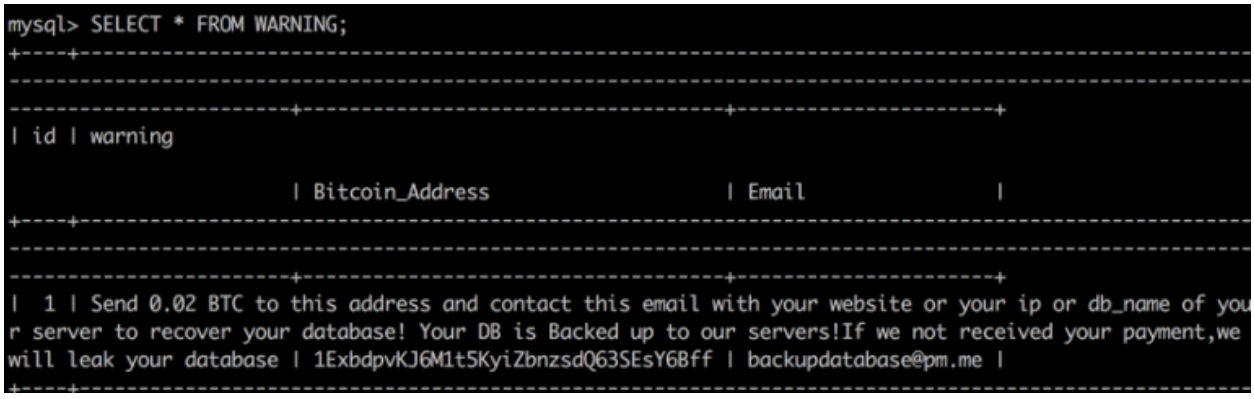

Рисунок 1.8 – Вимоги програми-вимагача XBash в базі даних жертви

1.2.3.8 REvil – це сімейство програм-вимагачів, пов'язаних з групою GOLD SOUTHFIELD і працюють способом програма-вимагач як послуга (RaaS) принаймні з квітня 2019 року. REvil володіє широкими можливостями налаштування і має код, аналогічний GandCrab RaaS. В таблиці 1.8 описані данні програм-здирників сімейства REvil.

Таблиця 1.8 – Дані програм-вимагачів сімейства REvil

Метод/техніка	Опис
Шифрування даних для впливу	REvil може шифрувати файли в системах жертв і вимагає викупу за розшифровку файлів.
Знищення даних	REvil може знищувати файли і папки.
Зашифрований канал (асиметрична криптографія)	REvil виконує шифрування зв'язку C2 за допомогою алгоритму ECIES [31].
Передача інструменту входу	REvil може завантажити свою копію з IP-адреси зловмисника на машину жертви.
Заборона відновлення системи	REvil може використовувати vssadmin для видалення тіньових копій томи і bcdedit для відключення функцій відновлення [31].

Файл з вимогами однієї із шкідливих програм із сімейства REvil зображено на рисунку 1.9.

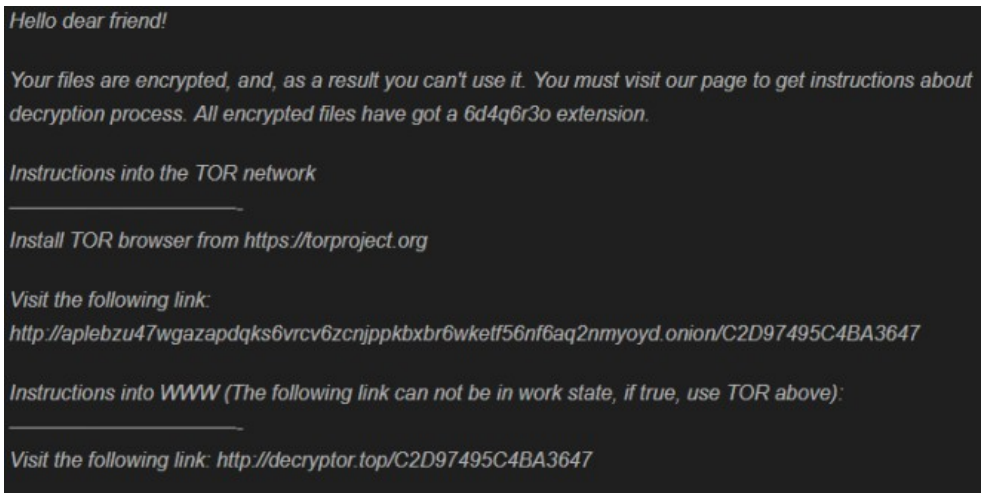

Рисунок 1.9 – Вимоги одного із програм-здирників з сімейства REvil

1.2.3.9 RobinHood – це програма-вимагач, яка вперше була помічена при атаці на комп'ютерну мережу уряду міста Балтімор. Така назва обумовлена тим, що розробники даного шкідливого програмного забезпечення начебто зробили його для збору коштів для населення Ємену, але данні висновки та теорії нічим не обґрунтовані. В таблиці 1.9 зображено

методи та техніки, які застосовувались під час нападу RobinHood програми-вимагача.

Таблиця 1.9 – Дані програми-здирника RobinHood

Метод/техніка	Опис
Шифрування даних для впливу	RobinHood спочатку шукає ключ шифрування RSA, а потім виконує процес шифрування системних файлів.
Вимикання або заміна інструментів	Робінгудів виконує пошук служб Windows (181), пов'язаних з антивірусним програмним забезпеченням, в системі і завершує процеси.
Зупинка служб	RobinHood зупиняє 181 службу Windows в системі перед запуском процесу шифрування.
Заборона відновлення системи	RobinHood видаляє тіньові копії, щоб гарантувати, що всі дані не можуть бути легко відновлені.
Командна оболонка Windows	RobinHood відключає всі мережеві ресурси від комп'ютера за допомогою команди net use*/ DELETE/Y [32].

На рисунку 1.10 зображено вікно програми-вимагача RobinHood, яке користувач бачить після повного захвату системи.

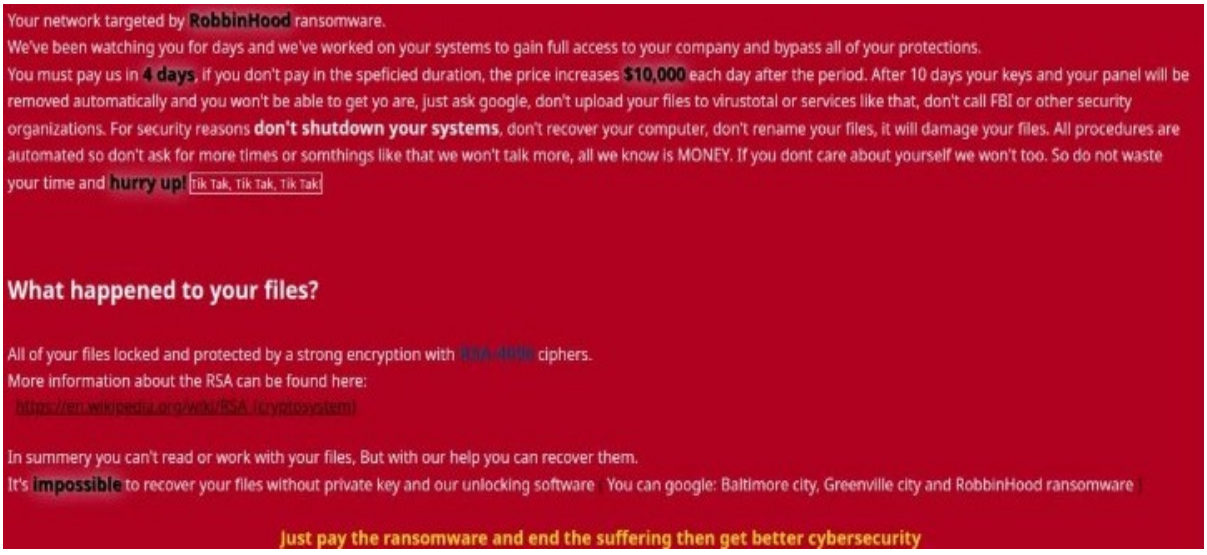

Рисунок 1.10 – Вимоги програми-здирника RobinHood

1.2.3.10 SamSam – це програма-вимагач, яка була представлена на початку 2016 року. На відміну від деяких програм-вимагачів, для реалізації деяких з її основних компонентів оператори повинні були вручну взаємодіяти з шкідливою програмою. Методи та техніки SamSam наведені в таблиці 1.10.

Таблиця 1.10 – Дані програми-здирника SamSam

Метод/техніка	Опис
Шифрування даних для впливу	SamSam шифрує файли жертв з використанням шифрування RSA-2048 і вимагає викуп в біткоїнах для розшифровки цих файлів.
Заплутані файли або інформація	Було відзначено, що SamSam використовує AES або DES для шифрування корисних даних і компонентів корисного навантаження.
Видалення своїх файлів	SamSam видаляє свої власні файли і корисні дані, щоб утруднити аналіз атаки.
Бінарне заповнення	SamSam використовує небажаний код для заповнення деяких своїх шкідливих компонентів.

На рисунку 1.11 зображено вікно з вимогами програми SamSam

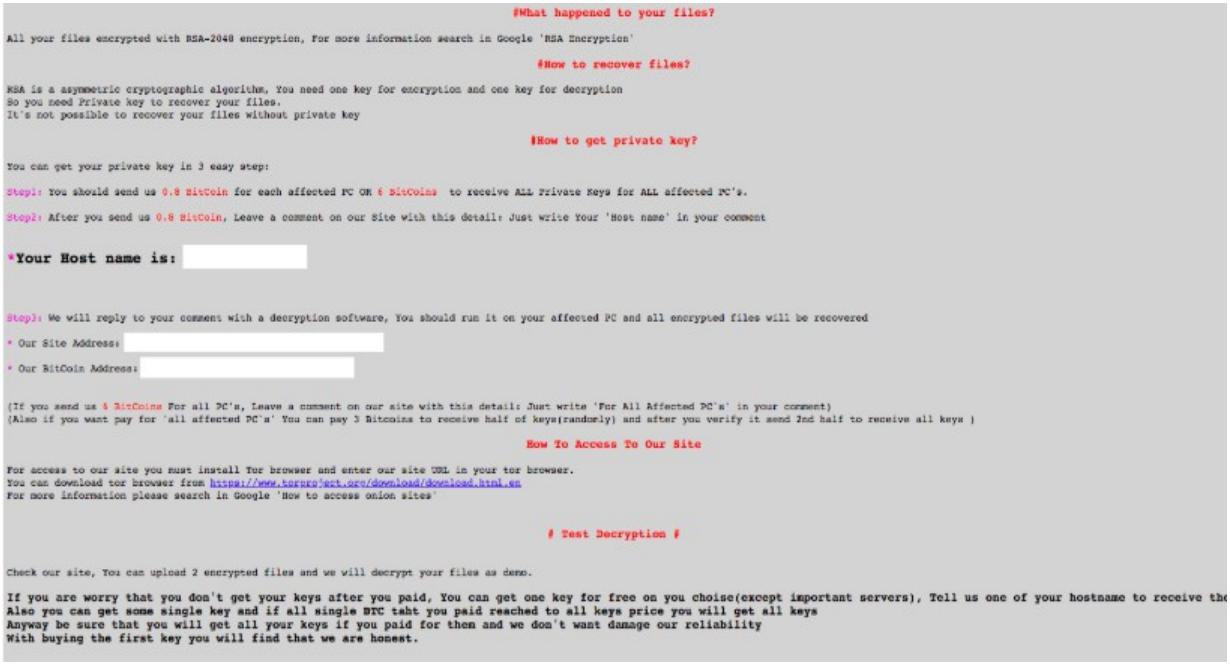

Рисунок 1.11 – Вікно з вимогами шкідливого ПЗ SamSam

1.2.3.11 WannaCry – це програма-вимагач, яка вперше була помічена в результаті глобального атаки в травні 2017 року, що торкнулася понад 150 країн.. На рисунку 1.12 зображено вимоги програми-здирника WannaCry.

Рисунок 1.12 – Вікно програми WannaCry з вимогами

Дана програма містить червоподібні функції для поширення по комп'ютерній мережі за допомогою експлойта SMBv1 EternalBlue. В таблиці

описані техніки та методи, які приміняються при захваті системи жертви та зашифровуванні файлів.

Таблиця 1.11 – Дані програми-здирника WannaCry

Метод/техніка	Опис
Шифрування даних для впливу	WannaCry шифрує файли жертви і вимагає сплати викупу в біткоїнах для розшифровки цих файлів.
Зашифрований канал: асиметрична криптографія	WannaCry використовує Tor для управління трафіком і направляє налаштовується криптографічний протокол по каналу Tor.
Використання віддалених сервісів	WannaCry використовує експлойт в SMBv1 для поширення на інші віддалені системи в мережі.
Пошук файлів і каталогів	WannaCry шукає файли з розширення перед їх шифруванням за допомогою RSA і AES, включаючи Office, PDF, зображення, аудіо, відео, вихідний код, а також файли ключів і сертифікатів.
Виявлення периферійних пристроїв	WannaCry містить потік, який буде намагатися сканувати нові підключені диски кожні кілька секунд. У разі ідентифікації він зашифрує файли на підключеному пристрої.
Створення або зміна системного процесу	WannaCry створює службу «mssecsvc2.0» з наведених ім'ям «Microsoft Security Center Service (2.0)».
Виявлення віддаленої системи	WannaCry сканує свій сегмент локальної мережі на наявність віддалених систем, щоб спробувати використовувати їх і скопіювати себе на них.

1.2.3.12 Rotexy – це банківська шкідлива програма для Android, яка розвивалася роками. Спочатку це був троян-шпигун SMS, вперше виявлений в жовтні 2014 року, і з тих пір він розширився, включивши в нього більше

функцій, включаючи функції вимагачів. В таблиці 1.12 перечисленні дані Rotexy.

Таблиця 1.12 – Дані програми-вимагача Rotexy

Метод/техніка	Опис
Перехоплення SMS-повідомлень	Rotexy обробляє вхідні SMS-повідомлення шляхом фільтрації за номерами телефонів, ключовими словами і регулярними виразами, приділяючи особливу увагу банкам, платіжним системам і операторам мобільного зв'язку. Rotexy також може відправляти список всіх SMS-повідомлень на пристрої на сервер управління і контролю.
Блокування пристрою	Rotexy може блокувати HTML-сторінку на передньому плані, вимагаючи від користувача введення інформації кредитної картки, яка повинна співпадати з інформацією, раніше перехопленої в SMS-повідомленнях, наприклад, останні 4 цифри номера кредитної картки. При виявленні спроб входу в режим адміністратора, Rotexy періодично відключає екран телефону, щоб запобігти видаленню дозволів.
Виявлення системної інформації	Rotexy збирає інформацію про зламаний пристрій, включаючи номер телефону, оператора мережі, версію ОС, модель пристрою і країну реєстрації пристрою.
Вхідна підказка	Rotexy може використовувати фішингові оверлеї для збору інформації про кредитні картки користувачів.

Rotexy програма-здирник орудує більш за все на території Росії і весь дисонанс відбувся там. На рисунку 1.13 зображено блокуюче вікно в браузері на телефоні жертви.

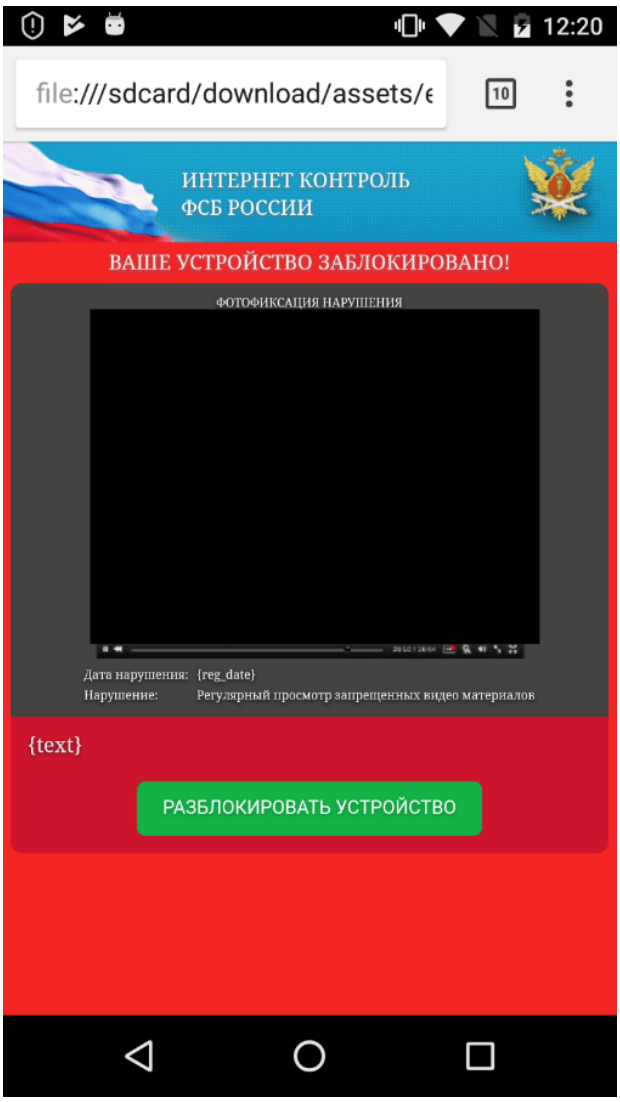

Рисунок 1.13 – Блокуюче вікно браузеру на телефоні під час атаки Rotexy

1.2.3.13 SynAck – це варіант троянської програми-здирника, націленої в основному на англомовних користувачів, принаймні, з осені 2017 року. В таблиці 1.13 приведені методи та техніки, які використовує програма-вимагач SynAck.

Таблиця 1.13 – Дані програми-здирника SynAck

Метод/техніка	Опис
Шифрування даних для впливу	SynAck шифрує машину жертви, а потім просить жертву заплатити викуп.

Виявлення файлів і каталогів	SynAck перевіряє розташування свого каталогу, щоб уникнути запуску в ізольованому програмному середовищі.
Виявлення системної інформації	SynAck збирає імена комп'ютерів, інформацію про версію ОС, а також перевіряє встановлені розкладки клавіатури, щоб дізнатися, чи був він запущений з певного списку країн.
Власний API	SynAck аналізує таблиці експорту системних DLL, щоб знайти і викликати різні функції Windows API.

На рисунку 1.14 зображений текстовий файл з вимогами програми-здирника SynAck.

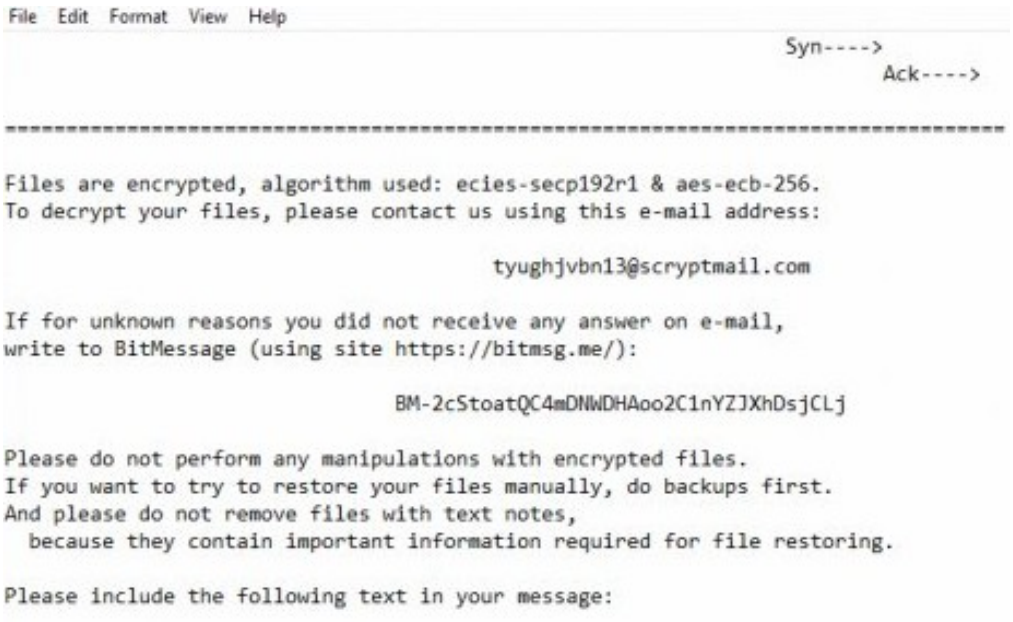

Рисунок 1.14 – Текстовий файл програми-вимагача SynAck після атаки

2 ПІДГОТОВКА ТЕСТОВИХ ДАНИХ

2.1 Налаштування віртуальної машини

Щоб не завдати шкоди особистій обчислювальній машині вибір пав на крос-платформний додаток Oracle VM VirtualBox для створення, керування та запуску віртуальних машин, який є безкоштовним та має відкритий вихідний код. Віртуальні машини – емуляція реального комп'ютеру, який працює на базі реального комп'ютеру. Система заздалегідь налаштовується для виконання певних задач. Наприклад обирається ОС, кількість оперативної пам'яті, тощо. На рисунку 2.1 зображено початкове меню VirtualBox з описом системи на котрій відбудеться запуск програми-вимагача. Перевагою Oracle VirtualBox є те, що даний крос-платформний додаток дозволяє налаштувати декілька віртуальних машин на одній реальній фізичній машині і використовувати їх одночасно разом на основній(з основною). Підтримка основних операційних систем, включаючи версії Microsoft Windows, Linux, Ubuntu, BSD і MS-DOS.

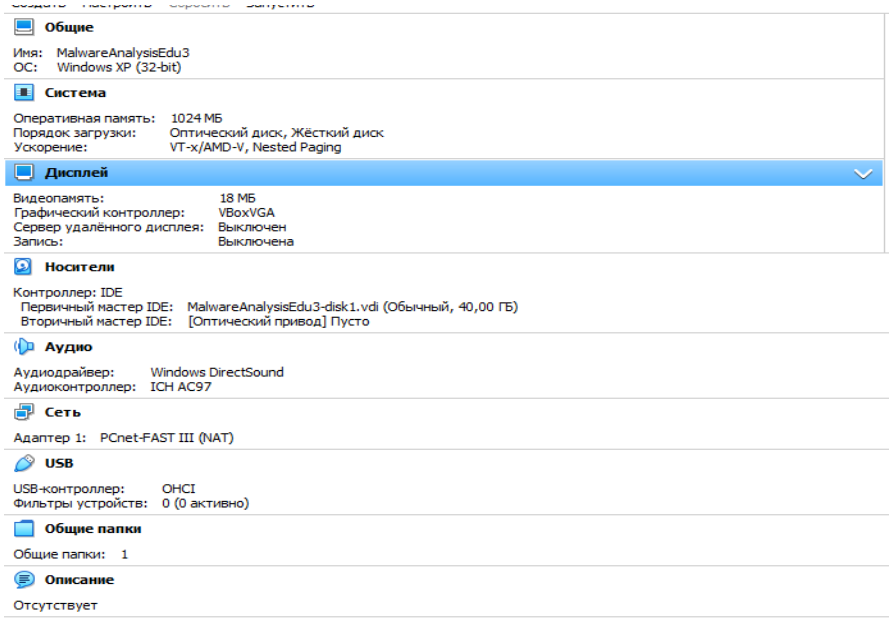

Рисунок 2.1 – знімок віртуальної машини в Oracle VM VirtualBox

З усих перечислених вище систем було обрано Windows XP, так як вона є останньою з систем корпорації Microsoft в якій не вбудовано антивірус. Данне підготування можна обгрунтувати тим, що шкідливе програмне забезпечення є дуже непередбачуванним й за допомогою утиліти Oracle VM VirtualBox, при найменшому, непропрацьованному заздалегідь, сценарію повернути обчислювальну машину до потрібного стану.

2.2 Запуск програми-вимагача. Підготовка тестових файлів.

Для зашифровування файлів були обрані реальні образи програм-вимагачів. На порталі VirusTotal було знайдено декілька подібних програм, а саме: CryptoLocker(далі X.exe) та DeriaLocker (далі D.exe), файли запуску яких зображені на рисунку 2.2. Для контролю запуску шкідливого програмного запезбечення, вони були переконвертовані в .exe файли, але розширення файлів було змінено на .ex_ для уникнення випадкоговго запуску програми-вимагача.

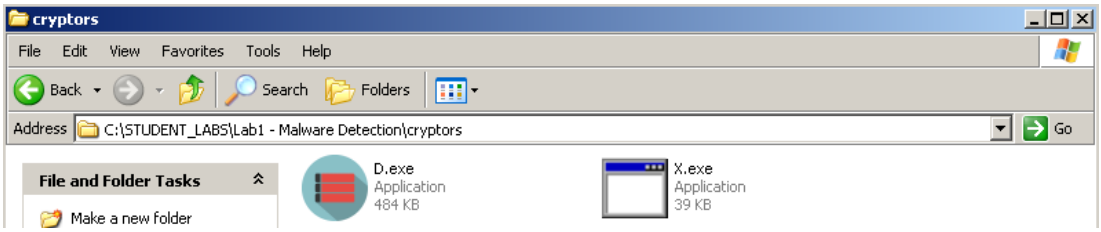

Рисунок 2.2 – файли запуску програм вимагачів

Після запуску X.exe вискочило вікно з повідомленням (рисунок 2.3) та вікно с полем для вводу пароля для розшифрування(який можна отримати тільки після виконання всіх вимог злодія-розробника, але і це не гарантує видачу ключа), що зображене на рисунку 2.4.

Рисунок 2.3 – повідомлення з вимогами

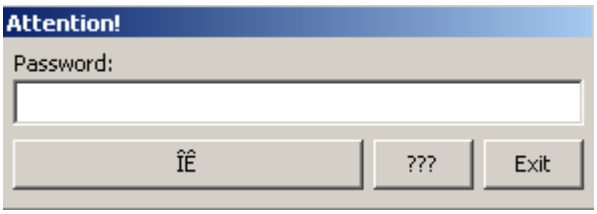

Рисунок 2.4 – поле для вводу ключа для розшифрування

При закриті всіх спливаючих вікон можна помітити, що файли насправді зашифровані. В кожній папці присутній файл HOW TO DECRYPT FILES.txt, в середині котрого знаходиться інструкція, яка зображена на рисунку 2.5.

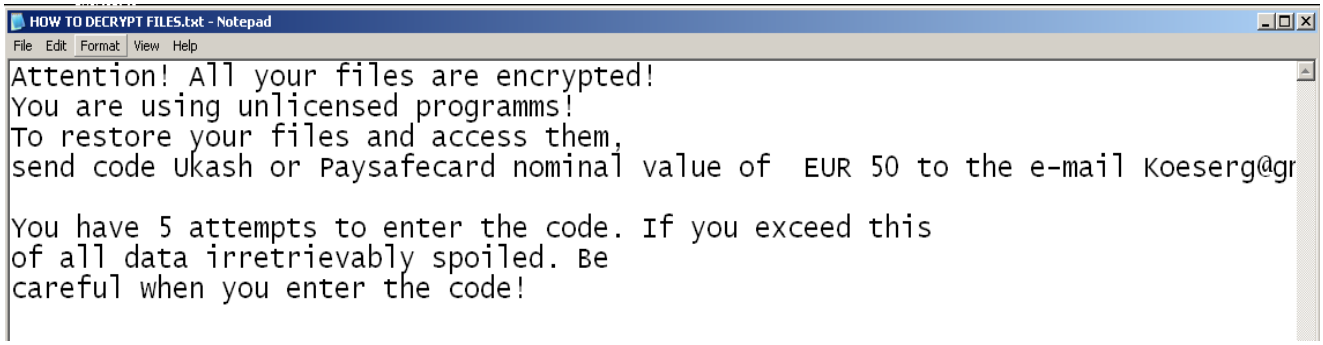

Рисунок 2.5 – інструкції в файлі HOW TO DECRYPT FILES.txt

На рисунку 2.6. зображено робочий стіл до атаки шифрувальника та після.

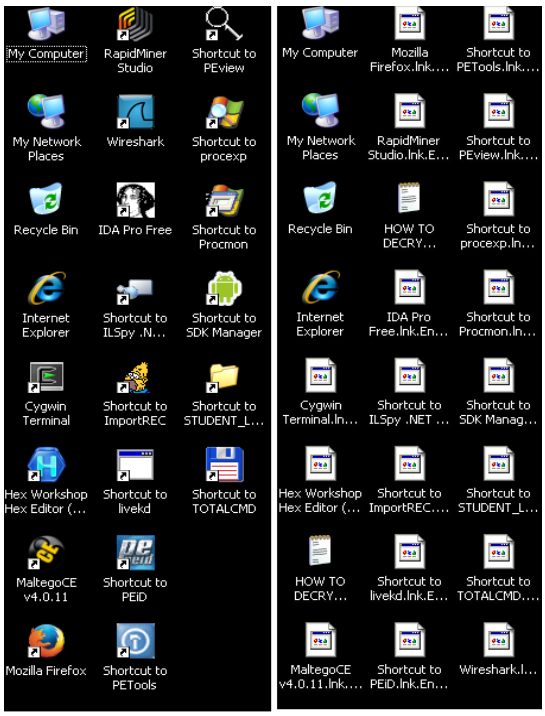

Рисунок 2.6 – робочий стіл «ДО» і «ПІСЛЯ» атаки програми-вимагача

Другим варіантом шкідливого програмного забезпечення є програма-вимагач DeriaLocker, який при запуску одразу захоплює обчислювальну машину та виводить вікно-повідомлення, яке зображене на рисунку 2.7.

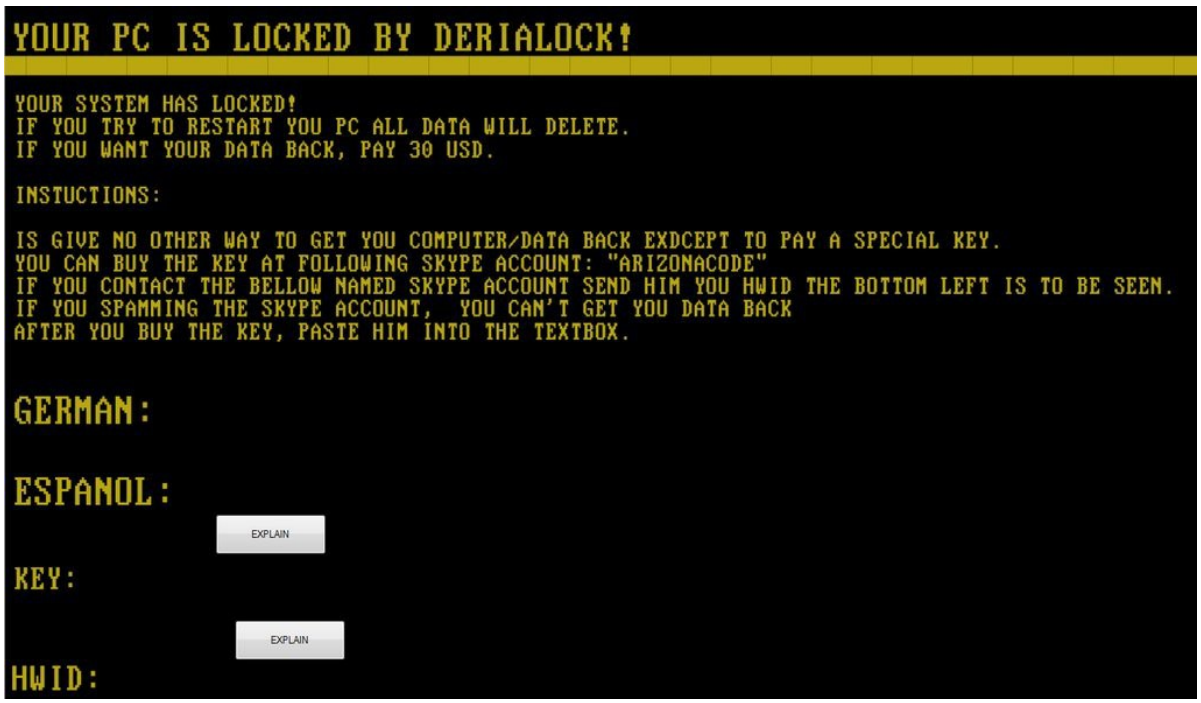

Рисунок 2.7 – повідомлення з інструкціями програми DeriaLocker

Закрити дане вікно неможливо(є можливість спробувати закрити за допомогою комбінації клавіш Alt + F4, але злочинець відпрацював цей «план відходу» і при спробі виводиться вікно, яку зображено на рисунку 2.8.), але в спільній папці віртуальної та реальної машини заздалегідь підготовлені файли будуть зашифровані з вторинним розширенням .deria.

Рисунок 2.8 – повідомлення при використанні комбінації клавіш Alt + F4 для виходу з вікна програми-вимагача DeriaLocker

Для більш продуктивної роботи програмної реалізації, тестові набори потрібно сформувати на реальній машині. Таким чином, після атаки шкідливого програмного забезпечення, зашифровані файли з віртуальної машини, через спільну папку, передаються до основної машини як тестові данні.

3 МЕТОДИ ВИЯВЛЕННЯ АКТИВНОСТІ ШИФРУВАЛЬНИКА

Розробники шкідливого програмного забезпечення, включаючи програми-вимагачі, навчились обходити різного роду заходи безпеки. На даний момент антивіруси та інші системи запобігання злому використовують так звані два підходи до аналізу файлів, які поступають на обчислювальну машину:

– статичний аналіз (аналіз структури двійкового файлу, його атрибутів, логічних структур, потоку виконання та даних);

– динамічний аналіз (відстеження дій програми під час виконання, побудова її профілю).

Кожен спосіб має свої переваги та недоліки. Тому їх зазвичай використовують одночасно для кращого виявлення шкідливих програм. Але навіть працюючи воєдино, є ймовірність помилкового спрацювання та пошкодження чистого файлу.

Динамічний аналіз дозволяє уникнути запуску шкідливих двійкових файлів. Наприклад, автори програм-здирників широко використовують системи упаковки, кодування та декодування даних, а також видаляють функції управління та потоки. Але ті ж методи використовуються розробниками для створення додатків інтелектуальної власності, які ускладнюють зворотне проектування. Метод визначає декілька основних дій, таких як видалення файлу, запис у файл, спілкування з мережею, відкриття порту прослуховування, надсилання електронних листів тощо. Цей профіль файлу та його діяльність перевіряються експертом або методами машинного навчання, щоб визначити, чи є зразок зловмисним. Однак динамічний аналіз можливий лише тоді, коли виконується тестовий код, що робить операційну систему більш вразливою, а деякі віруси можуть виявляти робоче

середовище, маскуватися і, таким чином, поводитися по-різному в тестовому та виробничому середовищах.

Статичний аналіз доповнює динамічний аналіз, надаючи інформацію про атрибути файлу. Статичний метод аналізує програму перед виконанням, витягує атрибути з бінарного файлу, обчислює статистику і на основі цієї інформації робить вирок щодо загрози сканованого файлу. Такий підхід безпечний – вирок видається перед виконанням файлу, але він погано працює із згорнутими файлами, упакованими в розділи. Крім того, із збільшенням розміру файлу збільшується час, необхідний для аналізу. Також, потрібно ідеально розуміти як працює навантажувач щоб реалізувати гарний алгоритм для статичного аналізу, а також розуміти різницю між документацією та фактичною поведінкою навантажувача. Віруси можуть використовувати деякі поля у двійковому файлі для власного використання. Наприклад, як сховище даних або адрес для виконання зловмисного коду.

Можна виділити та розглянути три основних методи статичного аналізу даних:

– метод вторинного розширення;

– метод перевірки байтів сигнатури типу файлу на початку файлу;

– метод обчислення значення ентропії.

3.1 Метод вторинного розширення

Під час більшості нападів програм-здирників, вони, в свою чергу, створюють вторинне розширення для кожного файлу. Робиться це з очевидної причини, а саме для того, щоб знати які файли вже зашифровані. Найчастіше розробники програм-вимагачів роблять вторинне розширення своєю «візитною» та надають його файлам таку ж назву як і сама хакерська програма. Таким чином можна легко з’ясувати чи є файл зашифрованим та зробити висновок щодо активності шифрувальника в системі.

Метод вторинного розширення базується на пошуку вторинних розширень, які найчастіше додають шифрувальники в файлах по типу .crypto, .encrypto і тд..

Для початку потрібно сформулювати вторинні розширення, по котрим потрібно шукати збіги. В таблиці 3.1 внесені розширення, котрі, розробники шкідливого програмного забезпечення, додають до зашифрованих файлів.

Таблиця 3.1 – розширення зашифрованих файлів

Назва програми	Розширення
Blower Ransomware Encrypted File	.blower
Locked Ransomware Encrypted File	.locked
Adame Ransomware Encrypted File	.adame
GandCrab Ransomware Encrypted File	.gdcb
Cerber Ransomware Encrypted File	.cerber
Jaff Ransomware Encrypted File	.wlu
Locky Ransomware Encrypted File	.locky
Litar Virus Encrypted File	.litar
XTBL Ransomware Encrypted File	.xtbl
Dharma Ransomware Encrypted File	.bip
Unknown Ransomware Encrypted File	.bc5b
Unknown Ransomware Encrypted File	.zzzzz
CryptXXX Ransomware Encrypted File	.crypt
GandCrab Ransomware Encrypted File	.gdcb
GandCrab V4 Ransomware Encrypted File	.krab
Wana Decrypt0r 2.0 Encrypted File	.wncry
Dharma Ransomware Encrypted File	.adobe
Cerber2 Ransomware Encrypted File	.cerber2
Lilocked Ransomware Encrypted File	.lilocked
Extractor Ransomware Encrypted File	.xxx
Wana Decrypt0r 2.0 Encrypted File	.wcry
WannaCry Virus Encrypted File	.wnry

Продовження таблиці 3.1

CryptoLocker Ransomware	.EnCiPhErEd
DeriaLocker Ransomware Encrypted File	.deria
Zepto Virus File	.zepto
Wallet Ransomware	.wallet

Сформульовані данні, в подальшому, будуть служити базою даних для реалізації першого методу.

Алгоритм для зчитування вторинного розширення файлу реалізований в декілька етапів:

1) представити ім’я файлу в виді рядку та перейти в його кінець;

2) зчитати символи до першої крапки і зберегти в кеш та перевірити наявність другої крапки:

a) якщо умова другої крапки позитивна, то перейти до третього пункту;

b) якщо умова щодо другої крапки негативна, то файл можна вважати незашифрованим за цим методом;

3) прогнати символи до першої крапки по таблиці шкідливих вторинних розширень:

a) якщо символи вторинного розширення файлу співпадають з розширеннями в таблиці 3.1, то файл можна вважати зашифрованим;

b) якщо збігів не знайдено, то файл можна вважати зашифрованим, але безпечною програмою.

На лістингу 3.1 зображена реалізація представлення імені файлу як рядок та лічильник для переходу між крапками:

Лістинг 3.1 – представлення імені файлу в виді рядка та лічильник

```
def checkFileEncryptExtension(FILE):
    fileNameList = list(FILE)
    extensionCount = 0
    i = len(fileNameList) - 1
    while i > 0:
        if fileNameList[i] == ".":
            extensionCount += 1
            if extensionCount == 2:
                return True
        i -= 1
    return False
```

З тестової вибірки файлів виділимо файли з шкідливим вторинним розширенням та розбавимо їх звичайними незашифрованими файлами, зображені в таблиці 3.2.

Таблиця 3.2 – Тестовий набір для тестування методу вт. розширення

Назва файлу	Розширення
1111	jpg.crypted
2	jpg.EnCiPhErEd
README	txt. EnCiPhErEd
qwerty	docx.EnCiPhErEd
documentation	docx.EnCiPhErEd
NEWS	txt. EnCiPhErEd
nio	png.deria
VBox	txt. EnCiPhErEd
Homework_SQL_P1	.sql
Homework_SQL_P2	.sql
rdpts	.pdf

На рисунку 3.1 зображені файли тестового набору:

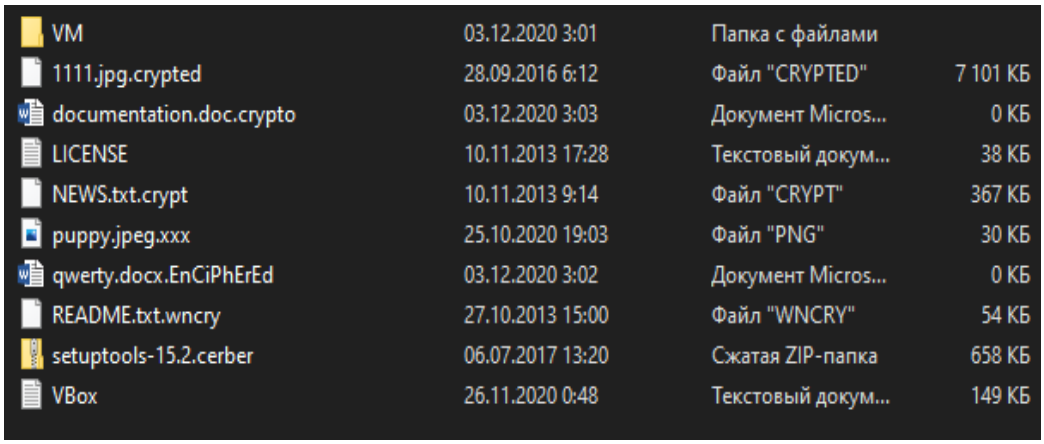

Рисунок 3.1 – файли тестового набору

Програмна реалізація методу спочатку просить ввести шлях до папки для сканування. Потім виводить список всіх файлів, котрі знаходяться в папці (рис. 2.10).

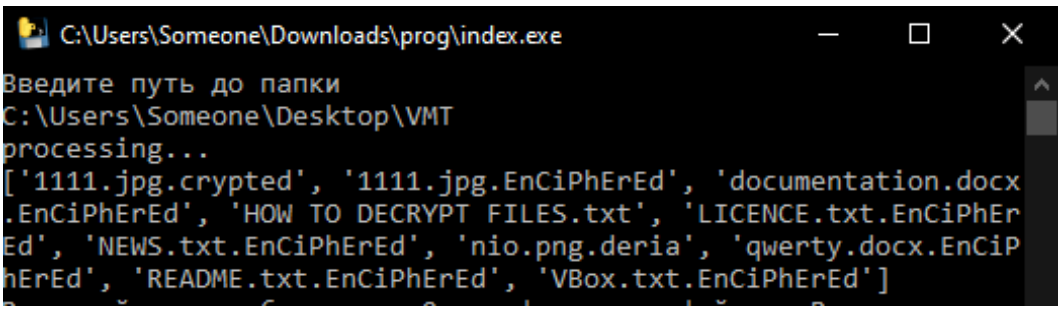

Рисунок 3.2 – вікно програмної реалізації методу

Після обробки файлів програма виводить кількість зашифрованих файлів та список із зашифрованих файлів з їх розширеннями (рис. 2.11).

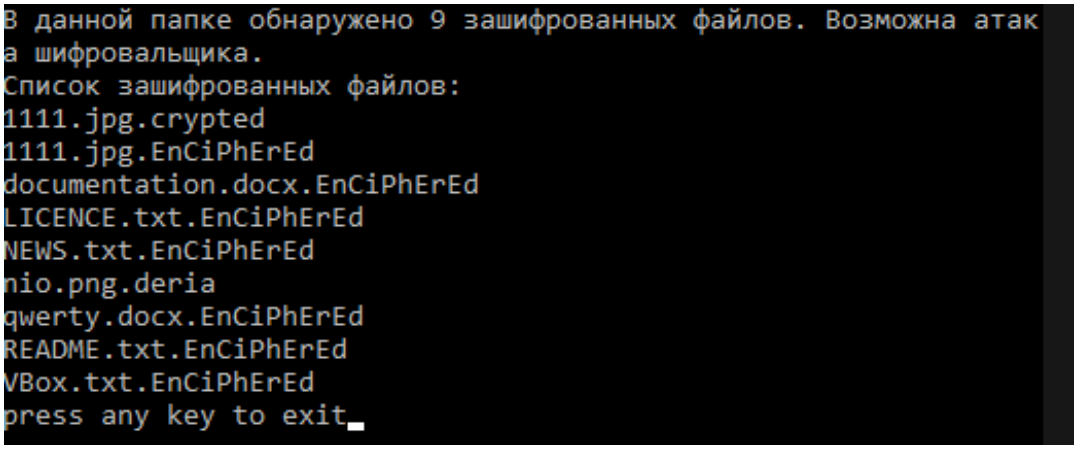

Рисунок 3.3 – вікно програмної реалізації методу

На цьому етапі реалізацію методу виявлення зашифрованих файлів можна вважати закінченою.

З плюсів даного методу можна виділити:

– швидкість;

– простота реалізації;

– точність.

Мінусом даного методу є те що, при відсутності оновлення бази даних шкідливих розширень, вищеописана програма не буде відрізнятись актуальністю і її ефективність з часом може впасти до нуля.

3.2 Метод перевірки байтів сигнатури типу файлу на початку файлу

Дані байти ще називають «магічними». Магічний байт – це не що інше, як перші кілька байтів файлу, які використовуються для його розпізнавання. Його не видно, якщо відкрити файл. Але це можна побачити за допомогою деяких спеціальних інструментів. Таким чином, зчитуючи байти сигнатури на початку файлу, система розуміє якою програмою потрібно скористатись щоб відкрити файл. Наприклад файл з розширенням .doc має 5 варіантів сигнатур (таблиця 3.3), а файл з розширенням .docx лише 2 (таблиця 3.4). Після зашифровування файлу байти сигнатури стираються або перетворюються на не зрозуміло що. Таким чином даний метод може допомогти краще знаходити зашифровані файли лише за допомогою однієї бібліотеці в Python.

Таблиця 3.3 – Сигнатури файлів формату .doc

Сигнатура	Опис
D0 CF 11 E0 A1 B1 1A E1	Microsoft Office document
0D 44 4F 43	DeskMate Document

Продовження таблиці 3.3

CF 11 E0 A1 B1 1A E1 00	Perfect Office document
DB A5 2D 00	Word 2.0 file
EC A5 C1 00	Word document subheader

Таблиця 3.4 – Сигнатури файлів формату .docx

Сигнатура	Опис
50 4B 03 04	MS Office Open XML Format Document

50 4B 03 04 14 00 06 00	MS Office 2007 documents

Метод на основі перевірки байтів сигнатури типу файлу на початку файлу базується на так званих магічних байтах, котрі використовуються для класифікації підписів загальних форматів файлів, які можна визначити, переглянувши перші кілька байтів розглянутого файлу.

Для початку треба підключити невелика бібліотеку для визначення типу файлу на основі підпису заголовка `from magic import magic`.

Далі просто зчитуємо перші 2048 байтів, так як цього буде достатньо для того, щоб дізнатися повний тип файлу. Якщо ж в даній бібліотеці збігів не буде знайдено, то файл можна вважати зашифрованим, або невизначеного типу. Реалізацію даного методу реалізовано в лістингу, приведеному нижче.

Лістинг 3.2 – Реалізація методу пошуку зашифрованого файлу за допомогою magic bytes

```
def checkFileTypeSignature(FILE):
    with open(FILE, "rb") as FILE:
        byteArr = FILE.read(2048)
    return magic.from_buffer(byteArr)
```

Цей метод також має свої плюси та мінуси. Із плюсів можна виділити:

– легкість реалізації;

– надійність;

– швидкість.

Мінусом ж можуть бути такі ситуації, коли розробники шкідливого програмного забезпечення передбачають подібні методи виявлення і вносять правки до свого алгоритму.

3.3 Метод обчислення значення ентропії

Якщо ми проаналізуємо як стислі, так і зашифровані фрагменти файлів, ми побачимо високу ступінь випадковості. Важливим методом виявлення стислих і зашифрованих файлів є випадковість байтів у файлі. Ця величина відома як ентропія і була визначена Клодом Е. Шенноном в його статті 1948 року. Дану величину можна розрахувати за формулою 3.1.

(3.1)

де – кількість інформації;

– кількість можливих подій;

– ймовірність окремих подій.

Максимальна ентропія виникає, коли всі байти у файлі розподіляються однаково, і коли файл не може бути стиснутий, оскільки він дійсно випадковий. Результуюче значення рівняння Шеннона зазвичай представлено між значеннями від нуля до восьми. Таким чином, значення ентропії конкретного файлу представлено цифровими значеннями від 0 до 8. Результат близький до 0 або 8, а також знаходиться між цими двома числовими значеннями. Остаточний результат буде зроблений на основі значень, отриманої з заданих наборів даних. Наприклад, якщо виміряне значення ближче до нуля, це означає, що значення даного набору даних має невипадковий або упорядкований формат. В іншому випадку, якщо значення ближче до восьми, то даний набір має випадковий чи невпорядкований формат. Це основна концепція обчислення ентропії файлу, яка пов'язана з рівнянням Шеннона.

Функція обчислення ентропії застосовується для різних цілей. Але в основному вона використовується для пошуку значень зашифрованих даних і стиснутих файлів. Зазвичай випадкові дані не схожі на звичайні призначені для користувача дані. З цією метою користувачі використовують функцію Шеннона для обчислення заданого значення даних в неоднорідному форматі.

Через незручний формат випадкових даних в порівнянні з типовими для користувача даними, виконувані файли зазвичай шифруються з використанням функції синхронізованого алгоритму дешифрування. Таким чином, користувачі можуть легко отримувати доступ щодо обсягів даних і можуть ефективно знаходити значення ентропії для цих файлів.

Файлова ентропія також використовується в області захисту від шкідливих програм, наприклад в антивірусах чи інших програмах, пов'язаних з безпекою, якими можна перевірити систему, щоб витягти всю інформацію, щоб визначити, чи є файл шкідливим або звичайним.

Отже даний метод є гарним прикладом знаходження зашифрованих файлів на основі їх характеристик.

Метод Ентропії базується на обчислюванні стиснення файлу, простіше кажучи це звичайна заміна шаблонів бітів файлу на більш короткі шаблони бітів. Отже, чим більше ентропії у файлі – тим менше його можна стиснути. Визначення ентропії допомагає виявити зашифрований файл, так як розробникам шкідливого програмного забезпечення потрібно втиснути якомога більше даних в файл та зробити його непридатним для розшифрування сторонніми програмами.

Для формування порогового значення ентропії потрібно реалізувати метод програмно, скориставшись формулами для її обчислення. Для початку потрібно відкрити файл в двійковому режимі та зчитати всі байти:

Лістинг 3.3 – реалізація відкриття файлу та зчитування всіх байтів

```
with open(FILE, "rb") as FILE:
      byteArr = FILE.read()
```

Далі потрібно обчислити частоту кожного байтового значення у файлі. Це можна зробити пробігши по масиву. Значення 256 представляє число можливих дискретних значень. Байт(b), що складається з восьми бітів, матиме 256 можливих значень.

Лістинг 3.4 – обчислення частоти кожного байтового значення у файлі

```
freqList = []
for b in range (256):
   ctr = 0
   for byte in byteArr:
       if byte == b:
           ctr += 1
   freqList.append(float(ctr) / fileSize)
```

Фінальним кроком є обчислення ентропії за формулою 3.1:

Лістинг 3.5 – Обчислювання ентропії файлу

```
ent = 0.0
for freq in freqList:
    if freq > 0:
        ent = ent + freq * math.log(freq, 2)
ent = -ent
return ent
```

Потрібно відмітити, що максимальна ентропія, на основі вищевказаного коду, буде дорівнювати значенню 8, що відповідає кількості бітів в байті.

Далі треба встановити значення ентропії при якому файл можна вважати зашифрованим. Так як ентропія – це міра невизначеності, то можна зробити висновок що ентропія у зашифрованого файлу буде більша за ентропію звичайного файлу.

Щоб дізнатися приблизне значення ентропії для зашифрованих файлів, потрібно сформулювати набір зашифрованих файлів для визначення їх значень ентропії та на основі отриманих даних вивести закономірне значення. Набір таких даних можна сформулювати на основі таблиці 3.5, відібравши декілька зашифрованих файлів та додати один незашифрований файл щоб побачити різницю в значеннях.

Таблиця 3.5 – Тестовий набір файлів для визначення порогового значення для зашифрованих файлів

Назва файлу	Розширення

index	py
LICENCE	txt.EnCiPhErEd
README	txt. EnCiPhErEd
NEWS	txt. EnCiPhErEd
Setuptools-15.2	zip. EnCiPhErEd
NEWS	txt. EnCiPhErEd
Setuptools-15.2	zip. EnCiPhErEd

Програмна для обчислення значення ентропії файлу зображена на рисунку 3.4, на основі котрого можна зробити висновки щодо порогового значення. На основі невеликої вибірки файлів випливає значення ентропії від 7.8 до максимального в 8. Також можна помітити, що ентропія звичайного файлу значно відрізняється від зашифрованого файлу і знаходиться в діапазоні від 5 до 6.

```
processing...
index.py
5.112654650600476
LICENSE.txt.EnCiPhErEd
7.9884393561391835
NEWS.txt.EnCiPhErEd
7.9936704077414245
README.txt.EnCiPhErEd
7.987163243094497
setuptools-15.2.zip.EnCiPhErEd
7.999666295896748
```

Рисунок 3.4 – Обчислення значення ентропії для кожного із файлів

Метод, який базується на математичних розрахунках можна вважати хорошим прикладом рішення задачі виявлення зашифрованих файлів, так як значення ентропії(міру випадковості) в зашифрованому файлі дуже важко занизити, що є беззаперечним плюсом.

Мінусом ж можна відмітити те, що даний метод має помилкове спрацьовування, яке проявляється при аналізі архівів. Так як zip-архівація –

це шифрування за допомогою нерівномірних кодів Хаффмана, то ентропія також буде мати високі показники. Архіватори начебто перемішують байти файлу щоб досягнути максимального їх розподілу. Щоб продемонструвати це візуально можна порівняти два файли на щільність байтів, скористувавшись програмою для відображення гістограм розподілу ймовірності повторів однакових байтів у файлі.

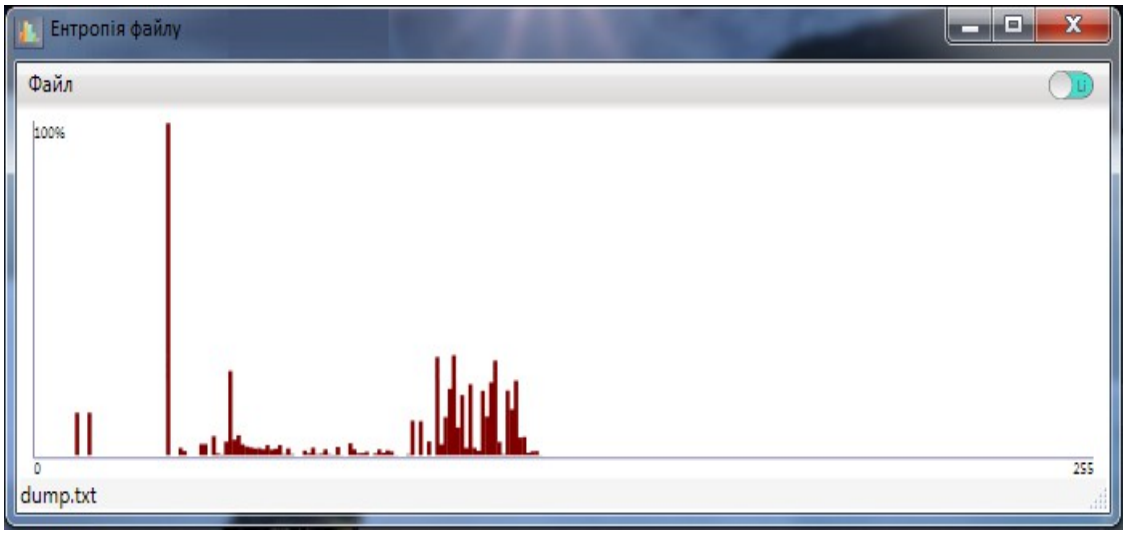

Рисунок 3.5 – Гістограма файлу з розширенням .txt

На рисунку 3.5 зображено розподіл ймовірності повторів однакових байтів у файлі формату .txt. Файл даного формату складається лише з тексту, який кодується за допомогою ASCII. Вважаючи це, можна помітити кінець гістограми на значені в 127(граничний код символу в таблиці ASCII). Пік значень під 100% зумовлений використанням якогось одного символу найчастіше (в тексті цей символ – пробіл).

На рисунку ж 3.6 відображено щільність байтів в архіві формату .7z. Можна відзначити майже однакову ймовірності повторів однакових байтів у файлі, що означає максимальну щільність файлу з відсутністю надмірності.

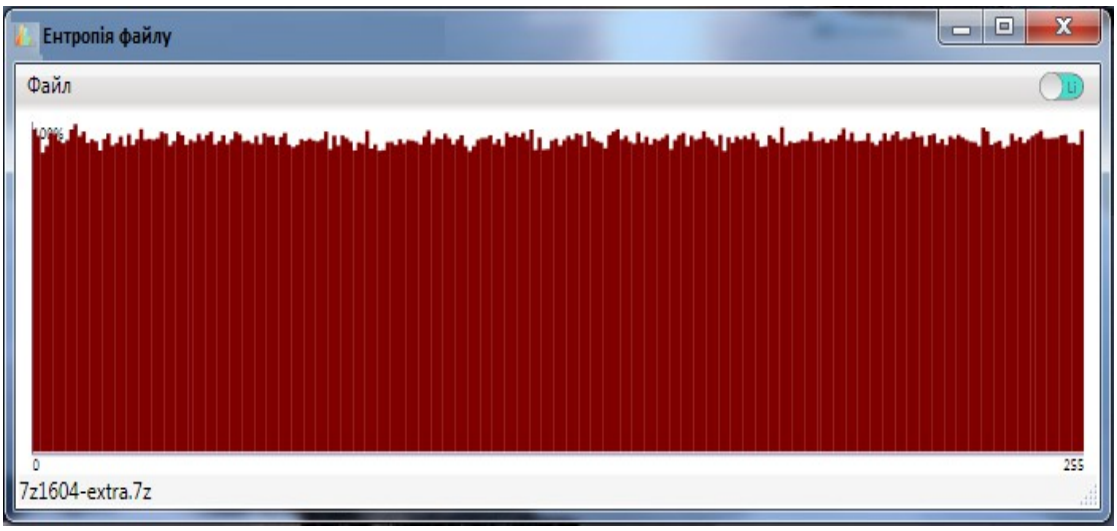

Рисунок 3.6 – Гістограма файлу з розширенням .7z

Таким чином можна зробити висновок, що значення еноропії файлу не тільки вказує на зашифрованість файлу, але також опосередковано є ознакою архіву.

3.4 Пошук зашифрованого файлу на основі трьох методів

Звертаючи увагу на позитивні та негативні сторони методів, можна зробити висновок, що як індивідуальні програми їх вже навчились обходити. Але, з'їдавши дані три методи в один спільний скрипт, тим самим перекривши низку мінусів кожного методу, можна досягнути кращих результатів.

Потрібно лише зв'язати всі методи за допомогою виводу. В лістингу 3.6 приведена реалізація «з'єднання» реалізацій в один скрипт за допомогою виводу. Спочатку потрібно зчитати шлях до папки та перевести список файлів в рядок.

Лістинг 3.6 – Реалізація «з'єднання» скриптів в один

```
def checkFolderForEncryptedFiles(path):
    listDir = os.listdir(path)
    print("processing...")
    print(listDir)
```

```
if len(listDir) != 0:
    encryptedFiles = []
    for FILE in listDir:
   if shannonEntropyCalculate(path + "/" + FILE) > 7.99:
            encryptedFiles.append(FILE)
            continue
elif checkFileTypeSignature(path + "/" + FILE) == "data":
            encryptedFiles.append(FILE)
            continue
elif checkFileEncryptExtension(path + "/" + FILE):
            encryptedFiles.append(FILE)
            continue
```

Для того щоб повністю бути впевненим в атаці шифрувальника, потрібно зробити обмеження на кількість файлів в папці, наприклад до 7. В лістингу 3.7 реалізований показ файлі в папці.

Лістинг 3.7 – Реалізація порогового значення для файлів в папці та виводу скрипту

```
if len(encryptedFiles) > 7:
            print("В данной папке обнаружено " + str(
              len(encryptedFiles)) + " зашифрованих файлів.
     Можлива   атака   шифрувальника   \Список   зашифрованих
     файлів:")
            for encryptedFile in encryptedFiles:
                print(encryptedFile)
    else:
            print("В даній папці мало значень для скану")
    else:
        print("В папці не виявлено файлів")
    wait = input("press any key to exit")
```

Додамо до даних в таблиці 3.6 ще декілька зашифрованих та не зашифрованих. В таблиці 3.6 зображені додаткові файли.

Таблиця 3.6 – Додаткові тестові файли

Назва файлу	Розширення
2020	docx.EnCiPhErEd
cartoon	.jpg.crypted
class	.docx.EnCiPhErEd
inter	.docx.EnCiPhErEd
One	.pdf. EnCiPhErEd
src	.png.deria

Two	.pdf. EnCiPhErEd
eicar	.zip. EnCiPhErEd
zara	.txt
Req	.pdf
prof	.docx
soon	.jpg

На рисунку 3.7 відображено шляху до папки де знаходяться файли та список файлі в папці.

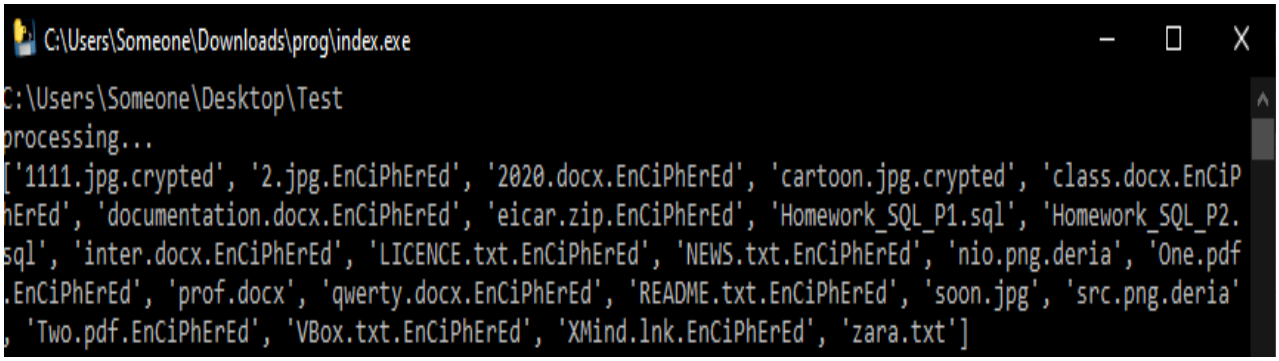

Рисунок 3.7 – Список файлів в папці

Також на рисунку 3.8 зображено список зашифрованих файлів та їх кількість в обраній папці.

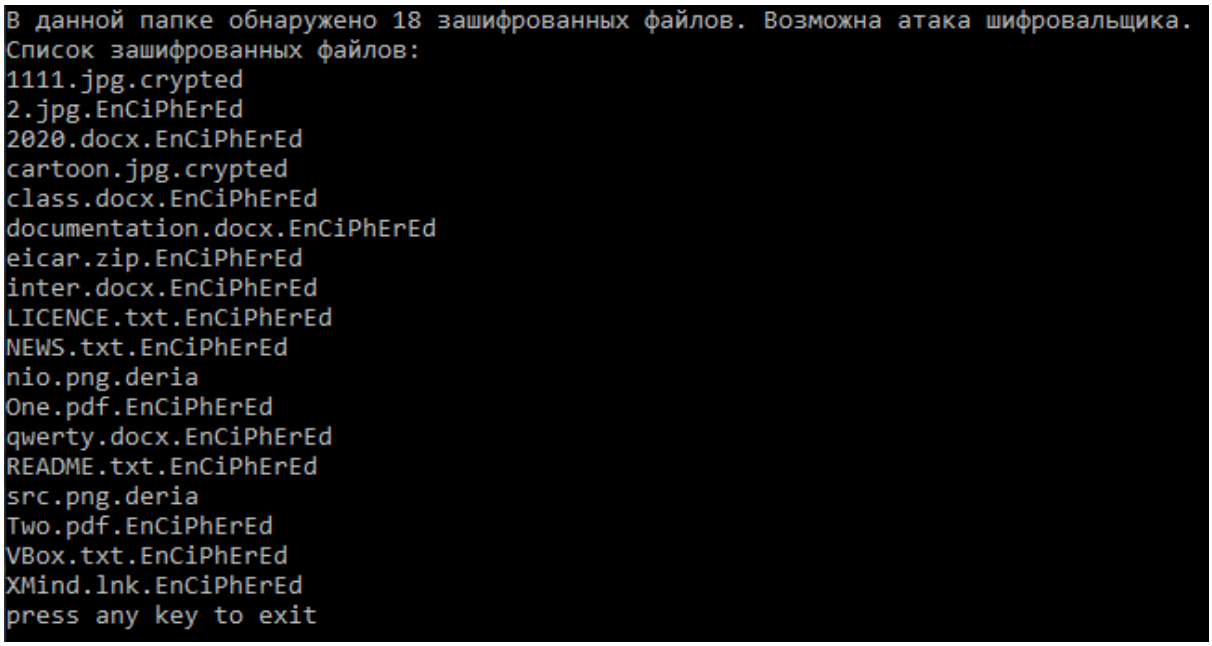

Рисунок 3.8 – Список зашифрованих файлів в обраній папці

ВИСНОВКИ

В даній атестаційній роботі було розглянуто поняття шифрування, програми-вимагачі, їх методи та техніки. Також було продемонстровано декілька програм-здирників та визначено три методи, за допомогою котрих будуть знаходитися зашифровані файли. Обрані методи були реалізовані програмно та, на основі тестових наборів, були перевірені на коректність роботи. Кожен із методів мав як слабкі, так і сильні сторони.

Метод детектування на основі вторинного розширення відзначився простотою реалізації та швидкістю роботи, але мав мінус із-за складності в підтримці, бо таблицю з небезпечними розширеннями потрібно обновляти майже кожного дня, так як нові програми-здирники з’являються с малою періодичністю та нинішні шкідливі програми обновлюють свої методи захоплення систем. Метод детектування зашифрованих файлів на основі аналізу байтів сигнатури файлу на початку файлу мав такі ж самі позитивні сторони як і у першого методу, але його також можна обійти, зберігаючи ту ж саму сигнатуру. Третій, і останній спосіб, пошуку зашифрованих файлів є метод на основі підрахунку значення ентропії. Він базується на математичному підрахунку ймовірності бітів в даних і мав гарний результат в детектуванні шкідливих зашифрованих файлів. Але в силу того, що ентропія є показником не тільки зашифрованого файлу, а й його стислості – цей метод мав хибне спрацювання на даних типу архівів (.zip, .bzip, .7z і тд.).

Для удосконалення роботи та усунення деяких мінусів методів, було прийняте рішення – об’єднати програмні реалізації в один скрипт, який знаходить файли, котрих не ураховували методи окремо один від одного. Також швидкість роботи стала меншою із-за кількості перевірок.

Як варіант покращення даної програмної реалізації можна застосувати машинне навчання, воно вдосконалить та прискорити метод детектування зашифрованих файлів на основі підрахунку значення ентропії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Security effectiveness Report. Deep dive into cyber reality, Mandiant, [Текст]: звіт / 2020
2. Holmes, G. Evolution of attacks on Cisco IOS devices [Текст] / G. Holmes. 2015 – 21 p. – ISBN 978-0-7573-05-4.
3. Omar Santos. Attackers Continue to Target Legacy Devices [Текст] / O. Santos. 2020 – 39 p. – ISBN 0-14-08-4510-5.
4. Bromiley, M. Attacking the Hospitality and Gaming Industries: Tracking an Attacker Around the World in 7 Years [Текст] / M. Bromiley, P. Lewis. 2016 – 18 p. – ISBN 978-0-415-98478-3.
5. Lucand, G. 2018. Detect DCShadow, impossible? [Текст] / G. Lucand / 2018 – 12 p. – CEW-2018-3201.
6. Anthony, N. Trickbot Shows Off New Trick: Password Grabber Module. [Текст] / N. Anthony, C. Pascual. 2018 – 57 p. – ISBN 978-978-97067-2
7. Adair, S. Virtual Private Keylogging: Cisco Web VPNs Leveraged for Access and Persistence. [Текст] / S. Adair. 2015 – 63 p. – MVE-2014-787.
8. Delpy, B. & LE TOUX, V. DCShadow. [Текст] / B. Delpy. 2011 – 14 p. – YVE-2010-3147.
9. Routin, D. Abusing network shares for efficient lateral movements and privesc (DirSharePivot). [Текст] / D. Routin. 2017 – 34 p. – FEAW-2015-245-02-368.
10. Mudge, R. Browser Pivoting. [Текст] / R. Mudge 2013 – 9 p. – CEW-2013-32-01
11. Annamalai, N. What is Azure Virtual Network?. [Текст] / N. Annamalai, C. Casey, M. Almeida 2019 – 13 p. – CVE-2019-3117.

12. Microsoft. Attractive Accounts for Credential Theft. [Текст] / 2016 – 7 p. – CVI-2016-378.

13. O'Donnell, L. Facebook: A Top Launching Pad For Phishing Attacks. [Текст] / L. O'Donnell. 2020 – 55 p. – CVE-2020-378.

14. Marczak, B. The Million Dollar Dissident: NSO Group's iPhone Zero-Days used against a UAE Human Rights Defender. [Текст] / B. Marczak, J. Scott-Railton. 2016 – 36 p. – DSF-154-22-54-1-0.

15. Falcone, R. UPS: Prior Zero-Days and the Pirpi Payload. [Текст] / R. Falcone, R. Wartell, 2015 – 37 p. – CVE-2015-3113.

16. Hartrell, G. Get a handle on cd00r: The invisible backdoor. [Текст] / G. Hartrell. 2002 – 22 p. – SDFE-14-1245-71-0.

17. Holmes, G. Evolution of attacks on Cisco IOS devices. [Текст] / G. Holmes 2015 – 63 p. – SAI-145-2587-55-01.

18. Shinotsuka, H. How Attackers Steal Private Keys from Digital Certificates. [Текст] / H. Shinotsuka. 2013 – 10 p. – QDE-147-154-2-01.

19. Adair, S. PowerDuke: Widespread Post-Election Spear Phishing Campaigns Targeting Think Tanks and NGOs. [Текст] / S. Adair. 2016 – 21 p. – CRAD-158-14-225-01.

20. Pierre-Marc Bureau. Linux/Cdorked.A: New Apache backdoor being used in the wild to serve Blackhole. [Текст] / P. Bureau. 2013 – 14 p. – DGRE-154-14-012-4

21. Tedesco, B. Security Alert Summary. [Текст] / B. Tedesco. 2014 – 2 p. – YDR-1547-59-12-1.

22. Creus, D. Sofacy's 'Komplex' OS X Trojan. [Текст] / D. Creus, T. Halfpop, R. Falcone. 2016 – 88 p. – CVR-2016-7413-06-5.

23. Serper, A. Cybereason Lab Analysis OSX.Pirrit. [Текст] / A. Serper. 2016 – 17 p. – ISBN 978-0-262-01966-8.

24. netbiosX. Token Manipulation. [Текст] / 2017 – 96 p. – DUDE-2017-5478.

25. McWhirt, M. Breaking the Rules: A Tough Outlook for Home Page

Attacks. [Текст] / M. McWhirt, N. Carr, D. Bienstock. 2019 – 20 p. – CVE-2017-11774.

26. Metcalf, S. Kerberos, Active Directory’s Secret Decoder Ring. [Текст] / S. Metcalf. 2014 – 47 p. – KDF-2014-7347-24.

27. Mundo, A. Ransomware Maze. [Текст] / A. Mundo. 2020 – 33 p. – ISTDG-2019-1245-1.

28. Szappanos, G. Netwalker ransomware tools give insight into threat actor. [Текст] / G. Szappanos, A. Brandt. 2020 – 40 p. – CER-2019-73822-4.

29. Chiu, A. New Ransomware Variant "Nyetya" Compromises Systems Worldwide [Текст] / A. Chiu 2016 – 11 p. – SRER-2016-88-56-45-2.

30. SophosLabs. Ragnar Locker ransomware deploys virtual machine to dodge security. [Текст] / 2020 – 31 p. – ISTB-2019-247-7.

31. Xiao, C. Xbash Combines Botnet, Ransomware, Coinmining in Worm that Targets Linux and Windows. [Текст] / C. Xiao 2018 – 56 p. – EDG-2018-1845-67-2.

32. Mamedov, O. Sodin ransomware exploits Windows vulnerability and processor architecture. [Текст] / O. Mamedov 2019 – 19 p. – ISBN 978-0-7573-17.