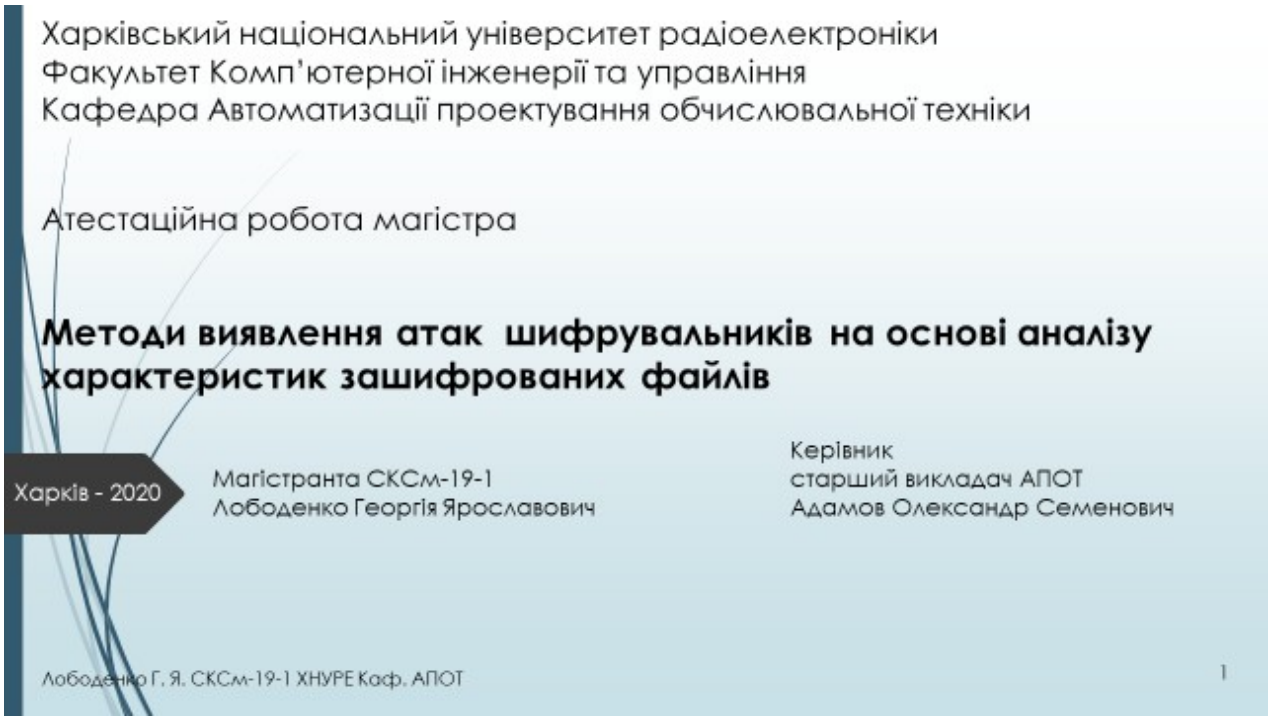
ДОДАТОК А

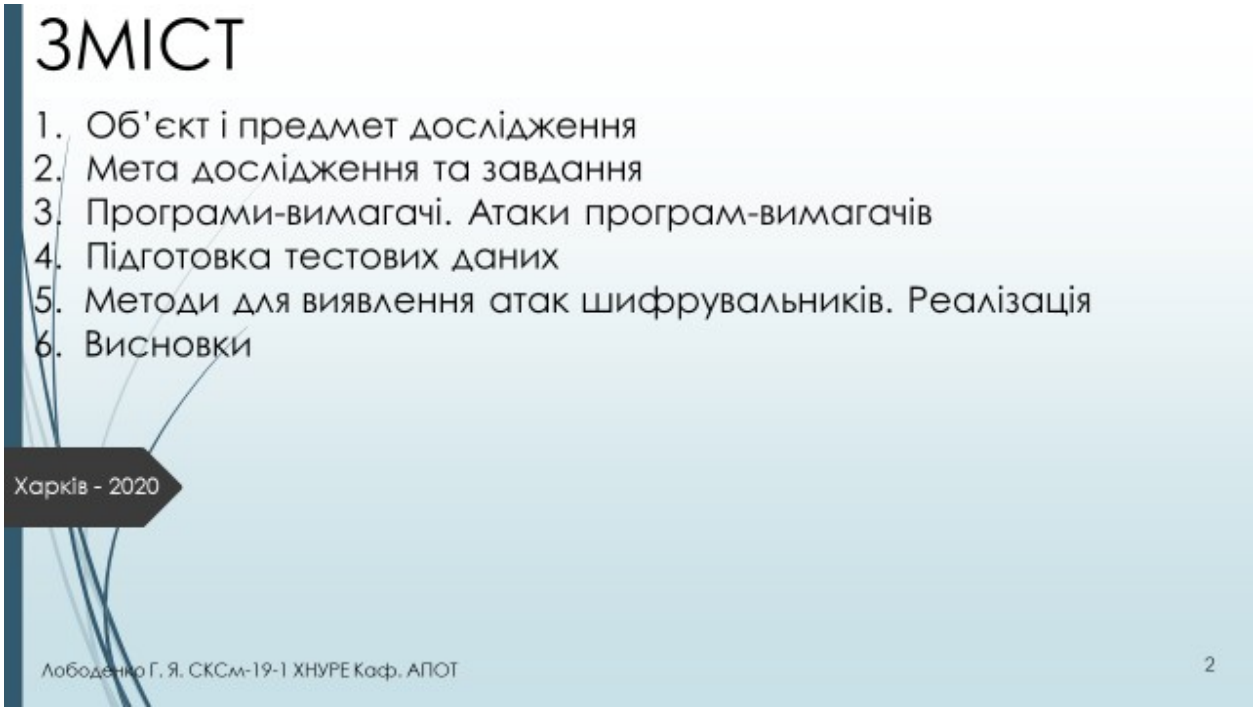
Лістинг програми

```
import math
import os
from magic import magic
def shannonEntropyCalculate(FILE):
    with open(FILE, "rb") as FILE:
        byteArr = FILE.read()
    fileSize = len(byteArr)
    freqList = []
    for b in range(256):
        ctr = 0
        for byte in byteArr:
            if byte == b:
                ctr += 1
        freqList.append(float(ctr) / fileSize)
    ent = 0.0
    for freq in freqList:
        if freq > 0:
            ent = ent + freq * math.log(freq, 2)
    ent = -ent
    return ent
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
def checkFileTypeSignature(FILE):
    with open(FILE, "rb") as FILE:
        byteArr = FILE.read(2048)
    return magic.from_buffer(byteArr)
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
        if len(encryptedFiles) > 7:
            print("В цій папці знайдено" + str(
         len(encryptedFiles)) + " зашифрованих файлів. Можлива
атака шифрувальника. \nСписок зашифрованных файлов:")
            for encryptedFile in encryptedFiles:
                print(encryptedFile)
        else:
                print("В цій папці файлів менше чим порогове
значення")
    else:
        print("В папці не знайдено файлів")
    wait = input("press any key to exit")
if __name__ == "__main__":
    path = input("Введіть шлях до папки\n")
    checkFolderForEncryptedFiles(path)
```

ДОДАТОК Б
Графічний матеріал атестаційної роботи





Об'єкт і предмет дослідження

Об'єкт дослідження: зашифровані файли після атаки програми-вимагача

Предмет дослідження: аналіз характеристик зашифрованих файлів для виявлення атак шифрувальників

Харків - 2020

Мета дослідження та завдання

Мета дослідження: аналіз та програмна реалізація методів для виявлення атаки шифрувальника (прогрми-вимагача)

Для досягнення даної мети необхідно вирішити завдання:

- розібрати поняття шифрування;
- описати методи, якими користуються розробники шкідливих програм-шифрувальників;
- розглянути техніки та методи, котрими користуються програми-здирники;
- привести приклади програм-вимагачів;
- виділити методи для детектування зашифрованих файлів;
- програмно реалізувати кожний метод окремо один від одного;
- проаналізувати кожний метод і на основі цих даних об'єднати їх таким чином щоб досягнути найліпшого результату.

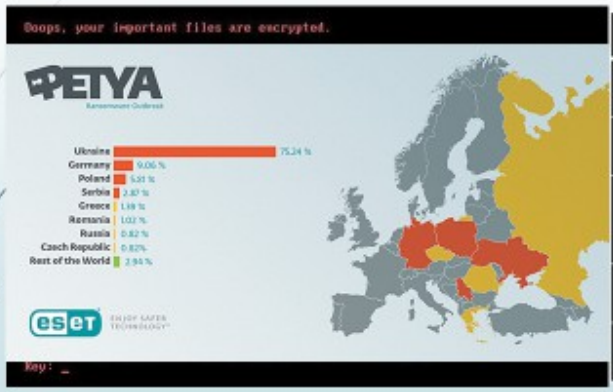
Харків - 2020

Програми-вимагачі

Харків - 2020

Атаки програм-вимагачів

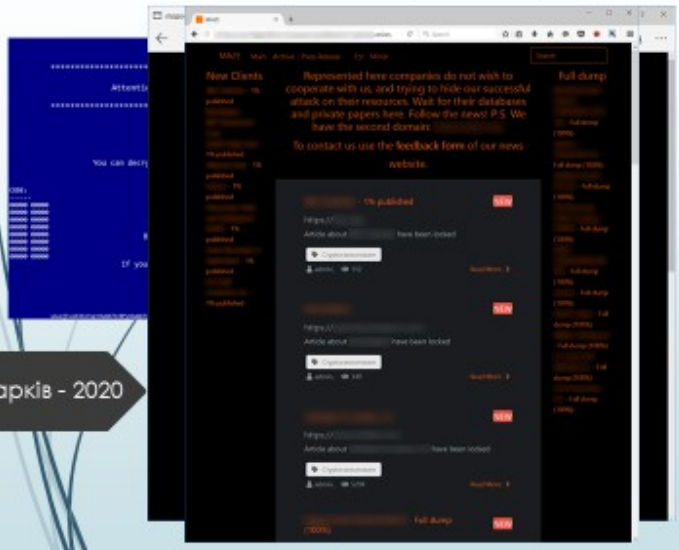
NotPetya



Метод/техніка
Шифрування даних для впливу
Використання віддалених сервісів
Інструментарій управління Windows
Запланована заздалегідь задача
Стирання журналу Windows

Харків - 2020

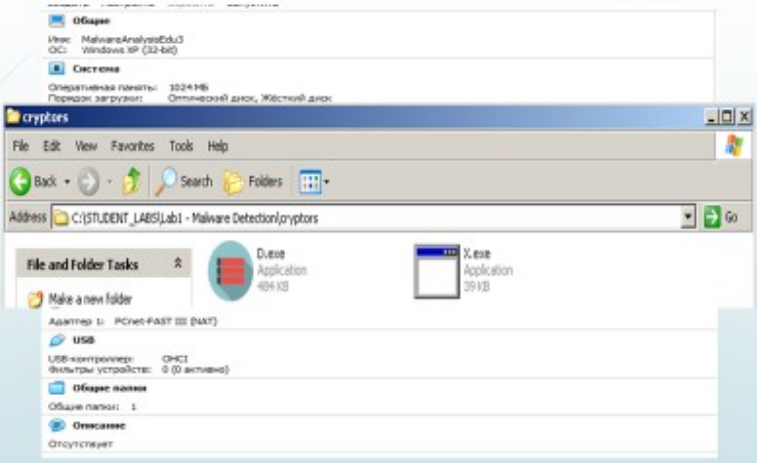
Атаки програм-вимагачів



Метод/техніка
Динамічне розширення
Шифрування даних для впливу
Власний API
Інструментарій управління Windows
Заборона відновлення системи

Харків - 2020

Підготовка тестових даних



Харків - 2020

Підготовка тестових даних

Харків - 2020

Методи виявлення атак

Метод вторинного розширення

```
В данной папке обнаружено 9 зашифрованных файлов. Возможна атак
а шифровальщика.
Список зашифрованных файлов:
1111.jpg.crypted
2111.jpg.EnCiPhErEd
documentation.docx.EnCiPhErEd
LICENCE.txt.EnCiPhErEd
NEWS.txt.EnCiPhErEd
nio.png.deria
qwerty.docx.EnCiPhErEd
README.txt.EnCiPhErEd
VBox.txt.EnCiPhErEd
press any key to exit_
```

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

Плюси:
+ швидкість
+ простота реалізації
+ точність

Мінуси:
- важко підтримувати актуальність

Харків - 2020

Методи виявлення атак

Метод перевірки magic-байтів

Плюси:
+ відсутність варіанту обходу
+ точність

Мінуси:
- в теорії є варіанти обходу, шляхом внесення змін до методу шифрування

Харків - 2020

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

.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

Методи виявлення атак

Метод обчислення значення ентропії

$$I = -\sum_{i=1}^{N} p(x_i)\log_2 p(x_i) = \sum_{i=1}^{N} p(x_i)\log_2 \frac{1}{p(x_i)}$$

де I – кількість інформації

N – кількість можливих подій

$p(x_i)$ – ймовірність окремих подій

Харків - 2020

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

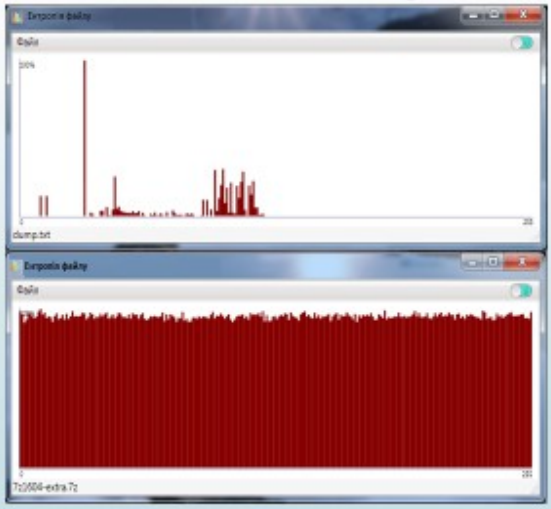
```
processing...
index.py
5.112654658680476
LICENSE.txt.EnCiPhErEd
7.9884393561391835
NEWS.txt.EnCiPhErEd
7.9936784877414245
README.txt.EnCiPhErEd
7.987163243894497
setuptools-15.2.zip.EnCiPhErEd
7.999666295896748
```

Методи виявлення атак

Метод обчислення значення ентропії

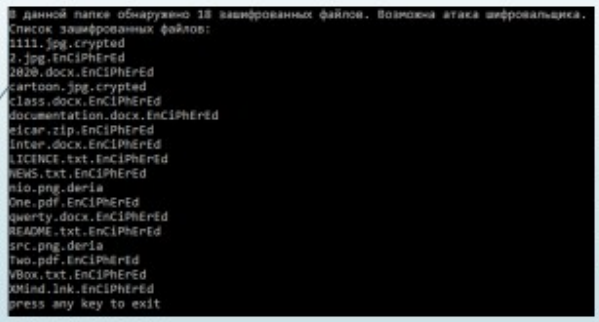
Плюси:
+ відсутність варіанту обходу
+ точність

Мінуси:
- хибне спрацювання на файлах типу архів



Харків - 2020

Результати



Назва файлу	Розширення
2020	docx.EnCiPhErEd
carloon	.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

Харків - 2020

Висновки

В результаті виконання атестаційної роботи було виконано:

- розібрати поняття шифрування;
- описати методи, якими користуються розробники шкідливих програм-шифрувальників;
- розглянути техніки та методи, котрими користуються програми-здирники;
- привести приклади програм-вимагачів;
- виділити методи для детектування зашифрованих файлів;
- програмно реалізувати кожний метод окремо один від одного;
- проаналізувати кожний метод і на основі цих даних об'єднати їх таким чином щоб досягнути найліпшого результату.

Харків - 2020

Наукова новизна. Запропоновано метод декілька методів для виявлення зашифрованих файлів та атак шифлувальників. Об'єднавши методи в одну програмну реалізацію вдалося досягти максимального результату.
Практична значимість та перспективи досліджень визначаються можливістю впроваждення розроблених методів в системи детектування з реалізацією за допомогою машинного навчання.

ДОДАТОК В

Тези доповіді, сертифікат

УДК 004.49 Технічні науки

МЕТОДИ ВИЯВЛЕННЯ АТАК ШИФРУВАЛЬНИКІВ НА ОСНОВІ АНАЛІЗУ ХАРАКТЕРИСТИК ЗАШИФРОВАНИХ ФАЙЛІВ

Лободенко Г.Я.,

студент факультету комп'ютерної інженерії

Харківський національний університет радіоелектроніки

м. Харків, Україна

Вступ. 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, с. 2].

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.

Мета дослідження – реалізація програми, яка допоможе попередити про напад програм-вимагачів на обчислювальній машині користувача.

Задача – аналіз та програмна реалізація методів, за допомогою котрих можна, виявити зашифровані файли та зробити висновок щодо атаки програми-вимагача.

Зміст дослідження. На даний момент антивіруси та інші системи запобігання злому використовують так звані два підходи до аналізу файлів, які поступають на обчислювальну машину[2, с. 481]:

– 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.

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.

Висновки. *Наукова новизна* визначається спроможністю системи, за допомогою програмної реалізації обраних методів, передбачувати напад програми-вимагача та захвату ним обчислювальної машини.

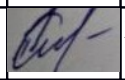
Література:

1. Security effectiveness Report. Deep dive into cyber reality, Mandiant, [Текст]: звіт / 2020

2. Eureka: A Framework for Enabling Static Malware Analysis / Sharif M. [et al.] // Recent Advances in Intrusion Detection, Lecture Notes in Computer Science. – 2008. – Vol. 5283. – Pp. 481-500.

№	Позначення	Найменування	Дод. відомості
		Текстовідокументи	
1	ГЮІК.ХХХХХХ.010 ПЗ	Пояснювальна записка	75 с.
		Графічнідокументи	
2		Слайд-презентація	15 сл.
3		Компакт-диск (CD)	1 шт.

					ГЮІК.ХХХХХ.010 ВД
Змін.	Арк.	Номер докум.	Підп.	Дата	

Розроб.	Лободенко Г.Я.		17.12.2020	Методи виявлення атак шифрувальників на основі аналізу характеристик зашифрованих файлів Відомість атестаційної роботи	Літ	Аркуш	Аркуш
Перевір.	Адамов О. С.		17.12.2020				
Н.контр.	Рожнова Т. Г.		20.12.2020		ХНУРЕ Кафедра АПОТ		
Затв.	Чумаченко С.В.						