LockBit ransomware borrows tricks to keep up with REvil and Maze

news.sophos.com/en-us/2020/04/24/lockbit-ransomware-borrows-tricks-to-keep-up-with-revil-and-maze/

Albert Zsigovits April 24, 2020

Ransomware operators are always on the lookout for a way to take their ransomware to the next level. That's particularly true of the gang behind LockBit. Following the lead of the Maze and REvil ransomware crime rings, LockBit's operators are now threatening to leak the data of their victims in order to extort payment. And the ransomware itself also includes a number of technical improvements that show LockBit's developers are climbing the ransomware learning curve—and have developed an interesting technique to circumvent Windows' User Account Control (UAC).

Because of recent dynamics in the ransomware world, we suspect that this privilege-escalation technique will pop up in other ransomware families in the future. We've seen a surge in "imposter" ransomware that are essentially rebranded variants of already-existing ransomware. Not a single day goes by where a new brand of ransomware does not come out. It has become surprisingly easy to clone ransomware and release it, with small modifications, under a different umbrella.

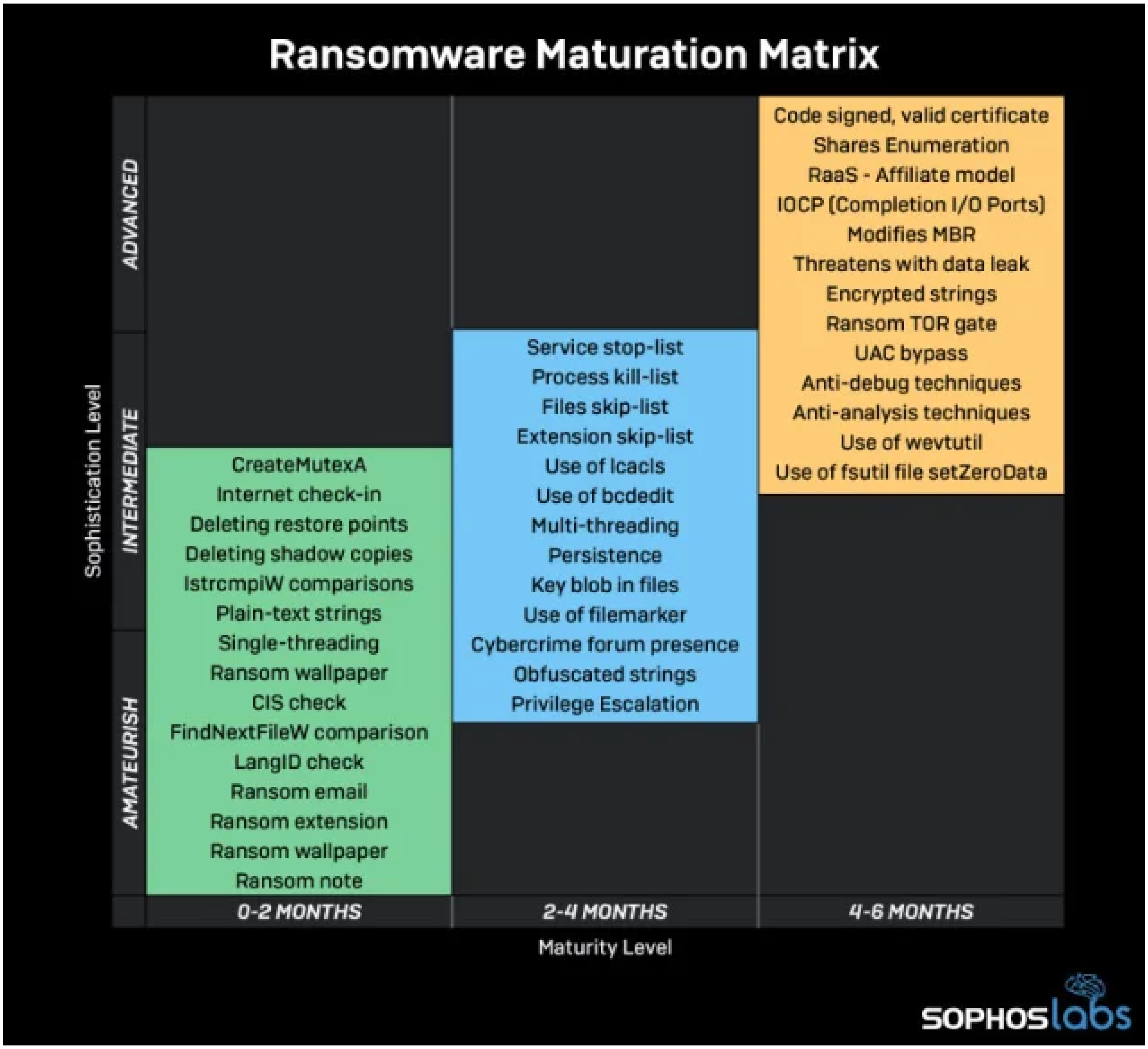
The Ransomware Learning Curve

Before we jump into the synopsis of LockBit, let’s take a moment to look at how ransomware is developed, in general. Many families follow a common timeline when it comes to the techniques and procedures ransomware developers implement at each stage. This appears to stem from the learning curve involved in creating ransomware, and the iteration of the malware as the developer builds his or her related knowledge of the malware craft.

Each ransomware seems to have an “infancy phase,” where the developer implements TTPs hastily just so the “product” can come out and start gaining its reputation. In this phase, the simplest ideas are implemented first, strings are usually plain text, the encryption is implemented in a way that only a single-thread is used, and LanguageID checks are in place to avoid encrypting computers in CIS countries. and avoid attracting unwanted attention from CIS law enforcement agencies.

After about 2 months into the ransomware operation, the developer starts implementing more sophisticated elements. They may introduce multi-threading, establish a presence in underground forums, obfuscate or encrypt strings in the binary, and there is usually a skip list/kill list for services and processes.

Around 4 months into the ransomware’s life, we start seeing things get more serious. The business model may now switch to Ransomware as a Service (RaaS), putting an Affiliate program in place. Oftentimes, binaries are cryptographically signed with valid, stolen certificates. There is a possibility that the ransomware developer starts implementing UAC bypasses at this stage. This appears to be the stage the LockBit group is entering.



Advertising the goods

As with most ransomware, LockBit maintains a forum topic on a well-known underground web board to promote their product. Ransomware operators maintain a forum presence mainly to advertise the ransomware, discuss customer inquiries and bugs, and to advertise an affiliate program through which other criminals can lease components of the ransomware code to build their own ransomware and infrastructure.

In January, LockBit's operators created a new thread in the web board's marketplace forum, announcing the "LockBit Cryptolocker Affiliate Program" and advertising the capabilities of their malware. The post claims that the new version had been in development since September of 2019, and emphasizes the performance of the encryptor and its lower use of system resources to prevent its detection.

Jan 17, 2020 Thread starter #1

Привет, друзья!

Разработка локбита ведется с сентября 2019 года, дешифровать не смогли, попытки были.

Никаких школьных емелйов, многопоточности, которая, по факту, больше грузит систему, чем шифрует, здесь нету.

A

Софт написан на си и асамблере, шифрование через IO порт завершения, порт-сканер по локальным подсетям, находит все шары DFS, SMB, WebDav, админка в торе, автоматическая тестововая дешифровка, выдача декриптора, чат с PUSH уведомлениями, Jabber-бот пересылающий переписку, завершение служб/процессов по списку и мешающих открыть файл в моменте. Установка прав на файл и снятие блокирующих атрибутов, удаление теневых копий, очистка логов, монтирование скрытых разделов, drag'n'drop файлов и папок, консольный/скрытый режим работы. Шифрует файлы кусками в разных местах, чем больше размер файла, тем больше кусков

forum post announcing LockBit's affiliate program.

LockBit's post indicates that "we do not work in the CIS," meaning that the ransomware will not target victims in Russia and other Commonwealth of Independent States countries. This comes as no surprise—as we have seen previously, CIS authorities don't bother investigating these groups unless they are operating against targets in their area of jurisdiction.

That does not mean that the LockBit group won't do business with other CIS-based gangs. In fact, they won't work with English-speaking developers without a Russian-speaking "guarantor" to vouch for them.

Escalating the extortion

In this most recent evolution of LockBit, the malware now drops a ransom note that threatens to leak data the malware has stolen from victims: *"!!! 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."*

```
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/?[redacted]
       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.
```

LockBit ransom note

If the threat were to be carried out, it might result in real-world sanctions against the ransomware victims from regulators or privacy authorities—for example, for violating the European Union's General Data Privacy Rules (GDPR) that make companies responsible for securing sensitive customer data in their possession.

An increasing number of ransomware gangs use extortion that threatens the release of private data, which might include sensitive customer information, trade secrets, or embarrassing correspondence to incentivize victims to pay the ransom, even if they have backups that prevented data loss. The data leak threat has become a signature of the REvil and Maze ransomware gangs; the Maze group has gone as far as to publicly publish chunks of data from victims who fail to pay by the deadline, taking down the dumps when they are finally paid.

Picking through LockBit's code

From a first glance at the recent LockBit sample with a reverse-engineering tool, we can tell that the program was written primarily in C++ with some additions made using Assembler. For example, a few anti-debug techniques employ the **fs:30h** function call to manually check the **PEB** (Process Environment Block) for the **BeingDebugged** flag, instead of using **IsDebuggerPresent()**.

The first thing the ransomware does at execution is to check whether the sample was executed with any parameters added from the command line. Usually, this is done to check for whether the sample is being executed in a sandbox environment. Contemporary malware often requires that the command to run the malware use specific parameters to prevent the malware from being analyzed by an automated sandbox, which often execute samples without parameters. But the LockBit sample we examined doesn't do that—it won't execute if there is *any* parameter entered from the command line. If there are no arguments in the command that executes it, Lockbit hides its console output, where the malware prints debug messages, and proceeds to do its job.

```
_start:
0040f970  55                  push    ebp {var_4}
0040f971  8bec                mov     ebp, esp
0040f973  83e4f8              and     esp, 0xfffffff8
0040f976  81ece4020000        sub     esp, 0x2e4
0040f97c  53                  push    ebx {var_2f0}  {0x0}
0040f97d  56                  push    esi {var_2f4}
0040f97e  57                  push    edi {var_2f8}
0040f97f  8d442410            lea     eax, [esp+0x10 {var_2e8}]
0040f983  c744241000000000    mov     dword [esp+0x10 {var_2e8}], 0x0
0040f98b  50                  push    eax {var_2e8} {var_2fc}
0040f98c  ff1520514100        call    dword [GetCommandLineW@IAT]
0040f992  50                  push    eax {var_300}
0040f993  ff1530524100        call    dword [CommandLineToArgvW@IAT]
0040f999  837c241002          cmp     dword [esp+0x10], 0x2
// check if arg[c] >= 2
0040f99e  8944240c            mov     dword [esp+0xc {var_2f4_1}], eax
0040f9a2  0f8daa040000        jge     0x40fe52  {0x0}
```

The command-line parameter checker in LockBit halts the ransomware if there's any parameter passed. This could be intended to detect if the sample was executed in a sandbox environment. But it's possible that either the malware author made a mistake in the implementation of the check (and wanted to check the other way around), or that this behavior is just a placeholder, and future versions will introduce different logic.

Hiding strings

LockBit's author also used several techniques to make it more difficult to reconstruct the code behind it. The Portable Executable (PE) binary shows signs of being heavily optimized, as well as some efforts by the group to cover their coding tracks—or at least get rid of some of the low-hanging fruit that reverse engineering tools look for, such as unencrypted text strings.

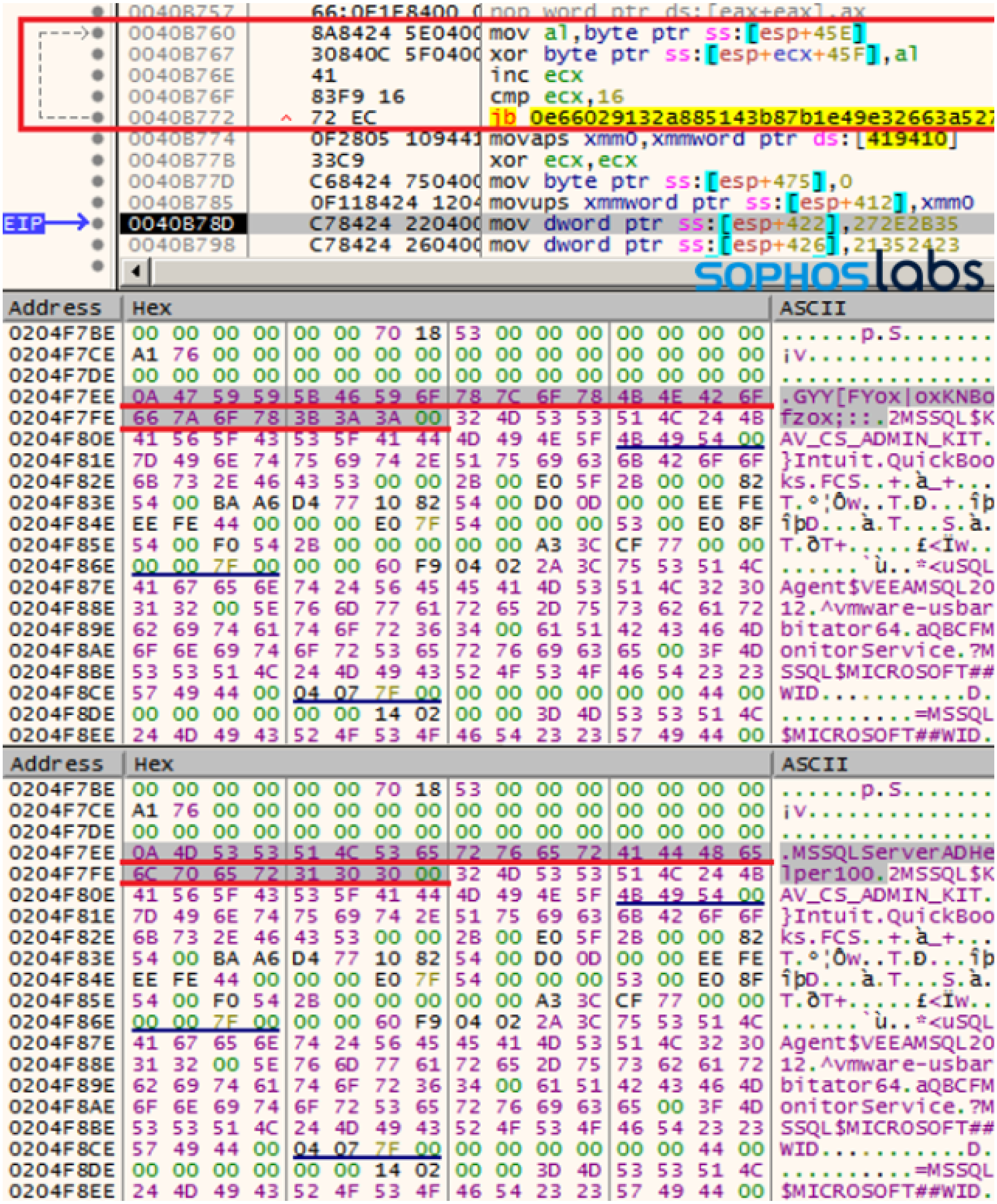
Those heavy optimizations also increase LockBit's performance. The binary makes heavy use of Intel's SSE instruction set and architecture-specific features to boost its performance. That includes the use of multiple XMM registers used to store and decrypt the service names, process names and other strings used to interact with the operating system that are unique to the ransomware.

```
.rdata:00419440 xmmword_419440  xmmword 3C372D312E5E3F2A3F3A5E2B31275E3Bh
.rdata:00419440                                  ; DATA XREF: ransom_note_gets_deobf+420↑r
.rdata:00419450 xmmword_419450  xmmword 3E29242F22392D00080A001D1F1F014Ch
.rdata:00419450                                  ; DATA XREF: deobfus_services_processes+1034↑r
.rdata:00419460 xmmword_419460  xmmword 3F271F2D242A293B210C7C7E3F271F48h
.rdata:00419460                                  ; DATA XREF: deobfus_services_processes+2FF0↑r
.rdata:00419470 xmmword_419470  xmmword 40456A01556101495E43585F497E702Ch
.rdata:00419470                                  ; DATA XREF: io_handle_ransom_note+27↑r
.rdata:00419480 xmmword_419480  xmmword 415458475A535B7C4C47504064417B35h
.rdata:00419480                                  ; DATA XREF: sub_410610+26B↑r
```

Xmmword registers **store encrypted LockBit strings**

These string variables get decrypted on the fly with a 1-byte XOR key unique to each string: the first hex byte of every variable.

Almost all the functions contain a small routine that loops around and is in charge of decrypting hidden strings. In this case, we can see that how the original **MSSQLServerADHelper100** service name gets de-obfuscated: the malware leverages a one-byte "0A" XOR key to decrypt the plaintext service name.



Deobfuscating service names in the source

Check your privilege

To ensure that it can do the most damage possible, LockBit has a procedure to check whether its process has Administrator privileges. And if it doesn't, it uses a technique that is growing in popularity among malware developers: a Windows User Account Control (UAC) bypass.

Leveraging **OpenProcessToken**, it queries the current process via a TOKEN_QUERY access mask. After that, it calls **CreateWellKnownSid** to create a user security identifier (SID) that matches the administrator group (**WinBuiltinAdministratorsSid**), so now the malware has a reference it can use for comparisons. Finally, it checks whether the current process privileges are sufficient for Administrator rights, with a call to **CheckTokenMembership**.

```
if ( OpenProcessToken(-1, TOKEN_QUERY, &TokenHandle) )// -1: CurrentProcess
{
  v9 = 68;
  if ( CreateWellKnownSid(WinBuiltinAdministratorsSid, 0, v6, &v9) )
  {
    if ( CheckTokenMembership(0, v6, &v10) )
    {
      if ( v10 )
      {
LABEL_15:
        *v1 = 1;
        goto LABEL_16;
      }
      if ( !GetTokenInformation(TokenHandle, TokenLinkedToken, &v7, 4, &v9) )
      {
        v4 = GetLastError();
        if ( v4 != 1312 && v4 != 1314 )
        {
          if ( v4 > 0 )
            v2 = (unsigned __int16)v4 | 0x80070000;
          else
            v2 = v4;
        }
        goto LABEL_16;
      }
      if ( CheckTokenMembership(v7, v6, &v10) )
```

Checking Administrator SID against the current process' SID

If the current process does not have Admin privileges, the ransomware tries to sidestep Windows UAC with a bypass. In order for that to succeed, a Windows COM object needs to auto-elevate to Admin-level access first.

To make this possible, LockBit calls a procedure called **supMasqueradeProcess** upon process initialization. Using supMasqueradeProcess allows LockBit to conceal its process' information by injecting into a process running in a trusted directory. And what better target is there for that than explorer.exe?

The source code for the masquerade procedure can be found in a Github repository.

```
v0 = NtCurrentTeb()->ProcessEnvironmentBlock;
BaseAddress = 0;
RegionSize = 4096;
if ( NtAllocateVirtualMemory((HANDLE)0xFFFFFFFF, &BaseAddress, 0, &RegionSize, 0x3000u, 4u) >= 0 )
{
  GetWindowsDirectoryW(winDir, 0x104);
  exe = 'e\0\\';                          // C:\Windows\explorer.exe
  v5 = 'p\0x';
  v6 = 'o\0l';
  v7 = 'e\0r';
  v8 = '.\0r';
  v9 = 'x\0e';
  v10 = 'e';
  lstrcpyW(BaseAddress, winDir);
  lstrcatW(BaseAddress, &exe);
}
RtlAcquirePebLock();
v17 = 0;
*(_DWORD *)SourceString = 'x\0e';        // explorer.exe
v12 = 'l\0p';
v13 = 'r\0o';
v14 = 'r\0e';
v15 = 'e\0.';
v16 = 'e\0x';
RtlInitUnicodeString(&v0->ProcessParameters->ImagePathName, (PCWSTR)BaseAddress);
RtlInitUnicodeString(&v0->ProcessParameters->CommandLine, SourceString);
RtlReleasePebLock();
return LdrEnumerateLoadedModules(0, EnumProc, 0);
}
```

sophoslabs

LockBit "masquerades" as explorer.exe

With the use of IDA Pro's COM helper tool, we see two CLSIDs—globally unique identifiers that identify COM class object—that LockBit's code references. CLSIDs, represented as 128-bit hexadecimal numbers within a pair of curly braces, are stored in the Registry path **HKEY_LOCAL_MACHINE\Software\Classes\CLSID**.

```
.rdata:00418F58 ; GUID CLSID_IColorDataProxy
.rdata:00418F58 CLSID_IColorDataProxy dd 0A16D195h        ; Data1
.rdata:00418F58                                          ; DATA XREF: uac_bypass+2A3↑o
.rdata:00418F58                 dw 6F47h                 ; Data2
.rdata:00418F58                 dw 4964h                 ; Data3
.rdata:00418F58                 db 92h, 87h, 9Fh, 4Bh, 0ABh, 6Dh, 98h, 27h; Data4
.rdata:00418F68 ; GUID CLSID_ICMLuaUtil
.rdata:00418F68 CLSID_ICMLuaUtil dd 6EDD6D74h             ; Data1
.rdata:00418F68                                          ; DATA XREF: uac_bypass+3D↑o
.rdata:00418F68                 dw 0C007h                ; Data2
.rdata:00418F68                 dw 4E75h                 ; Data3
.rdata:00418F68                 db 0B7h, 6Ah, 0E5h, 74h, 9, 95h, 0E2h, 4Ch; Data4
```

sophoslabs

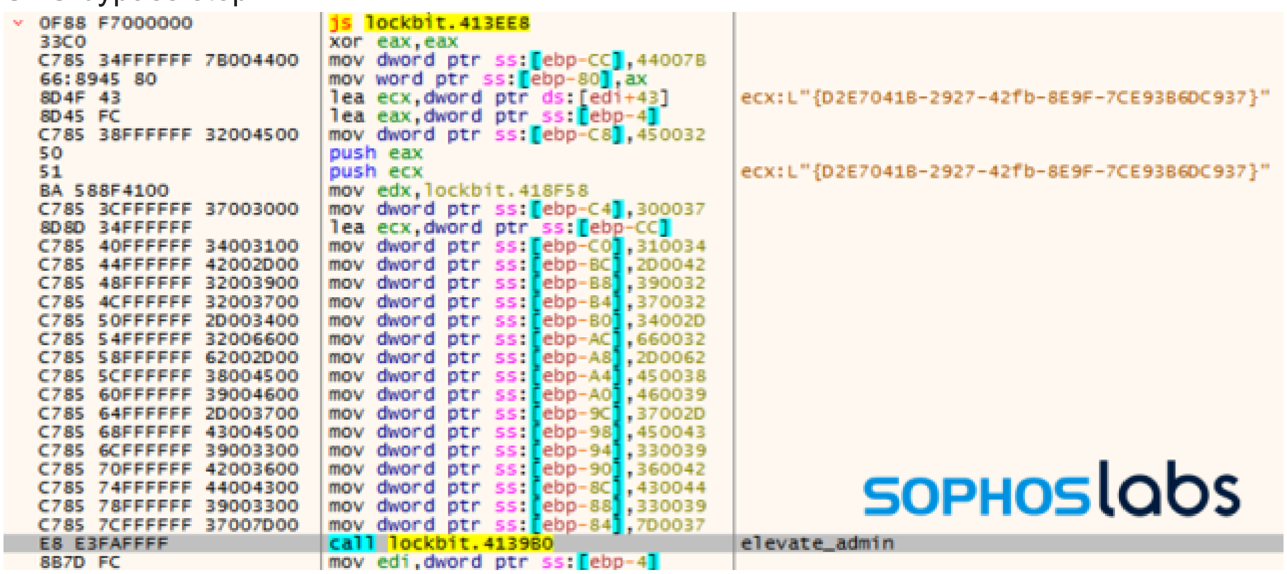
CLSIDs recognized by IDA.

Looking up these reveals that the two CSLIDS belong to **IColorDataProxy** and **ICMLuaUtil**—both undocumented COM interfaces that are prone to UAC bypass.

Name	CLSID	DLL
CMSTPLUA	{3E5FC7F9-9A51-4367-9063-A120244FBEC7}	..\system32\cmstplua.dll
Color Management	{D2E7041B-2927-42fb-8E9F-7CE93B6DC937}	..\system32\colorui.dll

Masquerading as explorer.exe, LockBit calls **CoInitializeEx** to initialize the COM library, with **COINIT_MULTITHREADED** and **COINIT_DISABLE_OLE1DDE** flags to set the concurrency model. The hex values here (CLSIDs) are then moved and aligned into the stack segment register, and the next function call (**lockbit.413980**) will further use them.

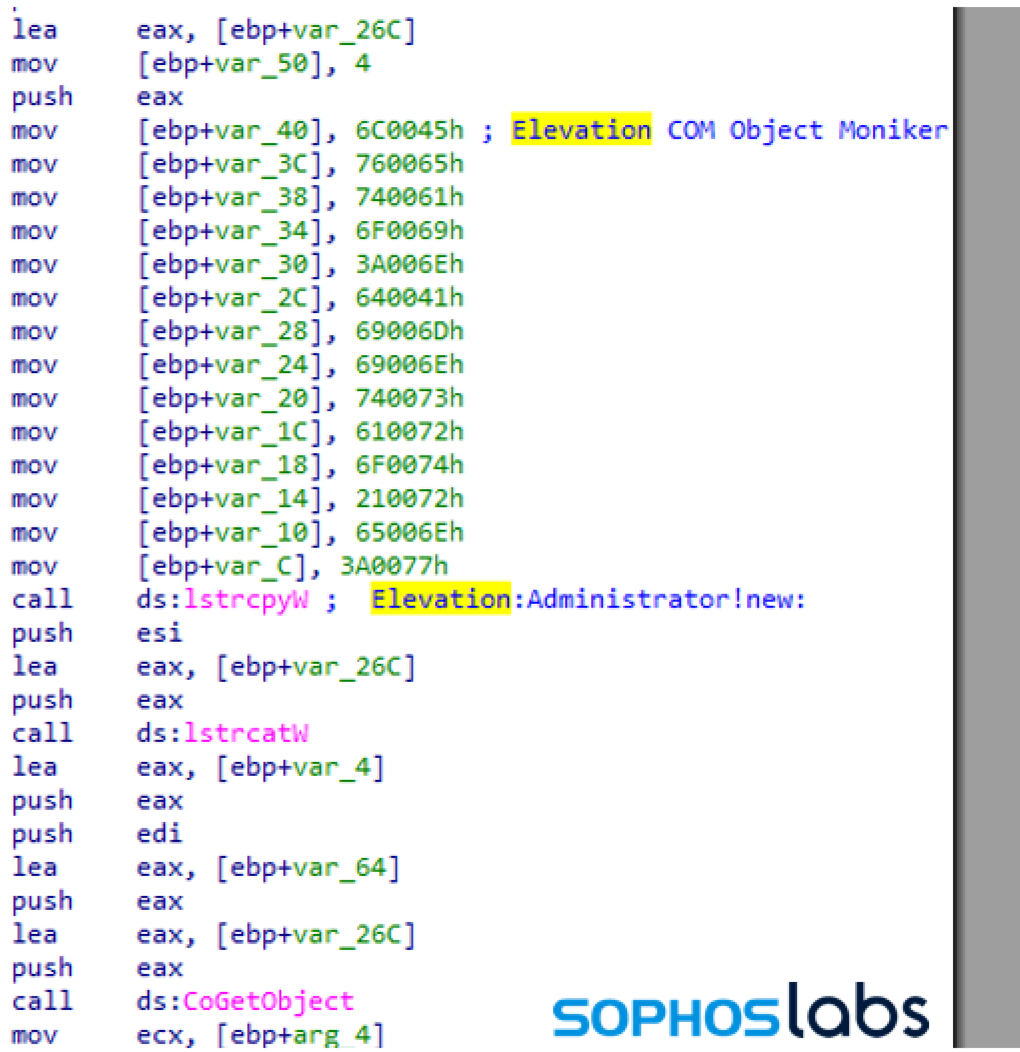
UAC bypass step 1



UAC bypass step 2

Lockbit.413980 hosts the COM elevation moniker, which allows applications that are running under user account control (UAC) to activate COM classes (via the following format: *Elevation:Administrator!new:{guid}*) with elevated privileges.

The malware adds the 2 previously seen CLSIDs to the moniker and executes them.



The COM Elevation Moniker in use.

Now, the privilege has been successfully elevated with the UAC bypass and the control flow is passed back to the ransomware. We also notice two events and a registry key change during the execution:

C:\WINDOWS\SysWOW64\DllHost.exe /Processid:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}

C:\WINDOWS\SysWOW64\DllHost.exe /Processid:{D2E7041B-2927-42fb-8E9F-7CE93B6DC937}

Key: Software\Microsoft\Windows NT\CurrentVersion\ICM\Calibration

Value: **DisplayCalibrator**

Kill or skip

LockBit enumerates the currently running processes and started services via the API calls **CreateToolhelp32Snapshot**, **Process32First**, **Process32Next** and finally **OpenProcess**, and compares the names against an internal service and process list. If one process matches with one on the list, LockBit will attempt to terminate it via **TerminateProcess**.

The procedure to kill a service is a bit different. The malware will first connect to the Service Control Manager via **OpenSCManagerA**. It then attempts to check whether a service from the list exists via **OpenServiceA**. If the targeted service is present, it then tries to determine its state by calling to **QueryServiceStatusEx**. Based on the status returned, it will call **ControlService** with the parameter **SERVICE_CONTROL_STOP (0x00000001)** on the specific service to stop it. But before that, another function (0x40F310) will cycle through all dependent services in conjunction with the target service, so dependencies are stopped too. The malware will initiate calls to **EnumDependentServicesA** to achieve this.

Hardcoded service names being checked against running services

The services that the malware tries to stop include anti-virus software (to avoid detection) and backup solution services. (Sophos is not affected by this attempt.) Other services are stopped because they might lock files on the disk, and might make it more difficult for the ransomware to easily acquire handles to files—stopping them improves LockBit's effectiveness.

Some of the services of note that the ransomware attempts to stop, in the order they are coded into the ransomware, are:

DefWatch	Symantec Defwatch
ccEvtMgr	Norton AntiVirus Event Manager Service
ccSetMgr	Symantec Common Client Settings Manager Service
SavRoam	Symantec AntiVirus suite
RTVscan	Symantec AntiVirus
QBFCService	QuickBooks is an accounting software

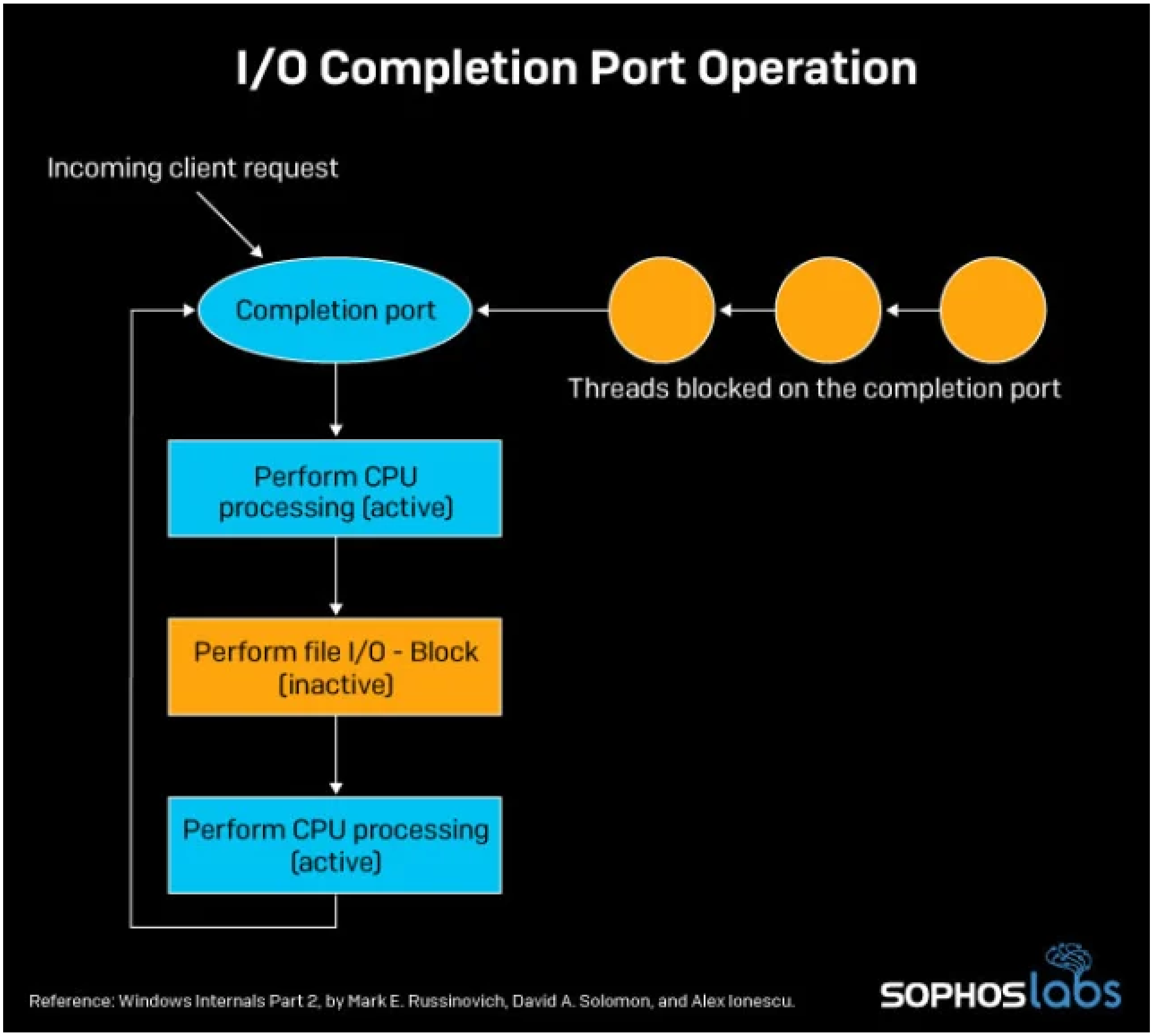
QBIDPService	QuickBooks for Windows by Intuit, Inc..
Intuit.QuickBooks.FCS	QuickBooks for Windows by Intuit, Inc..
QBCFMonitorService	QuickBooks for Windows by Intuit, Inc..
YooBackup	Wooxo Backup
YooIT	Wooxo Backup
zhudongfangyu	360 by Qihoo 360 Deep Scan
sophos	Sophos
stc_raw_agent	STC Raw Backup Agent
VSNAPVSS	StorageCraft Volume Snapshot VSS Provider
VeeamTransportSvc	Veeam Backup Transport Service
VeeamDeploymentService	Veeam Deployment Service
VeeamNFSSvc	Veeam Backup and Replication Service
veeam	Veeam
PDVFSService	Veritas Backup Exec PureDisk Filesystem
BackupExecVSSProvider	Veritas Backup Exec VSS Provider
BackupExecAgentAccelerator	Veritas Backup Exec Agent Accelerator
BackupExecAgentBrowser	Veritas Backup Exec Agent Browser
BackupExecDiveciMediaService	Veritas Backup Exec Media Service
BackupExecJobEngine	Veritas Backup Exec Job Engine
BackupExecManagementService	Veritas Backup Exec Management Service
BackupExecRPCService	Veritas Backup Exec RPC Service
AcrSch2Svc	Acronis Scheduler Service
AcronisAgent	Acronis Agent
CASAD2DWebSvc	Arcserve UDP Agent service
CAARCUpdateSvc	Arcserve UDP Update service

In addition to the list of services to kill, LockBit also carries a list of things not to encrypt, including certain folders, specific files and files with certain extensions that are important to the operating system—since disabling the operating system would make it difficult for the victim to receive and act upon the ransom note. These are stored in obfuscated lists within the code (shown below), A function within LockBit uses the **FindFirstFileExW** and **FindNextFileW** API calls to read through the file names and folder names on the targeted disk, and then a simple **lstrcmpiW** function is called to compare the hardcoded list with those names.

Accelerating file encryption

Recently, we have seen ransomware groups taking more advanced concepts and applying it to their craft. One of these advanced concepts applied in LockBit is the use of **Input/Output Completion Ports** (**IOCPs**).

IOCPs are a model for creating a queue to efficient threads to process multiple asynchronous I/O requests. They allow processes to handle many concurrent asynchronous I/O more quickly and efficiently without having to create new threads each time they get an I/O request.



That capability makes them well-suited to ransomware. The sole purpose of ransomware is to encrypt as many delicate files as possible, rendering the user's data useless. **REvil** (**Sodinokibi**) ransomware also uses IOCPs to achieve higher encryption performance.

LockBit's aim was to be much faster than any other multi-threaded locker. The group behind the ransomware claims to have used the following methods to boost the performance of their file encryption:

- *Open files with the FILE_FLAG_NO_BUFFERING flag, write by sector size*
- *Transfer work with files to Native API*
- *Use asynchronous file I/O*
- *Use I/O port completion*
- *Pass control to the kernel yourself, google KiFastSystemCall*

Once a file is marked for encryption—meaning, it did not match entries on the skip-list—a LockBit routine checks whether the file already has a **.lockbit** extension. If it does not, it encrypts the file and appends the .lockbit extension to the end of the filename.

Lockbit relies on **LoadLibraryA** and **GetProcAddress** to load **bcrypt.dll** and import the **BCryptGenRandom** function. If the malware successfully imports that DLL, it uses **BCRYPT_USE_SYSTEM_PREFERRED_RNG** which means *use the system-preferred random number generator algorithm*. If the malware was unsuccessful calling bcrypt.dll, it invokes **CryptAcquireContextW** and **CryptGenRandom** to invoke the Microsoft Base Cryptographic Provider v1.0 and generates 32 bytes of random data to use as a seed.

```
  v25 = 0x30002E;                           // Microsoft Base Cryptographic Provider v1.0
  strcpy(LibFileName, "bcrypt.dll");
  v2 = LoadLibraryA(LibFileName);
  if ( !v2 )
  {
    if ( !CryptAcquireContextW(&MS_DEF_PROV, 0, szProvider, 1u, 0xF0000000) )// CRYPT_VERIFYCONTEXT
      return 0;
    goto else;
  }
  strcpy(ProcName, "BCryptGenRandom");
  v4 = GetProcAddress(v2, ProcName);
  if ( v4 )
  {
    ((void (__stdcall *)(_DWORD, BYTE *, DWORD, int))v4)(0, pbBuffer, dwLen, 2);
    return 1;
  }
  result = CryptAcquireContextW(&MS_DEF_PROV, 0, szProvider, 1u, 0xF0000000);
  if ( result )
  {
else:
    if ( !CryptGenRandom(MS_DEF_PROV, dwLen, pbBuffer) )// 32 bytes random data at *0x18EFF8 (0x18F404)
    {
      CryptReleaseContext(MS_DEF_PROV, 0);
      return 0;
    }
    return 1;
  }
  return result;
}
```

BCryptGenRandom in use

Also, at this stage, the hardcoded ransom note, **Restore-My-Files.txt**, gets de-obfuscated and the ransomware drops the .txt file in every directory that contains at least one encrypted file.

Victim ID

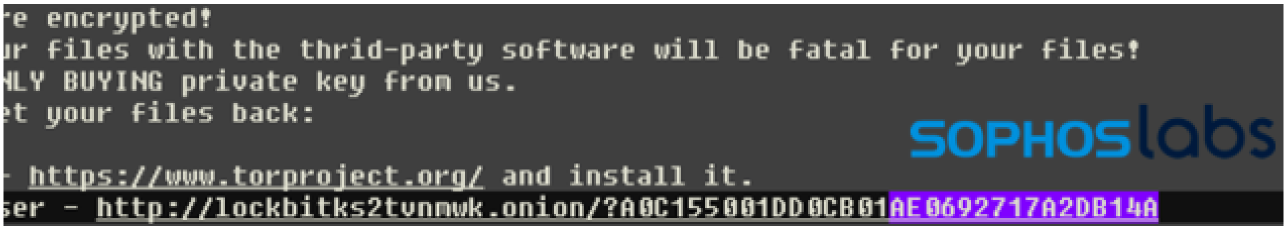
LockBit creates 2 registry keys with key blobs as values under the following registry hive: **HKEY_CURRENT_USER\Software\LockBit**

The two registry keys are:

LockBit\full
LockBit\Public

These registry keys correlate with the Victim ID, file markers, and the unique TOR URL ID that LockBit builds for each system it takes down.

Let’s take the unique TOR URL from the ransom note:



LockBit ransom note

In this example, the 16 byte long unique ID is at the end of the URL, http://lockbitks2tvnmwk[.]onion/?A0C155001DD0CB01AE0692717A2DB14A :

- The first 8 bytes used here (A0C155001DD0CB01)is the first 8 bytes of the file marker that is present in every encrypted file’s end .

  File marker at end of encrypted file.
- The last 8 bytes (AE0692717A2DB14A) is the first 8 bytes of the Public registry key.

The file marker (0x10 long) is divided into 2 sections:

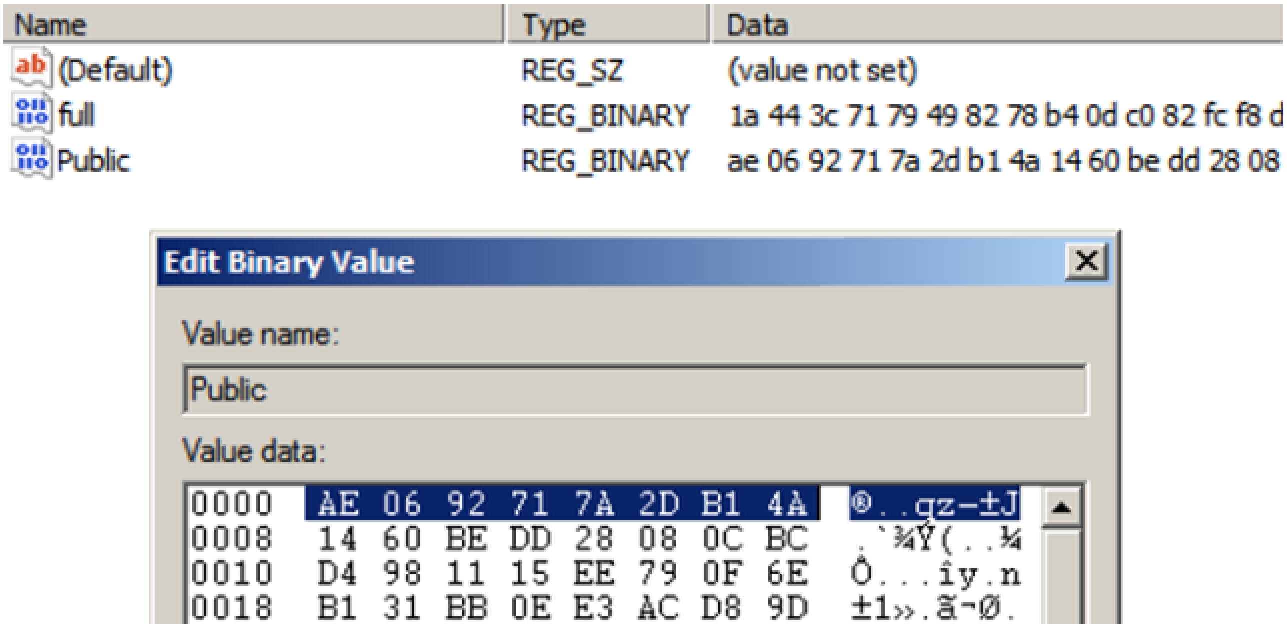
A0C155001DD0CB01

The first 8 bytes of the file marker and the first 8 bytes of the TOR unique URL ID.

D4EA7A79A0835006

The second 8 bytes are same for all encrypted files in a given run

Also, the value of the **full** registry key (0x500 long, starting as 1A443C7179498278B40DC082FCF8DE26… in this example) is also present in every encrypted file, just before the file marker.



LockBit registry keys (full and Public) that are related to the victim machine.

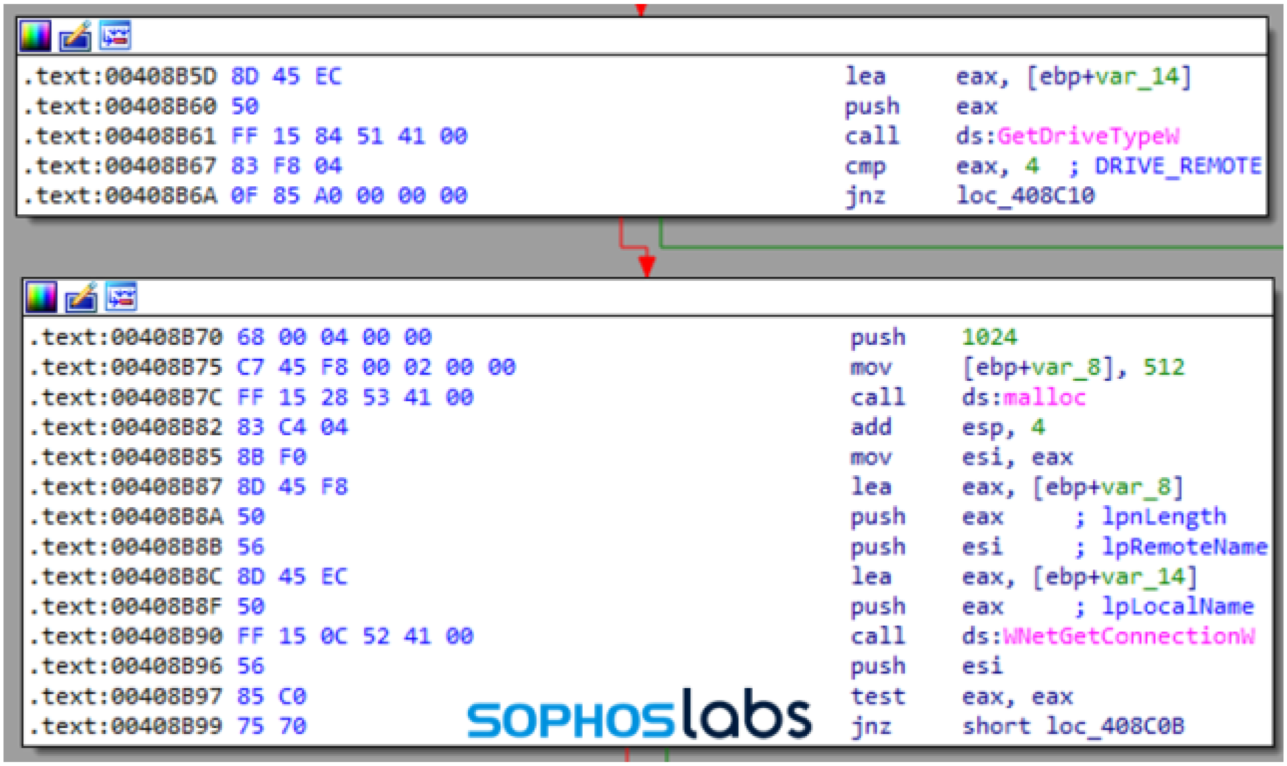
Share enumeration

For a successful ransomware hit and run, the goal is to encrypt as many files as possible. So naturally, LockBit scans for network shares and other attached drives with the help of the following API calls.

First, the malware enumerates the available drive letters with a call to **GetLogicalDrives**, then it cycles through the found drives and uses a call to **GetDriveTypeW** to determine whether the drive letters it finds are network shares by comparing the result with 0x4 (*DRIVE_REMOTE*).

Once it finds a networked drive, it calls **WNetGetConnectionW** to get the name of the share, then recursively enumerates all the folders and files on the share using the **WNetOpenEnumW**, **WNetEnumResourceW** API calls.

The ransomware can also enter network shares that might require user credentials. LockBit uses the **WNetAddConnection2W** API call with parameters *lpUserName* = 0 and *lpPassword* = 0, which (counterintuitively) transmits the username and password of the current, logged in user to connect to the given share. Then it can enumerate the share using the **NetShareEnum** API call.



Enumeration of attached, remote drives

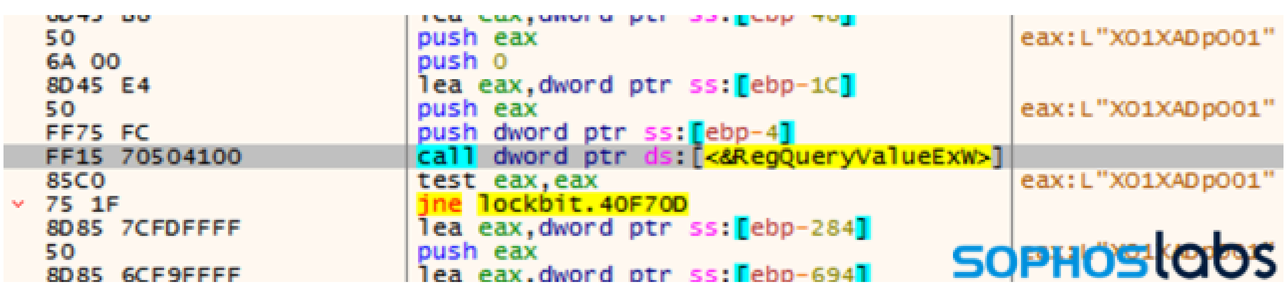
Don’t quit just yet

I an attempt to ensure that LockBit would not be kept from finishing its job by a system shutdown, the developers of this ransomware implemented a small routine that uses a call to **ShutdownBlockReasonCreate**.

The developers didn’t try to conceal the ransomware as the cause of the shutdown block: the ransomware sets the message for blocking shutdown as **LockBit Ransom.** Computer users would also see the message LockBit Ransom under the process’ name.

SetProcessShutdownParameters is also called to set the shutdown order level of the ransomware’s process to 0, the lowest level, so that the ransomware’s parent process will be active as long as it can, before a shutdown terminates the process.

If the system is shut down, the malware also has capability to persist after a reboot. LockBit creates a registry key to restart itself under HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\, called **XO1XADpO01**.



Placing a persistence Run key in registry

Stop me if you’ve heard this before

LockBit prevents multiple ransomware instances on a single system by way of a hardcoded mutex: Global**{BEF590BE-11A6-442A-A85B-656C1081E04C}.** Before LockBit starts encrypting, the ransomware checks that the mutex does not already exist by calling **OpenMutexA*,*** and calls **ExitProcess** if it does.

As soon as the ransomware is mapped into memory and the encryption process finishes, the sample will execute the following command to maintain a stealthy operation:

- **exe /C ping 1.1.1.1 -n 22 > Nul & \”%s\”** *(earlier version of LockBit)*
- **exe /C ping 127.0.0.7 -n 3 > Nul & fsutil file setZeroData offset=0 length=524288 “%s” & Del /f /q “%s”** *(recent version of LockBit)*

The ping command at the front is used because the sample can’t delete itself, due to the fact that it is locked. Once *ping* terminates, the command can delete the executable.

We clearly see an evolution to the applied technique here: in the earlier versions, the sample was missing a **Del** procedure at the end, so the ransomware would not delete itself.

In the recent version, the crooks had decided to use **fsutil** to basically zero out the initial binary to perhaps throw off forensic analysis efforts. After the file is zeroed out, the now null-file is deleted also, making double-sure the malware is not forensically recoverable.

Language matters

As we noted earlier, LockBit’s developers wanted to avoid having their ransomware hit victims in Commonwealth of Independent States (CIS) countries. The mechanism used by the ransomware to achieve this calls **GetUserDefaultLangID** and looks for specific language identifier constants in the region format setting for the current user. If the current user’s language setting matches any of the values below, the ransomware exits and does not start the encryption routine.

```
result = (unsigned __int16)GetUserDefaultLangID();
if ( (_WORD)result == 0x82C                        // Azerbaijani (az)
  || (_WORD)result == 0x42C                        // Azerbaijan, Latin (AZ)
  || (_WORD)result == 0x42B                        // Armenian (hy)
  || (_WORD)result == 0x423                        // Belarusian (be)
  || (_WORD)result == 0x437                        // Georgian (ka)
  || (_WORD)result == 0x43F                        // Kazakh (kk)
  || (_WORD)result == 0x440                        // Kyrgyz (ky)
  || (_WORD)result == 0x819                        // Russian (Moldova)
  || (_WORD)result == 0x419                        // Russian (ru)
  || (_WORD)result == 0x428                        // Tajik (tg)
  || (_WORD)result == 0x442                        // Turkmen (tk)
  || (_WORD)result == 0x843                        // Uzbek (uz)
  || (_WORD)result == 0x443                        // Uzbekistan, Latin (UZ)
  || (_WORD)result == 0x422 )                      // Ukrainian (uk)
{
  ExitProcess(0);
}
return result;
}
```

sophoslabs

If your computer's **UserDefaultLangId** is set to one of these values, LockBit does no damage

Changing the wallpaper

To get the affected user's attention, the malware (as is typical) creates and displays a ransom note wallpaper. A set of API calls are involved in this process, listed below.

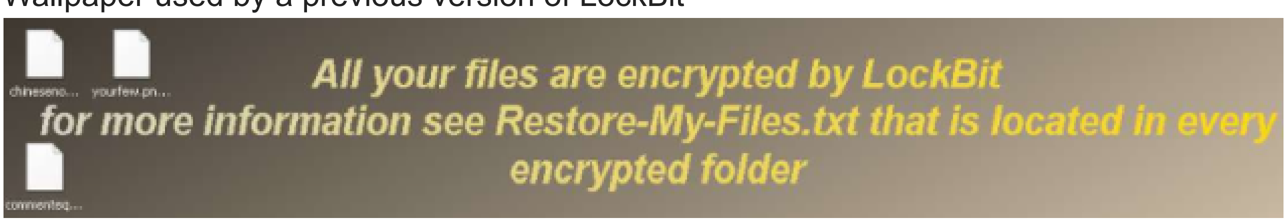
The created wallpaper gets stored under *%APPDATA%\Local\Temp\A7D8.tmp.bmp*.

In the meantime, the malware also sets a few registry keys so that the wallpaper is not tiled, and the image is stretched out to fill the screen:

HKEY_CURRENT_USER\Control Panel**Desktop**

- **TileWallpaper**=0 – (No tile)
- **WallpaperStyle**=2 – (Stretch and fill)

Wallpaper used by a previous version of LockBit

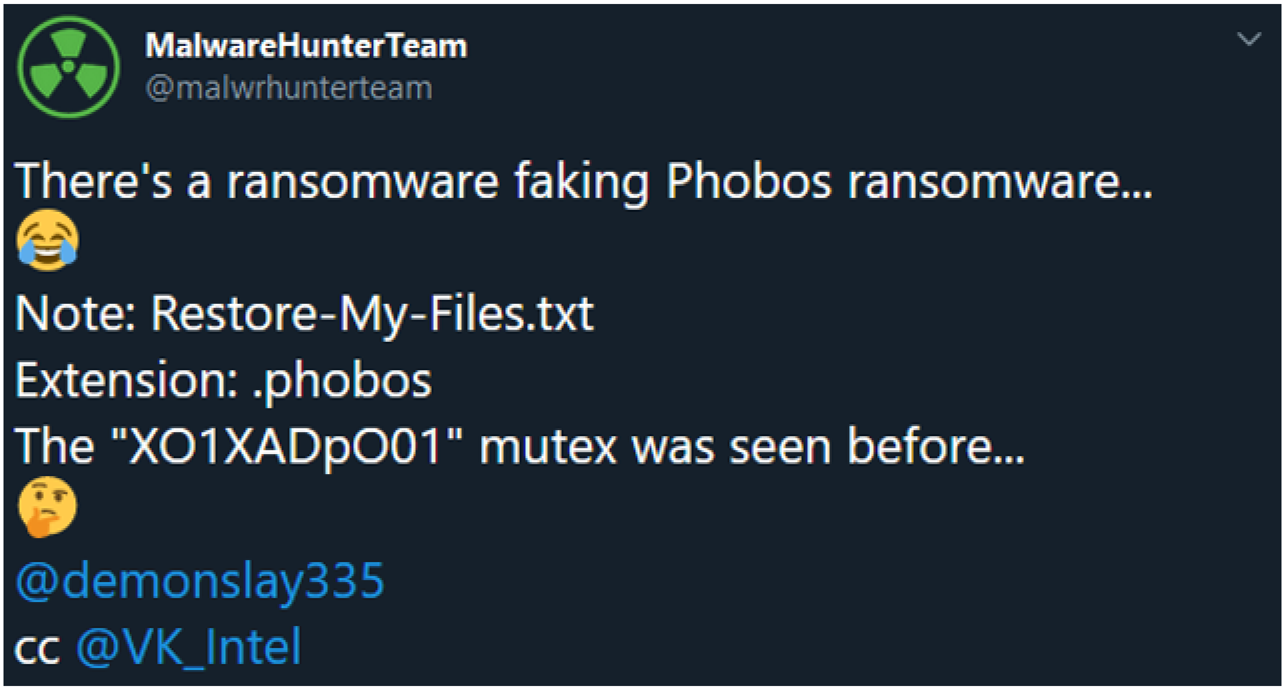


Wallpaper set by a recent version of LockBit

Stack Exchange for crooks

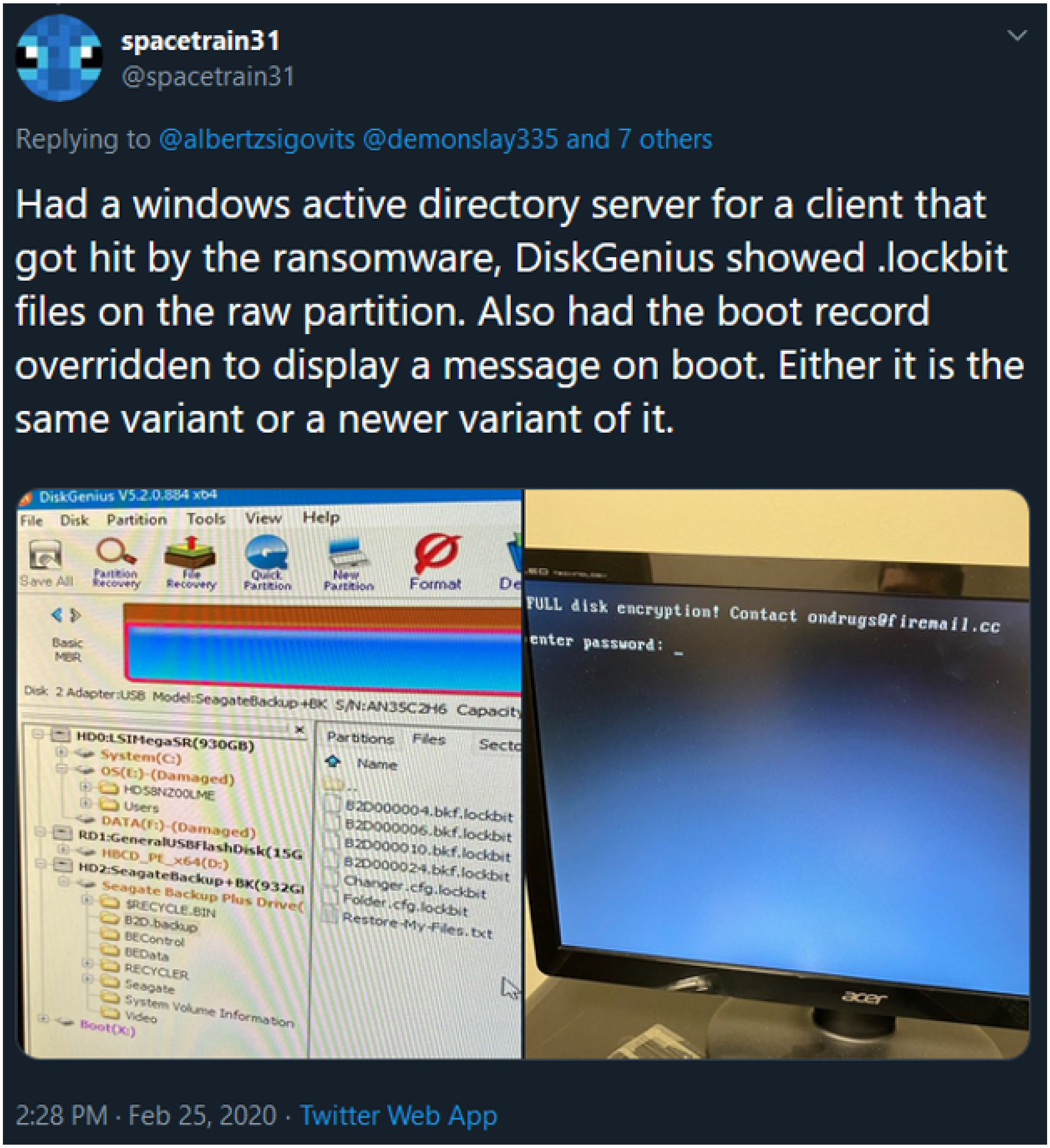
LockBit leverages a very similar service-list to MedusaLocker ransomware. It comes as no surprise that crooks copy these lists, so they don't have to reinvent the wheel.

The unique Registry run key and ransom note filename that was written by LockBit— **XO1XADpO01** and **Restore-My-Files.txt** — were also seen being used by Phobos, and by a Phobos imposter ransomware. This would suggest that there is a connection between these families, but without further evidence that is hard to justify.



The future for LockBit

A recent Twitter post demonstrates what the future looks like for LockBit. In a recent LockBit attack, the MBR was overwritten with roughly 2000 bytes; The infected machine would not boot up unless a password is supplied. The hash of this sample is currently not known.



https://twitter.com/spacetrain31/status/1232296412378955776

The e-mail used for extortion ondrugs@firemail.cc was also seen with STOP ransomware—an uncanny connection. The group behind might be related.

There is also speculation that application Diskcryptor was combined with the ransomware to add this extra lockdown layer. The MAMBA ransomware was also using this technique, leveraging Diskcryptor to lock the victim machine. DiskCryptor is currently being detected as **AppC/DCrpt-Gen** by Sophos Anti-Virus.

A list of the indicators of compromise (IoCs) for this post have been published to the SophosLabs Github.

Acknowledgments

The author would like to acknowledge the public contributions of @demonslay335 and @hfiref0x.