TECH BRIEF

Trend Research

TREND MICRO

# LockBit Attempts to Stay Afloat with a New Version

Appendix

# Technical Appendix: LockBit-NG-Dev Detailed Analysis

The new LockBit version we analyzed was packed using the MPRESS packer, possibly to evade static file detections. After unpacking, we can see that this new version seems to have been written in .NET and possibly compiled using CoreRT, which is different from the usual C/C++ language used for past versions.


Figure 1. Shows the use of MPRESS packer using the Detect-It-Easy tool

Like past versions, it still has an embedded configuration that dictates the routines it can perform. The configuration, which is in JSON format, is decrypted at runtime and includes information like date range for execution, the ransom note filename and content, unique IDs for the ransomware, the RSA public key, and some other flags and lists for its other routines. A table with the full configuration options is included at the end of this brief.

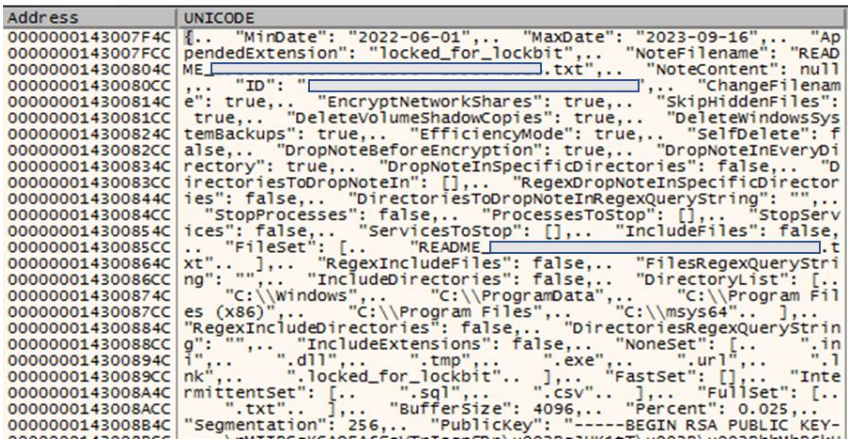


Figure 2. Decrypted configuration in JSON format

After decrypting the configuration, LockBit will then create a mutex using the value of *ID* field from the configuration as the mutex name. If the mutex already exists, the process will exit to avoid multiple instances of execution.

```
CreateMutex_14011BA80(v2, 1, _config->ID, v202);
if ( !LOBYTE(v202[0]) )                         // If Already Exist
{                                       *a4 = get_last_error != ERROR_ALREADY_EXISTS;
  if ( qword_1402771A8[-1] )
    sub_140001105();
  LODWORD(qword_1402771A8[0]) = 0;
  sub_1400E0130();
  v146 = qword_1402771A8[0];
  sub_140006400(&v145);
  ExitProcess(v146);
}
```

Figure 3. Mutex checking routine

One of the new behaviors of LockBit is its ability to check if the current date is within the date range set in the configuration. If the date is not within this range, the process will terminate. The analyzed sample only works between a specified start and end date. This is probably LockBit's way to limit affiliates from reusing their ransomware, forcing them to purchase a new version from the operators once the date expires. This can also be considered an anti-analysis and anti-sandbox technique — however, it is relatively simple for an analyst to bypass this during reverse engineering. On the other hand, it could would be more difficult for an affiliate to patch the binary before using it against a victim.

```
if ( SDWORD2(Date) < MinDate || SDWORD2(Date) > MaxDate )
{
  if ( qword_1402771A8[-1] )
    sub_140001105();
  LODWORD(qword_1402771A8[0]) = 0;
  sub_1400E0130();
  v7 = qword_1402771A8[0];
  sub_140006400(&v6);
  ExitProcess(v7);
}
```

Figure 4. Checking if the current date is within the valid date range

Similar to other ransomware, it terminates processes and stops services that may be accessing files it is attempting to encrypt or security-related processes and services that may hinder the execution of the ransomware to ensure the proper encryption of files,. To do so, it first needs to check if the *StopProcesses* or *StopServices* flags are true in the configuration. If true, it will terminate processes from the list of process names under the *ProcessesToStop* field and services from the list of services in the *ServicesToStop* field in the configuration.

```
if ( config->StopProcesses )
{
  sub_1401E44E0(v196, &qword_1402AAC48, *(config->ProcessesToStop + 8), 0i64, -1, -1, 0)

if ( v10->StopServices )
{
  sub_1401E44E0(v194, &qword_1402AAC48, *(v10->ServicesToStop + 8), 0i64, -1, -1, 0);
```

Figure 5. Routine checking if processes and services will be stopped

LockBit also inhibits recovery from shadow copies and backups by performing the following routines before encryption:

It checks if *DeleteVolumeShadowCopies* is true in the configuration, and if it is, deletes shadow copies by executing the following command:

> *"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe vssadmin Delete Shadows /All /Quite"*

To delete the Windows backups, it checks if *DeleteWindowsSystemBackups* is true in the configuration, and if it is, it executes the following command:

> *"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe wbadmin DELETE BACKUP –keepVersions:0 -quiet"*

```
if ( config_14042AC88[3]->DeleteVolumeShadowCopies )
  execute_via_powershell_140075100(&str_vssadmin_140271AA8, 0x2710u);// vssadmin Delete Shadows /All /Quiet
if ( *(&qword_140276FD8 - 1) )
  sub_140001440();
if ( config_14042AC88[3]->DeleteWindowsSystemBackups )
  execute_via_powershell_140075100(&str_wbadmin_140271BC8, 0x2710u);// wbadmin DELETE BACKUP -keepVersions:0 -quiet
```

Figure 6. Routine executing PowerShell commands to delete shadow copies and backups

One of the routines it possesses that was also in past versions is the ability to rename the encrypted files with random filenames. It does this by checking if the *ChangeFilename* field is true in the configuration, then it generates a random filename using a randomizer function.

The original file name will then be placed within the content of the file after the encrypted blob. For files that are not encrypted with full encrypt mode, it will just be appended on the file. Meanwhile, those encrypted via full encrypt mode will have filenames that will be included in the RSA encrypted buffer.

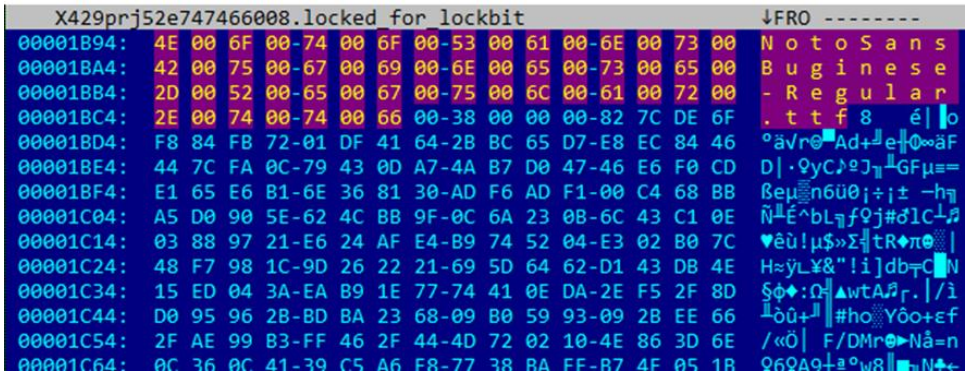


Figure 7. File encrypted with Intermittent mode has the original filename in its content

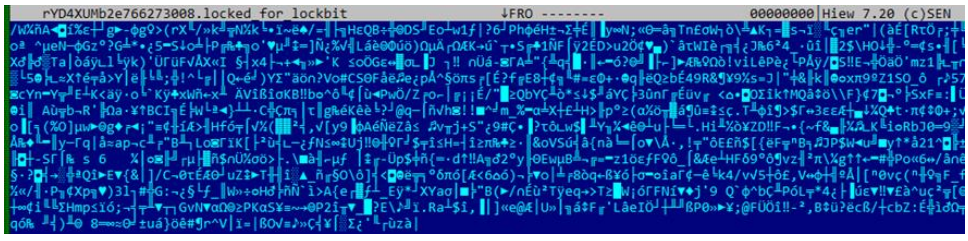


Figure 8. File encrypted with full encrypted mode has the original filename in the RSA encrypted buffer

LockBit has three encryption modes: fast, intermittent and full. Files are usually encrypted under fast mode to speed up encryption (an option commonly favored by affiliates), but it can be configured to perform different modes based on file extensions.

The sample we analyzed has set ".txt" file extensions to full encryption mode, while ".csv" and ".sql" are encrypted with intermittent mode. The three modes to encrypt files are as follows:

- **Fast** encrypts the first 0x1000 bytes of the file (files listed in Fast Set will use *Buffersize* value to determine the size to encrypt).

- **Intermittent** only encrypts a certain percentage of the file based on the value set in the configuration under the *Percent* field. Also, the field *Segmentation* determines the distance between encrypted blocks.

- **Full** encrypts the whole file.

Like other ransomware, LockBit-NG-Dev avoids encrypting certain directories, files and file extensions. These files are listed in the configuration under the *DirectoryList*, *FileSet*, and *NoneSet* fields. Also, the configuration fields *IncludeFiles*, *IncludeDirectories*, and *IncludeExtensions* need to be set to false. It also has a regex option for the files and directories to avoid under the fields *FilesRegexQueryString* and *DirectoriesRegexQueryString*.

For network encryption, if *EnableNetworkShares* is true, it also encrypts files on available network shares.

LockBit-NG-Dev encrypts files using the AES algorithm and encrypts the AES key using the embedded RSA public key that can also be found in the configuration. The AES keys are randomly generated for each file to be encrypted.

The ransom note content and file name are also in the configuration. It can also be set in the configuration if the ransom note will be dropped on all directories or only in specific directories by exact path/s that match a regular expression.

An option exists wherein the ransom note would first be dropped on target directories (or all traversed directories) before encryption begins. If this is enabled, dropping the same text file on multiple directories could be flagged by behavior monitoring tools as a suspicious routine and may terminate the execution process before the actual encryption begins.

```
if ( v28->DropNoteBeforeEncryption )
{
  v29 = sub_140077C40();
  sub_140004BF0(&v28->NoteContent, v29);
  v30 = v12[3];
  if ( v30->DropNoteInSpecificDirectories )
  {
    v31 = v30->DirectoriesToDropNoteIn;
    v32 = *(v24 + 32);
    if ( !v32 )
    {
      v32 = s_1400049E0(qword_1402ACDD0);
      sub_1400027AB(v32, *(v24 + 8));
      sub_140004BF0((v24 + 32), v32);
    }
```

Figure 9. Checking if the ransom note will be dropped before encryption

Finally, if *Self-delete* is true in the configuration, it will remove traces of the ransomware by zeroing out its contents using the following command:

*powershell.exe -Stop-Process -Id {process id} -Force; fsutil.exe file setZeroData offset=0 length=9999999999 "{Path of ransomware}"*

```
if ( config_->SelfDelete )
  config_ = sub_1400720C0(v141, v140, v142, v143);
```

Fig 10. Checking if "Self-delete" field is true

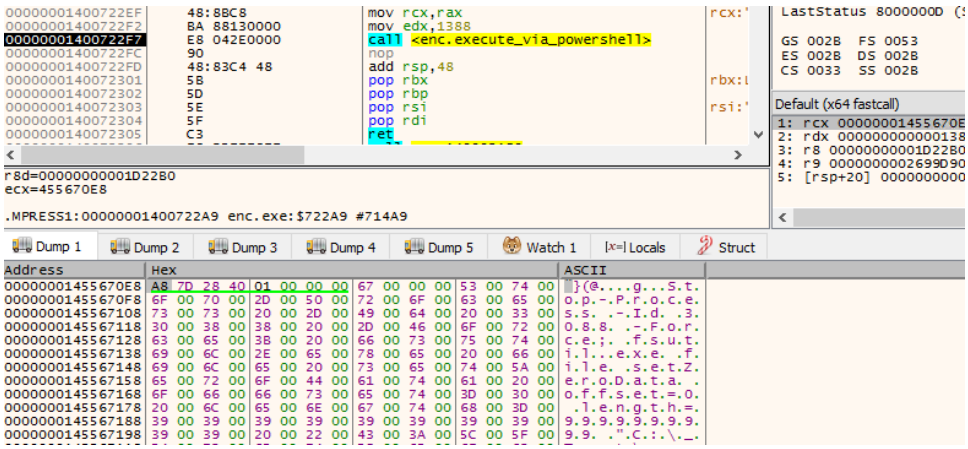


Figure 11. Executing commands via PowerShell

# Full configuration

| Field | Description |
|---|---|
| MinDate | Minimum Date where ransomware will execute (format. MM/DD/YYYY) |
| MaxDate | Maximum Date where ransomware will execute (format. MM/DD/YYYY) |
| AppendedExtension | Extension that will be appended on encrypted files (ex. locked_for_LockBit) |
| NoteFilename | Filename of the ransom note |
| ID | Unique identifier for the ransomware |
| ChangeFilename | If true, change filename of encrypted files to a random one. |
| EncryptNetworkShares | if true, include network drives in encryption |
| SkipHiddenFiles | if true, will not encrypt files with attribute hidden |
| DeleteVolumeShadowCopies | If true, delete shadow copies by executing vssadmin command |
| DeleteWindowsSystemBackups | If true, delete windows backup by executing wbadmin command |
| EfficiencyMode | Flag to check if will use enough or more resources during encryption |
| SelfDelete | If true, overwrite contents of ransomware with null bytes |
| DropNoteBeforeEncryption | If true, ransom note will be dropped first before encrypting |
| DropNoteInEveryDirectory | if true, drop ransom note on every directory |
| DropNoteInSpecificDirectories | if true, drop note only on the specified directory in field *DirectoriesToDropNoteIn* |
| DirectoriesToDropNoteIn | List of file path to drop ransom note in |
| RegexDropNoteInSpecificDirectories | if true, drop note on directory if it matches the regex expressions under the *DirectoriesToDropNoteInRegexQueryString* |
| DirectoriesToDropNoteInRegexQueryString | List of regex expression of desired directories to drop ransom note |
| StopProcesses | If true, terminate processes under the *ProcessesToStop* field. |
| ProcessesToStop | List of process names to stop |
| StopServices | If true, stop services under the *ServicesToStop* field. |
| ServicesToStop | List of service name to stop |
| IncludeFiles | If true, include the files under *FileSet* in encryption routine, otherwise avoid encrypting if set to false |
| FileSet | List of files that would be excluded in encryption if *IncludeFiles* is set to false. |
| RegexIncludeFiles | If true, include the files that matches the regular expression under *FilesRegexQueryString* in encryption routine, otherwise avoid encrypting if set to false |
| FilesRegexQueryString | Regular expression of files to include or avoid in encryption |

| | |
|---|---|
| IncludeDirectories | If true, include the directories under *DirectoryList* in encryption routine, otherwise avoid encrypting if set to false |
| DirectoryList | List of directories that would be excluded in encryption if *IncludeDirectories* is set to false |
| RegexIncludeDirectories | If true, include the directories that matches the regular expression under *DirectoriesRegexQueryString* in encryption routine, otherwise avoid encrypting if set to false |
| DirectoriesRegexQueryString | Regular expression of directories to include or avoid in encryption |
| IncludeExtensions | If true, include the files with extension under *NoneSet* in encryption routine, otherwise avoid encrypting if set to false |
| NoneSet | List of file extension to avoid during encryption, if *IncludeExtensions* is False |
| FastSet | List of extension of files that are targeted to only be encrypted with the first x bytes where x is value set in *BufferSize* |
| IntermitttentSet | List of extension of files that would only be partially encrypted depending on the value of *Percent* |
| FullSet | List of extension of files that would be fully encrypted regardless of size |
| BufferSize | Size to encrypt for files listed in *FastSet* (ex. 4096) |
| Percent | Percentage value of intermittent encryption to perform |
| Segmentation | Value to compute offset of blocks to encrypt under Intermittent Encryption (ex. 256) |
| PublicKey | RSA public key. |

Table 1. Full configuration settings

# Indicators of Compromise

Detected as **Ransom.Win64.LOCKBIT.YXDLS**

**SHA256**

f56cba51a4e86f3be5208dfce598d0d6a86cbbc820b214d5d5df7d327e580b82

**TLSH**

T1615533707F603835DB3BD27B546D0D8892FB39789A198BFAC0661F87185691F0907A8F


Trend Micro, a global cybersecurity leader, helps make the world safe for exchanging digital information. Fueled by decades of security expertise, global threat research, and continuous innovation, our unified cybersecurity platform protects hundreds of thousands of organizations and millions of individuals across clouds, networks, devices, and endpoints.

With 7,000 employees across 56 countries, and the world's most advanced global threat research and intelligence, Trend Micro enables organizations to simplify and secure their connected world.

TrendMicro.com