Back July 05, 2021 — Acronis

CYBERSECURITY

Threat analysis: LockBit ransomware

Share

Summary

- First discovered in September 2019
- Considered to be a descendant of LockerGoga and MegaCortex ransomware
- Nine corporations have been identified as victims so far
- Started an affiliate program (Ransomware-as-a-Service) in January 2020
- Two variants have been observed — one written in C# and one in C++ — and LockBit may be delivered with UPX or a custom packer.
- Terminates a list of security/protection services and processes before starting the encryption routine.
- Uses a variety of cryptographic algorithms, from simple XOR to TEA and AES-NI, to decrypt strings and encrypt files.

Intro

First observed in September 2019, LockBit employs a Ransomware-as-a-Service (RaaS) model and is used in targeted attacks against enterprises. There are two versions of the ransomware, one written in C# and one in C++. The latter is the latest version, and uses a custom packer. The LockBit sample we analyzed uses the rarely-seen Tiny Encryption Algorithm (TEA) to decrypt its code.

Attack vector

After compromising an IIS web server, the attackers run a PowerShell script that launches another script embedded in a Google Sheets spreadsheet. The latter script connects to a command-and-control (C&C) system, retrieving one final script that deploys a backdoor — through which LockBit is download and installed. The attackers also check if the target system has specific fingerprints to deploy a ransomware. Once the target is verified, WMI is used to deliver the copies of ransomware to Windows computers in the internal network.

Deobfuscation and unpacking

LockBit spans two levels of unpacking — first decrypting shellcode and allocating it in the memory, and then deciphering the executable itself and passing execution to it. This output from KryptoAnalyzer shows that LockBit uses TEA (also known as TEAN) to block cipher encryption.

Looking at the code in a disassembler, we see mostly junk, except for the TEA decryption routine and lpAddress, which is located at the end of the main function.

Typical of the TEA decryption procedure, the key is split into four parts. Decryption also uses the summed value, 32, and a delta of these values.

Once the shellcode is allocated, LockBit resolves API function by providing hashes of library and function.

LockBit refers to PEB structure to enumerate dynamic link libraries imported by the executable.

After retrieving the DLL name in Unicode, the ransomware performs custom operations to calculate a unique hash value, which is compared with the hardcoded hash.

After the DLL is determined, LockBit enumerates exported functions of the current DLL by its ordinals and calculates its hashes, unless the hash is matched with the value stored in the stack.

After resolving two main functions, LoadLibrary() and GetProcAddress(), the ransomware decrypts the executable. As shown in the following screenshot, LockBit contains hardcoded Unicode values. These are hash values of the imported functions. The ransomware employs this obfuscation technique to hide the names of the imported functions in IAT during runtime linking.

Finally, LockBit uses CreateToolhelp32Snapshot() to retrieve all the modules for the current process and Module32First()to obtain the current module and decrypt obfuscated parts of the ransomware, using the custom algorithm that includes shifting operations.

The unpacked LockBit starts with the following piece of code:

To hide the strings, LockBit also stores them in an encrypted form:

The strings above are also hex-encoded. The following code snippet performs a decryption, where the whole string is XORed with the first byte.

Anti-debugging and other checks

When starting, LockBit checks *NtGlobalFlag* for the *FLG_HEAP_VALIDATE_PARAMETERS*, *FLG_HEAP_ENABLE_TAIL_CHECK*, and *FLG_HEAP_ENABLE_FREE_CHECK* values used by the debugger to control the heap state. This clearly indicates whether the malware is being analyzed, allowing LockBit to react accordingly.

The next check is related to the system locale. LockBit executes *GetSystemDefaultUILanguage()* and *GetUserDefaultUILanguage()* functions to verify the locale on the victim system, with a predefined list containing 11 countries of the former Soviet Union — Russia, Ukraine, Belarus, Tajikistan, Armenia, Azerbaijan, Georgia, Kazakhstan, Kyrgyzstan, Turkmenistan, and Uzbekistan.

If the locale does not belong to the list above, the ransomware moves to the next check. At first glance, the following piece of code might seem like an anti-VM check, but it is used for a different purpose. LockBit retrieves information regarding the CPU to check if the system supports F16C used for AES-NI. In short, it is a set of instructions used for AES encryption acceleration at CPU level.

Overall, LockBit checks the most common processor vendors such as GenuineIntel and AuthenticAMD. It also checks for the value of 29th bit not equal to “0” which indicates “F16C” support for half-precision floating-point conversion.

The most interesting technique used by the ransomware is the anti-debugging method which crashes the debugger. After declaring an I/O completion port, LockBit creates two threads per processor and hides them from the debugger. Any further operations with processes or threads cause immediate termination of the debugger.

LockBit ensures that only one instance is running in the system by checking whether the following mutex exists:

‘Global\{BEF590BE-11A6-442A-A85B-656C1081E04C}’

Privilege escalation

LockBit tries to elevate user privileges. First, it checks if the process is already running with administrator rights.

If not, the ransomware checks the process’ privileges through the following steps:

1. LockBit obtains an access token associated with the current process using NtOpenProcessToken().
2. LockBit creates a SID belonging to the Administrator group using CreateWellKnownSID().
3. Having the process access token and SID, LockBit checks whether the SID is enabled in the access token.
4. If it fails, LockBit retrieves information regarding the linked token using NtQueryInformationToken() and again checks whether the SID is enabled .

LockBit can also try checking the major version number of the operating system using GetVersion(). If this is 5, indicating Windows 2000, the process can get administrator privileges without any additional steps, because UAC was only introduced in Windows Vista.

If none of these methods work, LockBit performs a UAC bypass procedure through COM objects. First, LockBit impersonates the *explorer.exe* process which runs from the ‘C:\Windows\’ folder. Then, it creates a CMSTPLUA COM object by passing the following parameters to CoGetObject() function:

Next, LockBit executes the *combase_ObjectStublessClient10()* undocumented function, which is a stub function for COM proxies — on GitHub, the function is associated with SetRegistryStringValue(). LockBbit passes the following parameters to the function:

- Payload
- 80000002
- Software\Microsoft\Windows NT\CurrentVersion\ICM\Calibration
- DisplayColaborator
- Path to the LockBit executable

After that, LockBit creates a *ColorDataProxy* COM object:

Finally, it calls *combase_ObjectStublessClient14()* to launch the COM object, and then releases both the *ColorDataProxy* and *CMSTPLUA* COM objects using *rpcrt4_IUnknown_Release_Proxy*.

Now having administrator privileges, LockBit creates an access control list for the current process and adds an entry specifying administrator privileges.

After performing modifications in the process handle, the ransomware enables new privileges using RtlAdjustPrivilege() function.

As shown, LockBit has a number of methods through which it can obtain administrator privileges ahead of the encryption process.

Logging

LockBit initializes a console window to display debug information. This is probably an artifact of the ransomware's development that remained in the production version. By default, the window is declared as hidden in ShowWindow(), but this can be easily modified by changing the byte from 'SW_HIDE' to 'SW_SHOW'.

LockBit logs every execution step, including notifications about the state of processes' rights and the killing of processes and services.

After patching the byte, we can track the following log information provided by LockBit's authors.

Looking at the log above, we see a suspicious operation in the mounting of volume C:\ to the newly created Z:\. The following code is responsible for this:

If LockBit encounters any mounting problems, it will enumerate volumes from D:\ through Y:\ in a reverse order until the process succeeds.

Termination of processes and services

LockBit contains a list of 80 services and 99 processes to terminate before encryption starts. When LockBit stops a service or process, it writes the corresponding output to the console window.

The ransomware intends to stop AV (Sophos), backup, and SQL services. The following list defines services that LockBit attempts to terminate, including backup services offered by Acronis and Veeam. Acronis' self-protection capabilities prevent this action from succeeding.

This denylist also contains AV, backup, SQL, web application, and Microsoft product processes.

The ransomware additionally kills processes that deny access to files, unlocking them for further encryption.

Deletion of backups

Before starting encryption, LockBit removes shadow copies and backups, modifies Windows boot configuration data store (BCD), and clears security, system, and application events logs.

Lockbit performs these operations in two stages. First, it executes the following command:

"C:\Windows\System32\cmd.exe" /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wbadmin delete catalog -quiet

Second, LockBit starts the following processes, additionally targeting event journals:

Finally, LockBit empties the Recycle Bin.

Generating a victim’s identifier

LockBit also creates the registry key [SOFTWARE\Lockbit] to identify the victim’s system. The key contains randomly generated *full* and *Public* values.

The value of the *full* registry key is one byte in length:

The value of the *Public* registry key is 258 bytes in length:

LockBit checks for *full* and *Public* keys at runtime. If not found, these are generated again using BcryptGenRandom().

As mentioned earlier, LockBit uses I/O ports to speed up the encryption process.

File encryption

LockBit uses AES-128-CBC for file encryption. The program first generates a 16-byte key and IV for each file.

If AES-NI support is enabled, LockBit uses the processor‘s AES library instructions to encrypt files with AES-CBC-128.

Otherwise, it checks the number of rounds — which should be 10 — and performs AES encryption without using WinAPI functions.

LockBit defines the following lists of directories, files, and files extensions to be skipped during the encryption:

Another denylist targets additional file extensions:[ES1]

With the exception of files indicated on these denylists, LockBit encrypts the entire system as well as network resources. Additional individual files and directories can be marked for encryption through the command line:

The following screenshot shows the contents of the encrypted directory:

Each file contains a special LockBit marker unique to each sample, as well as the contents of the full registry key:

LockBit creates the *XO1XADpO01* registry parameter in [SOFTWARE\Microsoft\Windows\CurrentVersion\Run] with a path to the LockBit executable. If the victim shuts down the system during encryption, the executable will run after the system boots. When encryption completes, LockBit removes the key and performs auto-deletion by executing the following command:

"C:\Windows\System32\cmd.exe" /C ping 127.0.0.7 -n 3 > Nul & fsutil file setZeroData offset=0 length=524288 "C:\Users\User\Desktop\lockbit\lockbit.exe" & Del /f /q "C:\Users\User\Desktop\lockbit\lockbit.exe".

LockBit even changes the wallpaper on the victim's desktop, letting them know where to find the ransom note.

Monitoring the registry changes shows us the location of the new wallpaper:

Ransom note

Before LockBit completes execution, it opens an HTML page that can be displayed as a standalone window using *mhta.exe*.

The *.hta* ransom note looks as follows:

Additionally, the ransomware adds the ransom note to the autorun key [HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run]:

LockBit also deposits the *Restore-My-Files.txt* ransom note into every folder with encrypted files. The contents of this note are shown below:

The ransom note contains the following links:

- hxxp://lockbit-decryptor.top/?85C01E35FD24495CA4FA9D603F156529 — opens the LockBit site in any browser
- hxxp://lockbitks2tvnmwk.onion/?85C01E35FD24495CA4FA9D603F156529 — opens the LockBit site in the TOR browser only

Both URLs contain the victim ID, which in this case is *85C01E35FD24495CA4FA9D603F156529*. The first eight bytes are from the LockBit marker and the next eight bytes are from the public registry key.

The decryption site looks as follows:

As proof of ability to restore encrypted data, LockBit offers to decrypt a single file for free:

Detection by Acronis

Acronis' Active Protection technology uses advanced, AI-driven behavioral analysis to successfully identify and stop LockBit attacks — as well as any other known or unknown cyberthreats. Backups are protected against tampering, and enable the automatic and rapid restoration of any encrypted files.

Conclusion

LockBit is an advanced piece of ransomware used in targeted attacks, and is capable of elevating privileges in the victim's system and leveraging the anti-debugging, anti-analysis, anti-AV, and anti-backup capabilities while adopting AES-NI acceleration to speed up the encryption process.

IoCs

MD5: 7f0312a1f928c3aeab672ca8d5afc6a9

SHA256: 43ced481e0f68fe57be3246cc5aede353c9d34f4e15d0afe443b5de9514d3ce4

MD5 (unpacked): eb9176b89f8a96d3963628b21b87c07d

SHA256 (unpacked):
ea5f8b184783979f3e32802b6942525b9f75cefae9d6e527c493a340cfc57c73

MD5: eb9176b89f8a96d3963628b21b87c07d

SHA256:
EA5F8B184783979F3E32802B6942525B9F75CEFAE9D6E527C493A340CFC57C73

Mutex name = Global\{BEF590BE-11A6-442A-A85B-656C1081E04C}

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\XO1XADpO01

HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run\{2C5F9FCC-F266-43F6-BFD7-838DAE269E11}

SOFTWARE\Lockbit\full

SOFTWARE\Lockbit\Public

C:\Users\User\Desktop\LockBit-note.hta

Restore-My-Files.txt

.lockbit

Usage of AES-NI: AESKEYGENASSIST, AESENC, AESENCLAST.

Software\Microsoft\Windows NT\CurrentVersion\ICM\Calibration\DisplayCalibrator

{D2E7041B-2927-42fb-8E9F-7CE93B6DC937}

{3E5FC7F9-9A51-4367-9063-A120244FBEC7}

[ES1]Aside from the obvious content differences, I'm not quite understanding how this list differs from the ones defined above — do both contain file extensions that are skipped over during encryption? If not, what does the following list describe?

Share

Previous post Acronis Next post

About Acronis

A Swiss company founded in Singapore in 2003, Acronis has 15 offices worldwide and employees in 50+ countries. Acronis Cyber Protect Cloud is available in 26 languages in 150 countries and is used by over 20,000 service providers to protect over 750,000 businesses.

RANSOMWARE PROTECTION MSP CYBERSECURITY

Stay up-to-date

Subscribe now for tips, tools and news.

More from Acronis

October 31, 2024 — 5 min read

Highlights from MSP Global 2024 in Barcelona

Two words dominated the show floor at MSP Global 2024: Go native. But what does that mean? It's a reference to natively integrated cybersecurity, data protection and endpoint management offered in a single, comprehensive solution. In this blog, we reflect on key takeaways from the exciting and eye-opening event.

October 24, 2024 — 5 min read

Ghost in the machine: When cyberattacks go undetected

The prospect of any data breach is scary, but the thought that an attacker could fundamentally take over an organization's data for weeks or months is downright terrifying. Read on to learn how you can protect yourself from these cyberattacks.

October 23, 2024 — 10 min read

What is security patching and why it is important for business sucess

Discover the importance of regular security patching in 2024 and how to stay ahead of evolving cyber threats. Understand the security patching process, from identifying vulnerabilities to applying patches seamlessly with the best practices for effective patch management, ensuring your systems stay updated and secure.

Engage with Acronis