Diving into Lockbit's Arsenal

May 2024

Contents

Forward

- The LockBit ransomware group, first emerged on September 2019 under the name of ABCD ransomware group, receiving its name from the notorious abcd extension left on the files it has encrypted. As of 2022, LockBit was the most deployed ransomware variant across the world, and until 2023, has managed to maintain its position as the intrusion set with the largest number of attacks.
- Nevertheless, the year 2023 marked a watershed in LockBit's market share, as its overall share of ransomware impact has seen a steady decline over the last two years[i], due to a series of logistical, technical, and reputational problems. LockBit's reputation as one of the top RaaS providers, is inextricably linked to its ability to recruit affiliates to conduct its ransomware operations, using its tools and infrastructure.
- Unfortunately, even in light of the joined law-enforcement effort, 'Operation Cronos' against LockBit's infrastructure and operators, INCD still receives reports from organizations suffering from LockBit attacks.
- A substantial part of the report will dive into the technicalities of LockBit 3.0 builder, and suggest a possible vaccination against the ransomware, to prevent the malware to execute, and additional victims to suffer.
- Finally, the purpose of the following paper is to explore a possible linkage between the development of new tools and the efforts to maintain their ransomware affiliate model. The report will cover the variety of tools developed and deployed by LockBit, and similarities to other known malware of the underground.

Part I : Technical Analysis of LockBit3.0

The following analysis is based on LockBit 3.0 leaked builder.

Builder Usage

1. Using the builder, keygen is used to generate a keypair, private and public, for the affiliate's usage. The keys are used to generate .enc and .dec files.
2. LockBit 3.0 has two modes of operation: with, or without a password.
3. To evade detection by sandboxes, a password argument is used. The password is randomly generated by the builder, thus rendering it impossible to reverse.

Main Ransomware Logic

In the main function of the executable file, the malware first parses the passed arguments through the command line.

Argument	Description
-path	Accepts additional argument, path for which the malware will specifically run at and encrypt its contents
-pass	Accepts additional argument, a password, used for encrypting pages
-safe	Attempts to force running in safe boot, thus enabling additional capabilities
-wall	Change screen saver background
-gspd	Lateral movement via GPO
-psex	Lateral movement via PsExec
-del	Complete deletion of file, according to provided flags from configuration file
-gdel	Erase from GPO

Disrupting Windows Defender

1. If the `kill_defender` flag is set, the malware attempts to obtain a handle for the process `TrustedInstaller.exe` (the process used for installing modules). If such flag is not provided, the malware reboots the service.
2. The malware attempts to copy the process' token, using it to iterate over all active services on a given machine, and to shut down all services related to Windows Defender.
3. Next, the malware terminates the `SecurityHealthSystray.exe` process, responsible for presenting the Windows Defender status icon on system toolbar.

Deleting the Recycle Bin

The malware creates a thread, which finds all folders defined as recycle bin, iterates over the folders' content, alters the content for randomized unreadable content, then deletes the file. Thus, restoring the file is impossible.

Deleting Snapshots

1. To prevent restoration of information using existing snapshots, the malware creates a thread that sets up a COM object with a `IWbemServices`. Then, `ExecQuery` function is used to WQL query the following:
   `SELECT * FROM Win32_ShadowCopy`
2. Then, results are iterated to delete the ShadowCopy, using the `DeleteInstance` method of the COM object.

Deleting Processes and Services

1. If the `kill_services` flag is set, the malware iterates over active services in the machine.
2. If one of the active services matches the defined services in the configuration file, the malware terminates the process, and deletes it using `DeleteService`.
3. Similarly, the malware creates a thread for pre-defined processes in `kill_processes`, but keeps the thread running in a loop, to ensure defined processes don’t run while encryption is occurring.

Defining Execution Environment

Importing Configuration File

1. The configuration file exists in memory in an encrypted, compressed form, with the malware's unique sequence cipher.
2. After being decrypted completely, the malware fills out the global variables which are used in the order they appear in the file, which made researching the malware easier.
3. Note that the encryption does not occur on the first two bytes in the configuration, as these bytes are used to encrypt the sequence.

Language Check

If the flag `language_check` is enabled, the malware reads the `InstallUILanguage` and `DefaultUILanguage` settings, and terminates execution if one of the following is true:

- Russian
- Ukrainian
- Belarusian
- Tajiki
- Armenian
- Azeri (Arabic / Cyrilic alphabet)
- Georgian
- Kazakh
- Turkmen
- Uzbek (Arabic / Cyrilic alphabet)
- Tatar
- Moldovan (Russian / Romanian)
- Arabic (Syria)

UAC Bypass

1. To suffice the privileges to execute in, the malware checks if its process token is part of the group `S-1-5-20`, (SID ID for Network Service, which requires high privileges when executing in a network context).
2. In addition, the malware iterates on all groups in which it's a member, to check if one of them is System Domain Admin.
3. If such privileges do not exist or the OS is older than Windows 7, the malware performs privilege escalation via `CMSTPLua`.
4. Such method is known to be used to achieve privilege escalation. The execution includes defining the `ImagePathName` and `ProcessParameters` in the process' PEB as dllhost.exe, allowing COM object execution in high privileges.
5. The malware uses `ShellExec` function in `ICMLuaUtil` Interface of `CMSTPLua` object to re-execute itself with high privileges.

Creating File Extension

The encrypted files and the ransom note dropped in each folder receive a unique extension. The ransom note contains the generated extension as a prefix, and 'README.txt' as the file's suffix. The file extension is generated in the following manner:

1. MD5 hashing the pubic RSA key, defined in the malware's configuration.
2. Formatting the signature as GUID in the pattern:

`{%02X%02X%02X%02X%02X%02X%02XX-%04X-%04X-%02X%02X-%08}`

3. MD5 hashing the created GUID.
4. Encoding the MD5 hash using base64.
5. Replacing special characters ("=","/","+") in the encoded string.

Finding High Privilege Token

1. In order to receive a token, which will be used by the malware to execute high privilege processes, first the malware checks if its `TokenUser` has the value of `S-1-5-18` (SID ID for user System).
2. If the token is indeed a System user, the malware defines the highest privilege token as the `ActiveSessionId` of the current process. This is achieved by accessing the hist `x2d80` in the structure `KUSHER_SHARED_DATA`, found in memory on `x7ffe00000`.

Anti-Debugging Methods

Nearly every value in the malware is being XORed using the key `0x4803BFC7,` even if a given value does not mean anything by itself, or is a part of some other part of encryption/obfuscation method.

Dynamic API Resolving

As expected from ransomware, most API functions which the malware uses are imported while execution through shellcode hashing mechanism.

Encrypting Strings and GUID Values

Besides using XOR encryption, as mentioned before, a substantial part of the strings and values of importance in the malware are being encoded in the following manner:

```
do
{
     *buffer ^= 0x4803BFC7u;
     *buffer = ~*buffer;
     ++buffer;
     --count;
}
while ( count );
```

`decode_str` routine: XOR encryption with aforementioned key, following a logical NOT operation.

Obfuscating comparison Using Hashing

Most comparison actions between strings in the malware are executed using ROR13 hashing, after lowercasing all chars in the strings.

```
HIWORD(chr)=0;
do
{
     LOWORD(chr) = *str++;
     If( chr >= 'A' && chr <= 'Z' )
          LOWORD(chr) = chr | 0x20;
     val = chr + __ROR4__(val, 13);
}
while ( chr );
return val;
```

ROR13LOWER function, as appears in LockBit's code

Encrypting Buffers

Relatively large buffers are being sequence encrypted in a unique manner. The first two DWORDs in the configuration file (as it is in memory) are being used as initial state, then using a linear function to update them for each iteration. The mentioned cipher is breakable, however of little reversing interest, as the encryption is used for internal processes in the malware itself.

Data Compression

Large pieces of data, like shellcodes and the configuration file, are compressed in the malware using aPlib algorithm. Every time they need to be used; the malware decompresses them. For most, the information is encrypted using the aforementioned algorithm, used for the buffer encryption.

API-Wrapping Processes

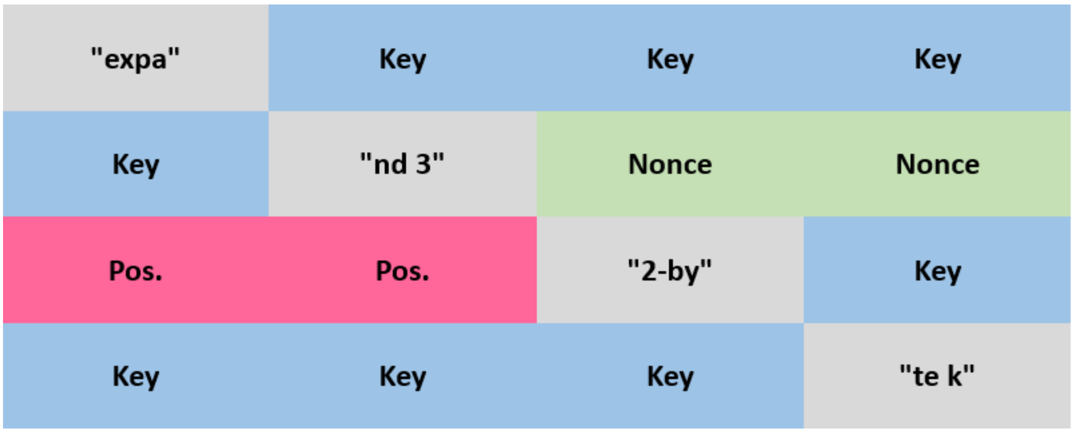
1. Besides processes responsible for file iteration and encryption, used to improve the malware's efficiency and latency, the ransomware implements a variety of processes which call the `WinAPI` function.
2. Such processes, though presenting a substantial overhead in relation to simple reading, allow process mimicking while running.
3. For this reason, it is believed that such functionality was made to evade detection. Nonetheless, multi-threading makes dynamic analysis more difficult.

Encryption Method

1. The encryption occurs using a queue, which passes between threads. One of the threads is responsible for tracing files to encrypt and queuing them, with additional information to assist the adversary in file decryption (when needed).
2. The encryption mechanism is comprised of three keys, a ”checksum,” and two additional functions responsible for key randomization.
3. Thus, there exists one global key, another key which changes every 1000 files, and a key which changes for every file being encrypted.
    a. A public RSA 1024 key, used to encrypt the second key.
    b. A key which changes every 1000 files being encrypted, used to encrypt third key.
    c. A key which changes every file being encrypted, used to encrypt actual file content.

Usage of Salsa Algorithm

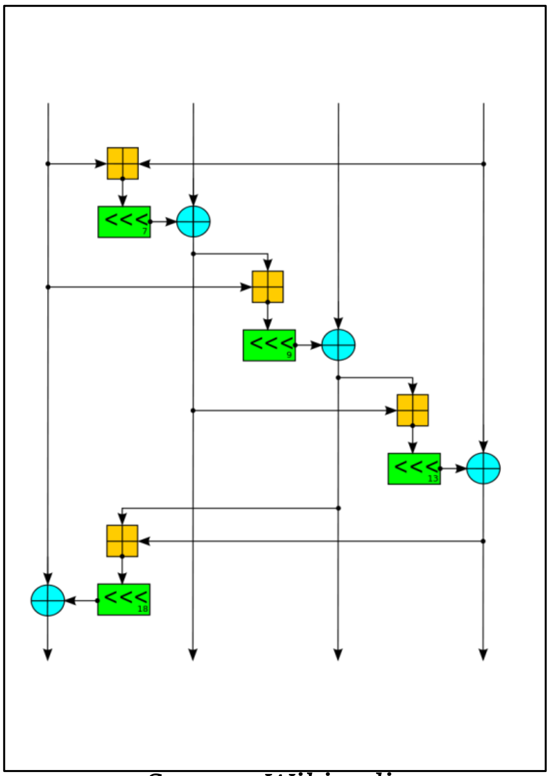
1. Salsa20 is a sequence cipher. Meaning, from a small key, a sequence of keys of an arbitrary size is being created.
2. Initial state of salsa is comprised of the following bytes:



"expa"	Key	Key	Key
Key	"nd 3"	Nonce	Nonce
Pos.	Pos.	"2-by"	Key
Key	Key	Key	"te k"

    a. Grey colored bytes are constant
    b. Blue colored bytes are comprised of original key
    c. Green colored bytes are constant determined by IV
    d. POS is determined by the IV and is changed every iteration

3. The state with according POS enters the block function, which is comprised of several rounds. Each round executes the following action four times:



Source: Wikipedia

4. LockBit's implementation is unique. It uses the same block functions; however, the initial state is initiated in peculiar manner.
5. Instead of using an 8int size key, meaning 32 bytes long. The randomized Salsa keys (second and third keys), 128 bytes long, are used for the initial states of Salsa20.
6. IV usage in Salsa:
    a. While encrypting the file, the POS moves just like the original Salsa. Meaning, the IV increases with each encryption, and does not use twice the same sequence to encrypt different parts of the file.
    b. Encrypting the third key using Salsa changes the POS locally. However, this does not change the rest of the files. Meaning, all 1000 keys encrypted by the second key have been XORed into the same constant.

Randomizing Methods

1. Naturally, all randomizing methods use a basic function, which depends on the processors' features. whether its `rdrand` or `rdseed`.

2. If none of the two exists, ROR13 is used on the rdtsc output is used. The meaning of such is, a very bad implementation of a randomizing method.
3. The third key randomizing method is executed in a quite strange manner. Only first 64 bytes are being randomized, while the rest of the key is created using a linear function.
4. The second key randomization method is executed in a more coherent manner – all int32 are randomized by creation.

Encryption Process

1. The malware does not encrypt all file content. Instead, it encrypts 0x200000 size chunks. By calculating the file size, the malware decides how much of the file to encrypt, and which bytes to skip.
2. The thread responsible for file encryption has three actions, depending on state of request in queue:
   a. State 0: buffering the required file content for encryption on queue.
   b. State 1: decrypting the key using RtlDecryptMemory, encrypting the buffer using the key, updating IV and finally encrypting the key again. If there is additional data to encrypt, state is changed to '0'. If not, outputting results of encryption, then state is changed to '2'.
   c. State 2: adding encrypted key to the file, with additional information such as checksum (size int32) to assist recognizing whether the file has been encrypted.

C&C Communication

Communication Module

URL	Protocol	Function	Notes
test.white-datasheet[.]com	HTTP/S	C2	Hardcoded URL in configuration file for HTTP/S communication.

As found in analyzed samples.

If the send_report flag is enabled, the malware attempts to send reports to specified URLs in the field gate_urls found in configuration file. There are two types of reports, with different formatting.

Execution Initiation Report

Before the malware initiates encryption, it sends to the server general information regarding the machine it executes on, in the following format:

```
{
     "bot_version":"%s",
     "bot_id":"%s",
     "bot_company":"%.8x%.8x %.8x %.8x%",
     "host_hostnames":"%s",
     "host_user":"%s",
     "host_os":"%s",
     "host_domain":"%s",
     "host_arch":"%s",
     "host_lang":"%s",

     "disks_info":[
          {
               "disk_name":"%s",
               "disk_size":"%u",
               "free_size":"%u",
          }
     ]
}
```

All fields are filled by reading registry values and using API functions. In addition, for every disk on the machine, an additional value will be generated in the `disks_info` array in the JSON report.

The field `bot_id` is created by MD5 hashing the public RSA key, transforming it to GUID, then ROR13 lower the value, and finally MD4 hashing the final value.

Execution Termination Report

After the encryption process is terminated, the malware sends a summary report in the following format:

```
{
     "bot_version":"%s",
     "bot_id":"%s",
     "bot_company":"%.8x%.8x %.8x %.8x%",
     "stat_all_files":"%u",
     "stat_not_encrypted":"%u",
     "stat_size":"%s",
     "execution_time":"%u",
     "start_time":"%u",
     "stop_time":"%u",
}
```

All stat fields are updated using global fields during encryption process.

Exfiltration

1. To send aforementioned report, LockBit encrypts the information using the AES key obtained from configuration. Then, encodes using base64.
2. After randomizing an HTTP object, in addition to randomizing information in the POST request, next to encrypted and encoded report; the information is sent to a server.
3. The described HTTP routine is done using `Win32API` functions.

Identification and Disruption Means

Vaccination

1. A set of action(s) to be done in your environment, to render the ransomware's functionality unusable and prevent its proper execution and encryption of data.
2. INCD wishes to share possible vaccination methods to the public domain, to mitigate potentially affected and threatened organizations.
3. Enabling the system flag (S) for every folder wished to be vaccinated:
    a. `S+ <FolderPath> attrib.exe<C:\Windows\SysWOW64>`
4. After the malware executes in `FolderPath`, all sub-folders and files are not encrypted.
5. This can be achieved thanks to the malware skipping folders with a system flag enabled during iteration on the file system.
6. Important to note, all folder contents can still be changed, copied, deleted, renamed etc. as before.

Part II : LockBit's Tool Arsenal

LockBit 2.0

In June 2021, the second version of **LockBit (LockBit 2.0)** was released, also known as **LockBit Red**, including a built-in data exfiltration tool called **StealBit**[ii,iii], allowing affiliates to easily exfiltrate victim data, which also serves as a backdoor. Written in the C programming language, it has string and API-obfuscation capabilities along with anti-debugging features to impede analysis. As part of the tool modus operandi, it first exfiltrates data from compromised windows systems, and then encrypted files with LockBit ransomware. One of the main issues LockBit aimed to address when developing its own data exfiltration tool, was introducing affiliates with the ability to quickly transfer large amounts of data, without being detected. Initially, the LockBit ransomware payload relied on legitimate publicly available tools to exfiltrate and steal data, as well as legitimate online data storage and distribution services, which alleviate law enforcement in the process of denying access to such service providers. To address these issues, StealBit uploaded data to LockBit data leak site, as well as exfiltrating files in parallel for efficiency, implementing inter-process communication for scalability, and supporting the dragging and dropping of files or folders for convenience. Most importantly, StealBit simplified the process of stealing and exfiltrating data, by providing affiliates with a central management console, incorporating multiple attack features, allowing affiliates to target a certain file path that is copied to a server, using HTTP controlled by the attacker[iv]. Once StealBit reads file content using the ZwReadFile function, available StealBit worker threads, commonly used to handle background tasks that does not disrupt the continued use of the application[v], exfiltrate the read file content by issuing the HTTP 1.1 PUT request to an attacker-controlled endpoint, where the exfiltrated files content is stored[vi].

Moreover, unlike recent versions of the tool which are executable on any system, older variants will not execute on systems located in former Soviet countries. A plausible explanation might be that at the time StealBit was released, most if not all members of the LockBit ransomware group, came from former Soviet countries, however by the time recent variants were released, the group had broadened the scope of its affiliates. Moreover, access to StealBit was only granted to vetted affiliates, and as per March 2023, the malware was no longer listed in the affiliate panel.

LockBit Linux-ESXi

In October, 2021, LockBit introduced a new tool called **LockBit Linux-ESXi Locker version 1.0**, in the underground forum "RAMP", where potential affiliates could find it. Equipped with new expanded capabilities to target systems to Linux and VMware ESXi, this tool signified the group's effort to expand its impact to victim organizations, as ESXi, VMware's hypervisor helps in managing servers. In addition, this tool is using a combination of Advanced Encryption Standard (AES) and elliptic-curve cryptography (ECC) algorithms for data encryption. This version of LockBit also accepts parameters and possesses logging capabilities. Later, an improved version of the tool was released named **LockBit Linux/ESXi locker version 1.46**, with the capability to extract and encrypt files from VMware ESXI virtual disk storage. Once it identifies the virtual machines executed on ESXI servers, it locates the virtual disk files containing the storage, and enrolls them in LockBit Linux's configuration exclusion lists. This technique prevents the ransomware from encrypting critical files and destroying Windows VMs managed by ESXi server. This capability to encrypt files within VMs, allow fully preventing extraction or recovery of data, that might be left unencrypted due to the ransomware's default behavior of encrypting data chunk in VM disk storage files. This newly added feature is aimed at providing affiliates with an enhanced capability to disrupt victims' recovery efforts during ransomware operations (assessed with medium confidence).

LockBit Black 3.0

On November 2021, another prominent RaaS provider called BlackMatter, announced it was shutting down its operation, and given the strong working relationship it shared with LockBit, they pushed its affiliates to transition to LockBit operation as well as its most recent victims to LockBit chat portal, to continue the negotiation process. As it turned out, LockBit recruited one of BlackMatter's most valuable resources, its developer, which soured the relationship between the two gangs. Six months later, this developer created a new ransomware variant known as **LockBit Black 3.0**, which shares similarities with BlackMatter and Alphv (also known as BlackCat) ransomware[vii]. Some of the overlaps identified with BlackMatter include bypassing User Account Control (UAC) by leveraging the CMSTPLUA COM interface, hiding threads from a debugger via NtSetInformationThread, and using encrypted pointers for API function resolution. In addition, as identified by security researchers, LockBit Black and BlackMatter both apply an additional anti-debugging technique, where the malware inspects whether 'heap tail checking' is enabled, by searching for the constant 0xABABABAB at the end of an allocated heap block, and exiting if it's found. Moreover, to properly execute LockBit Black, a key with the "-pass" argument must be provided for decrypting the main ransomware code, adding another layer of complexity for researchers when analyzing recovered ransomware samples. The following trend of introducing execution keys to hinder analysis, was prevalent among ransomware families such as SODINOKIBI and ALPHV, which might suggest that a common actor has worked for both RaaS programs.

On the 21st of September, 2022, the LockBit ransomware group has suffered a breach, after a Twitter user named @ali_qushji, stated that his team was able to hack several LockBit servers, where they found **LockBit 3.0 builder** and shared a password alongside a copy of the builder source code, via a file-hosting website and storage platform called Sendspace[viii]. Subsequently, VX-underground, a platform dedicated to malware research and cybersecurity education, shared that they were contacted by a user named "Proton", aka "protonleaks", who shared a copy of the builder, claiming it was leaked by an insider and not hacked by Ali Qushji[ix]. Furthermore, Jon DiMaggio, a chief security strategist at Analyst1, claimed he found an identical message to the Ali Qushji post, on a telegram channel linked to the Proton account. As soon as the news of the leak became public knowledge, hacking forums and markets became saturated with discussions regarding this unresolved mystery. Now affiliates as non-

LockBit affiliates, were also able to use LockBit 3.0, which allowed its users to quickly build the executables required to launch their own operation, including encryptor, decryptor, and specialized tools to launch the decryptor[x]. Only four days upon its release, on the 25th of September, cybersecurity researcher Vladislav Radetskiy reported the discovery of a new Bl00Dy Ransomware Gang encryptor, which had launched an attack against a Ukrainian entity utilizing the leaked builder[xi]. We were able to identify activity clusters of ransomware attack groups deploying the LockBit Builder in the Israeli cyberspace, including 'TronBit', 'CriptomanGizmo' and 'Tina Turner'[xii].

In an effort to restore its reputation, LockBit came up with an explanation for the leak, claiming it was a disgruntled employee who "regularly drank sedative pills, had mental problems, paranoia and constant stress, who was responsible for the leak. However, a more plausible explanation which may explain the motivation behind the leak, is that a few months earlier in July 2022, LockBit paid out 50,000$ to an individual who found a vulnerability in the LockBit Black code, which was originally present in BlackMatter and allowed files to be decrypted without the decryption key. The theory is that LockBit took that payment out of the developers' salary, since he failed to correct the flaw in LockBit Black, which provides us with a motive for why the developer betrayed LockBit[xiii]. It is also possible that the tale about the twitter user named Ali Qushji and his team, hacking LockBit infrastructure, was tailored to cover the real perpetrator behind the leak[xiv]. According to Jon DiMaggio, the developer who leaked LockBit Black, is the same individual who previously developed malware for Fin7, a gang which has also been linked to BlackBasta, and then created the DarkSide ransomware, which was utilized in the Colonial Pipeline attack, and later developed the BlackMatter ransomware. Two weeks after BlackMatter announced the retirement of its operation, the gang launched a new ransomware and a RaaS program called BlackCat (Alphv). Hence, all four gangs, namely DarkSide, BlackMatter, BlackCat (Alphv) and Fin7, share the same individuals who simply rebranded their operations, not to mention that LockBit itself confirmed that DarkSide's core members were the same individuals behind BlackMatter and BlackCat ransomware, and that the leadership of BlackBasta also ran the former Conti ransomware operation[xv].

LockBit Green

In January 2023, **LockBit Green** was released, which was thought to be a major new version, however various security experts quickly dispelled this belief, finding it to be a rebranded version of a Conti encrypter, and not a new LockBit 4.0 version[xvi]. According to SentinelOne's Antonio Cocomazzi, upon conducting an analysis of the LockBit Green sample, a significant overlap of 89% with the Conti Ransomware was found, specifically its v3 version. The command-line flags for LockBit Green, which constitute a common way to customize the behavior of a command-line application, are identical to those of Conti v3, making it a derivative of the original source code. Nonetheless, SentinelOne adds that a small portion of the LockBit Green source code has been modified by LockBit, including its ransom note, which is identical to the one used by the LockBit Black variant, directing victims to LockBit negotiation infrastructure. Second, this variant appends a random 8-character file extension, instead of the ".lockbit" extension used by the two previous LockBit variants[xvii]. As Cocomazzi concludes, this approach of modifying a source code written by reputable competitors is highly lucrative, as it saves time and money spent on the development of new tools, and instead diverts these efforts for the releasing of new tools, to attract new affiliates.

It is worth mentioning that the LockBit Green case further resonate an existing behavioral pattern in the gang's conduct, which dates back to December 2022, when LockBit began a collection and theft campaign against its criminal competitors, when it approached affiliates and personnel from the Royal ransomware gang, in an attempt to steal their builder. LockBit even admitted it wanted to acquire and integrate it into their service offering, and expand their arsenal of available ransomware variants, directly from the LockBit admin panel, which will provide additional options for affiliates to leverage during attacks. This new code of conduct, might provide a solution to LockBit's failure to adapt and expand its infrastructure to accommodate new affiliates, as well as addressing its existing development needs. In addition, in case defense mechanisms will block affiliates from deploying the LockBit ransomware, they could simply deploy an alternative RaaS program from one of LockBit's competitors, and resume operation. This important trend might disclose the future direction the gang is heading[xviii].

LockBit MacOS

As apparent from LockBit use of self-developed tools, they were designed for attacks on Windows, Linux, and VMware ESXi servers. Yet, on April the 16th, 2023, the prevailing belief that Apple products are almost immune to such attack was challenged, as @malwrhunterteam, a renowned group of cybersecurity researchers and analysts, provided details on their X account about a new sample of **LockBit MacOS (RAM) version 1.0**, compiled solely for the Apple ARM M1/M2 (aka Apple silicon) architecture. According to information posted on X by @vxunderground, the sample may have been compiled as early as the 17th of November, 2022[xix]. As for the archive, it contained test builds suggesting that LockBit 3.0 also has payloads not only for encrypting macOS ARM architecture, but also **FreeBSD, MIPS, and SPARC CPUs** architectures[xx]. The new MacOS variant was written in C and compiled as an ARM64 Mach-O binary. It is also important to note, that the MacOS variant is nearly identical to LockBit's Linux/ESXi version with only one exception- the MacOS variant includes a file-extension blocklist containing file extensions unique to MacOS. Nevertheless, the variant in question lacks any functionality for exfiltrating the data it locks, nor any method of persistence. Another point which further supports the argument that it's still a work in progress, pertains to the fact that the signature that verifies the reliability of the executable is invalid, namely, the Apple's Gatekeeper protections will prevent it from being run[xxi].

LockBit 4.0?

On February the 22nd, 2024, a new version of the LockBit ransomware was discovered, dubbed **LockBit-NG-Dev** (NG for Next Generation), likely to become

LockBit 4.0, which was discovered when law enforcement took down the cybercriminal's infrastructure. This new LockBit version was packed using the MPRESS packer, possibly in an effort to avoid static file detections. Other adversaries and malwares utilizing the MPRESS packer include the Emotet payload[xxii], TA505 cyber-criminal group[xxiii], DarkComet remote access trojan, and the Bisonal and Daserf malwares[xxiv]. When unpacked, it is evident that it was most probably has been written in .NET, and possibly compiled using CoreRT, in contrast to the usual C/C++ language used in previous LockBit versions. This variant also shares similarities with other LockBit variants, as it avoids encrypting certain directories, files and file extensions. Also, it terminates processes and services that may hinder the execution of the ransomware and the proper encryption of files, as well as inhibiting recovery from shadow copies and backups, by performing a set of routines prior to encryption. Like the LockBit Linux-ESXi Locker version 1.0, it also encrypts files using the AES algorithm, but also encrypts the AES key using RSA public key, which can also be found in the configuration, and each file is being encrypted by randomly generated AES keys. As for network encryption, if EnableNetworkShares is true, files on available network shares will also be encrypted.

On the other hand, this new variant also behaves differently than past versions, due to its ability to check if the current date is within the date range set in the configuration, and once inconsistency arises it will terminate the process. Upon examining the execution date range on the variant's embedded JASON format configuration file, which is contingent upon parameters such as MinDate and MaxDate, it appears that the date range is 01/06/2022-16/09/2023. There are two plausible explanations for this feature, one is that it allows LockBit to increase its profits at the expense of its affiliates, as it forces them to purchase a new version from the operators once the date expires. The second holds that this is an anti-analysis and anti-sandbox technique, however it is easier for an analyst to bypass this during reverse engineering than for an affiliate to patch the binary before using it against a victim[xxv].

[i] https://www.trendmicro.com/en_us/research/24/b/lockbit-attempts-to-stay-afloat-with-a-new-version.html

[ii] https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-165a

[iii] https://kcm.trellix.com/corporate/index?page=content&id=KB95252&locale=en_US

[iv] https://cyware.com/resources/research-and-analysis/lets-talk-about-lockbit-an-in-depth-analysis-7cf0

[v] https://learn.microsoft.com/en-us/cpp/parallel/multithreading-creating-worker-threads?view=msvc-170

[vi] https://www.cybereason.com/blog/research/threat-analysis-report-inside-the-lockbit-arsenal-the-stealbit-exfiltration-tool

[vii] https://analyst1.com/ransomware-diaries-volume-1/

[viii] https://medium.com/@lavanya.agre.cyb/ransomware-lockbit-3-0-code-leaked-on-github-now-anyone-can-start-their-ransomware-business-d3d49565b3ef

[ix] https://analyst1.com/ransomware-diaries-volume-1/

[x] https://analyst1.com/ransomware-diaries-volume-1/

[xi] https://www.bleepingcomputer.com/news/security/leaked-lockbit-30-builder-used-by-bl00dy-ransomware-gang-in-attacks/

[xii] https: //www.gov.il/BlobFolder/news/lockbit_3_0/he/lockbit_3_0.pdf

[xiii] https: //analyst1.com/ransomware-diaries-volume-1/

[xiv] https: //analyst1.com/ransomware-diaries-volume-1/

[xv] https: //analyst1.com/ransomware-diaries-volume-1/

[xvi] https: //www.bankinfosecurity.com/lockbit-ransomware-group-building-new-locker-before-takedown-a-24422

[xvii] https: //www.fortinet.com/blog/threat-research/lockbit-most-prevalent-ransomware

[xviii] https: //analyst1.com/ransomware-diaries-volume-3-lockbits-secrets/

[xix] https: //www.sentinelone.com/blog/lockbit-for-mac-how-real-is-the-risk-of-macos-ransomware/

[xx] https: //live.paloaltonetworks.com/t5/community-blogs/threat-alert-cortex-vs-lockbit-3-0/ba-p/545191

[xxi] https: //thehackernews.com/2023/04/lockbit-ransomware-now-targeting-apple.html

[xxii] https: //www.cynet.com/attack-techniques-hands-on/emotet-vs-trump-deep-dive-analysis-of-a-killer-info-stealer/

[xxiii] TA505 is Expanding its Operations - Yoroi

[xxiv] https: //attack.mitre.org/techniques/T1027/002/

[xxv] https: //www.trendmicro.com/content/dam/trendmicro/global/en/research/24/b/lockbit-attempts-to-stay-afloat-with-a-new-version/technical-appendix-lockbit-ng-dev-analysis.pdf