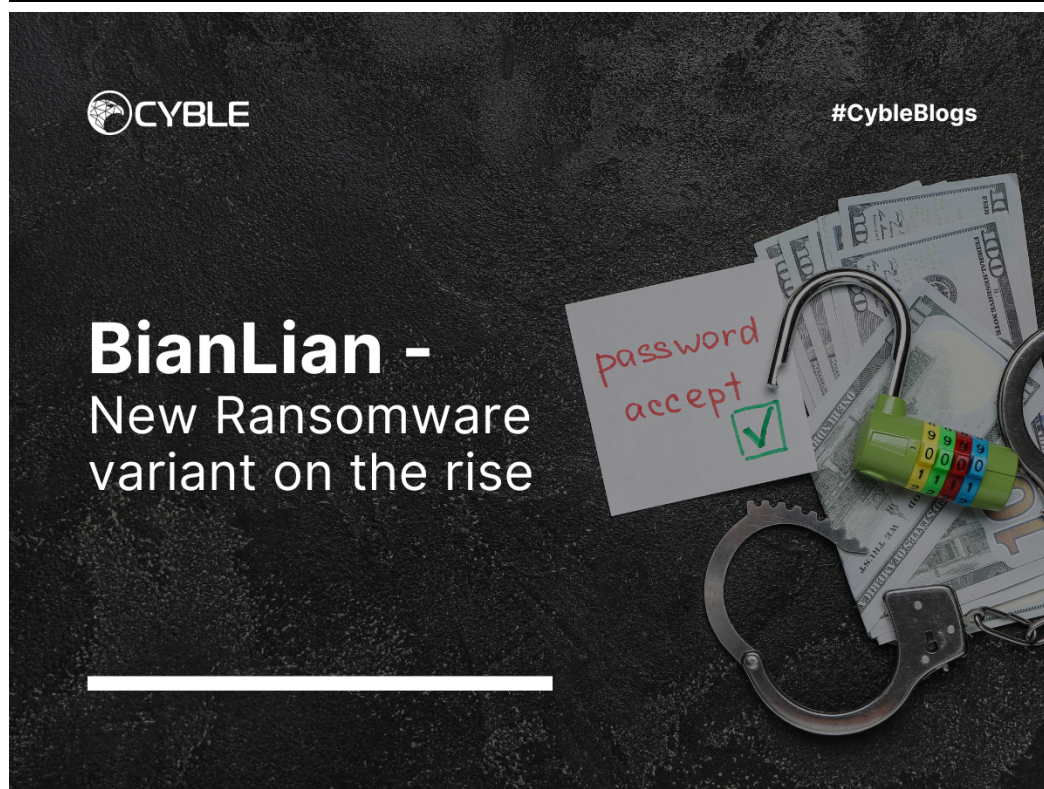


## BianLian: New Ransomware variant on the rise

: 8/18/2022



### GoLang-based Ransomware targets multiple industries

Cyble Research Labs has observed that malware written in the programming language “Go” has recently been popular among Threat Actors (TAs). This is likely due to its cross-platform functionalities and the fact that it makes reverse engineering more difficult. We have seen many threats developed using the Go language, such as Ransomware, RAT, [Stealer](#), etc.

During our routine threat-hunting exercise, we came across a [Twitter](#) post about a ransomware variant written in Go named “BianLian,” which was first identified halfway through July 2022.

The ransomware has targeted many well-known organizations (9 victims so far) across several industry sectors such as Manufacturing, Education, Healthcare, BFSI, etc. In the figure below, we have prepared a breakdown of the industries targeted by the BianLian ransomware.

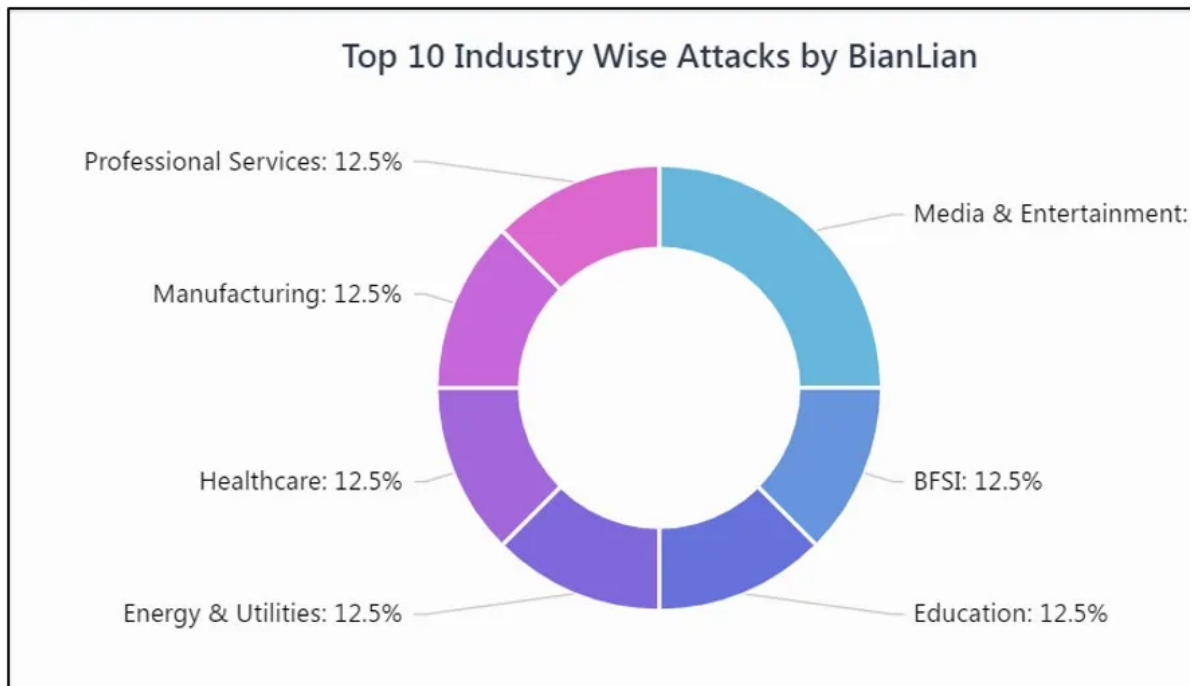


Figure 1 – Industries Targeted by the BianLian Ransomware

### Technical Analysis

We have taken the below sample hash for the purposes of this analysis: (SHA256), `eaf5e26c5e73f3db82cd07ea45e4d244ccb3ec3397ab5263a1a74add7bbcb6e2`, which is a 64-bit GoLang binary executable.

The unique build ID of the GoLang ransomware is shown below.

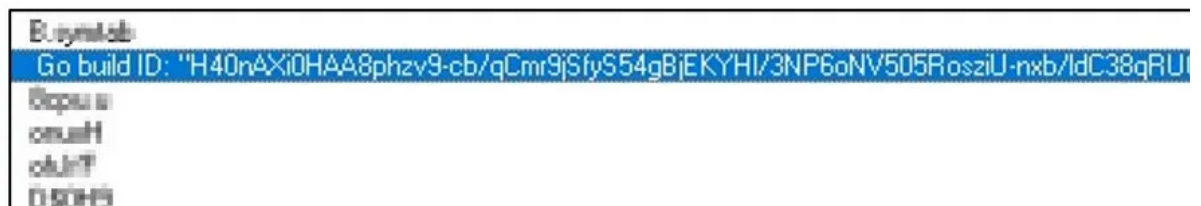


Figure 2 – Go Build ID

Upon execution of the ransomware, it attempts to identify if the file is running in a WINE environment by checking the `wine_get_version()` function via the `GetProcAddress()` API.

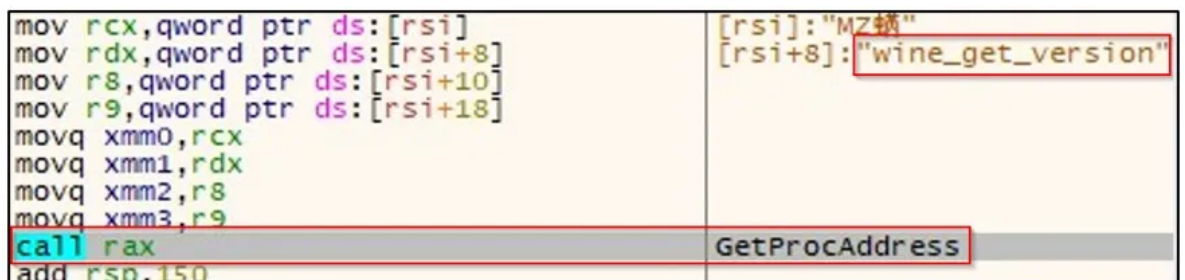


Figure 3 – Anti-analysis Technique

Then, the ransomware creates multiple threads using the `CreateThread()` API function to perform faster file encryption, making reverse engineering the malware more difficult. The below figure shows the multiple threads created by the ransomware.



Number	ID	Entry	TEB	RIP	Suspend Count	Priority	Wait Reason	Last Error	User Time	Kernel Time	Creation Time
44	16440	0000000000830FC0	000000C7744B0000	00007FF98F70BE24	1	Normal	Executive	00000000	00:00:00.0000000	00:00:00.0156250	16:02:13.990771
44	6644	0000000000830FC0	000000C7744B0000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0156250	00:00:00.0000000	16:02:49.340520
44	1816	0000000000830B00	000000C7744AF000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0781250	00:00:00.0468750	16:02:47.676250
44	3268	0000000000830FC0	000000C774501000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:01:56.943977
44	8016	0000000000830FC0	000000C7744F7000	00007FF98F70E5D4	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:48.692050
44	8175	00007FF98F6820E0	000000C774481000	00007FF98F70F7F4	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0156250	16:02:13.919131
44	7088	00007FF98F6820E0	000000C774483000	00007FF98F70F7F4	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:13.930939
44	7180	0000000000830FC0	000000C77448D000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0156250	16:02:13.985451
44	7256	0000000000830FC0	000000C7744F7000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:48.912841
44	8940	0000000000830FC0	000000C774481000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0312500	16:02:13.965300
44	7344	0000000000830FC0	000000C774489000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:13.966327
44	288	0000000000830FC0	000000C77450B000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:49.349811
44	9160	0000000000830FC0	000000C77448B000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0156250	00:00:00.0625000	16:02:13.984110
44	7112	0000000000830FC0	000000C774487000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:13.965650
44	4844	0000000000830FC0	000000C7744C1000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0156250	00:00:00.0468750	16:02:13.993911
44	2068	0000000000830FC0	000000C7744C3000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0156250	00:00:00.0156250	16:02:13.997499
44	2352	0000000000830FC0	000000C77450F000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:49.353851
44	8636	0000000000830FC0	000000C7744E3000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0312500	16:02:48.081988
44	9012	0000000000830FC0	000000C7744C5000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0312500	16:02:14.003060
44	2832	0000000000830FC0	000000C7744D7000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0156250	16:02:47.687880
44	6816	0000000000830FC0	000000C7744C7000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0312500	16:02:44.997120
44	4576	0000000000830FC0	000000C7744C9000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:44.997544
44	4252	0000000000830FC0	000000C774505000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:49.130088
44	8932	0000000000830FC0	000000C7744EB000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:48.290600
44	4396	0000000000830FC0	000000C7744CB000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:45.486231
44	6600	0000000000830FC0	000000C7744C9000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0312500	16:02:45.486920
44	1752	0000000000830FC0	000000C7744CF000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0312500	16:02:45.493330
44	6704	0000000000830FC0	000000C7744D3000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0156250	00:00:00.0156250	16:02:47.679000
44	6252	0000000000830FC0	000000C7744D5000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0156250	16:02:47.682200
44	340	0000000000830FC0	000000C7744D9000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:47.700644
44	6472	0000000000830FC0	000000C7744D8000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:47.704311
44	7700	0000000000830FC0	000000C7744D0000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0156250	16:02:47.716931
44	5920	0000000000830FC0	000000C774527000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:49.603177
44	5328	0000000000830FC0	000000C7744DF000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0156250	16:02:47.876330
44	3804	0000000000830FC0	000000C7744F9000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:48.692100
44	7352	0000000000830FC0	000000C7744EF000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0312500	16:02:48.301210
44	6904	0000000000830FC0	000000C7744E1000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:47.877220
44	3300	0000000000830FC0	000000C7744E3000	00007FF98F70BE24	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	16:02:48.082004

Figure 4 – Multiple Thread Creation

Next, the malware identifies the system drives (from A:\ to Z:\) using the *GetDriveTypeW()* API function and encrypts any files available in the connected drives. Then, the malware drops a ransom note in multiple folders with the file name "Look at this instruction.txt."

The ransomware creates a ransom note with the content shown below.

Figure 5 – Malware Writing Ransom Notes

After dropping the ransom note, the malware searches files and directories for encryption by enumerating them using the *FindFirstFileW()* and *FindNextFileW()* API functions.

The ransomware excludes the below file extensions and file/folder names from encryption.

- File extension .exe, .dll, .sys, .txt, .lnk and .html
- File names bootmgr, BOOTNXT, pagefile.sys, thumbs.db, ntuser.dat and swapfile.sys
- Folder names Windows, Windows.old

The ransomware uses GoLang Packages such as "crypto/cipher," "crypto/aes" and "crypto/rsa" for file encryption on the victim machine.

```

crypto/cipher.newCBC
crypto/cipher.dup
crypto/cipher.NewCBCEncrypter
crypto/cipher.(*cbcEncrypter).BlockSize
crypto/cipher.(*cbcEncrypter).CryptBlocks
crypto/internal/subtle.InexactOverlap
crypto/internal/subtle.AnyOverlap
crypto/cipher.xorBytes
crypto/cipher.init
crypto/cipher.xorBytesSSE2
crypto/aes.encryptBlockGo
encoding/binary.bigEndian.Uint32
encoding/binary.bigEndian.PutUint32
crypto/aes.expandKeyGo
crypto/aes.rotw
crypto/aes.subw
crypto/aes.KeySizeError.Error
crypto/aes.NewCipher
crypto/aes.newCipherGeneric
crypto/aes.(*aesCipher).BlockSize
crypto/aes.(*aesCipher).Encrypt
crypto/aes.newCipher
crypto/aes.(*aesCipherAasm).BlockSize
crypto/aes.(*aesCipherAasm).Encrypt
crypto/aes.init

```

Figure 6 – Hardcoded Strings of “Crypto” GoLang Packages

For encryption, the malware divides the file content into 10 bytes chunks. First, it reads 10 bytes from the original file, then encrypts the bytes and writes the encrypted data into the target file. Dividing the data into small chunks is a method to evade detection by Anti-Virus products.

The figure below shows the code snippet of the encryption loop and the original and infected file content before and after encryption.

The screenshot displays a debugger interface with three main panes:

- Original File:** Shows a hex dump of the original file content. The first few lines are:
 

```

0000h: 42 33 36 35 43 39 45 31 43 45 41 34 38 39 39 44
0000h: 43 45 43 31 34 33 33 38 46 45 22 C2 22 37 22 3A
0000h: 22 44 35 45 34 45 41 33 36 37 33 38 35 32 45 34
0000h: 37 36 45 42 42 30 37 43 30 38 46 39 42 36 39 36
0100h: 34 34 38 33 35 38 30 42 39 38 44 35 37 44 34 34
0110h: 43 33 42 32 34 34 38 34 35 35 42 34 30 44 43 39
0120h: 45 22 2C 22 35 22 3A 22 45 30 39 31 31 36 45 44
0130h: 34 38 39 45 32 39 46 44 36 38 33 35 45 38 32 43
0140h: 45 30 37 33 34 39 45 30 32 44 35 38 32 33 44 37
0150h: 44 34 45 45 32 44 39 37 44 39 35 30 34 33 39 32
0160h: 43 45 31 39 30 43 33 31 22 C2 22 33 22 3A 22 45
0170h: 39 33 33 38 32 37 38 44 41 36 36 37 35 36 38 42
0180h: 46 31 42 35 34 31 43 42 46 31 35 39 34 38 33 44
0190h: 36 36 32 39 32 44 46 46 41 45 33 30 42 31 37 42
01A0h: 36 35 35 32 32 31 45 35 35 45 38 35 43 43 32 22
01B0h: 2C 22 38 22 3A 22 44 35 45 34 45 41 33 36 37 33
01C0h: 38 35 32 45 34 37 36 45 42 42 30 37 43 30 B8 46
01D0h: 39 42 36 39 36 34 34 38 33 35 38 30 42 39 38 44
01E0h: 35 37 44 34 34 43 33 42 32 34 34 38 34 35 35 42
01F0h: 34 30 44 43 39 45 22 C2 22 36 22 3A 22 49 6E 74
0200h: 65 6C 28 52 29 20 38 32 35 37 34 C2 20 47 69 67
0210h: 61 62 69 74 20 4E 65 74 77 6F 72 6B 20 43 6F 6E
0220h: 6E 65 63 74 69 6F 6E 22 7D 0B

```
- Encrypted File:** Shows the same hex dump but with the content encrypted. The first few lines are:
 

```

0000h: C2 C2 3F D3 96 8F C5 DF 1D C6 55 48 7B 36 FA A3
0000h: FB FF FD 6D 7B 56 06 43 7F 84 2D CD F5 ED 8B B7
0000h: 57 62 38 A0 E7 F9 92 B6 99 37 19 2C 8D EA 83 DE
0000h: 3C EC 86 E1 6B 68 12 06 E3 7A 63 39 E9 25 10 9A
0100h: B5 2B 8A F1 C3 97 9F 49 A2 05 08 47 02 98 AC 69
0110h: 37 D6 90 A4 5E 5E D3 BA D3 1C 3A 04 D1 34 76 8F
0120h: 9D 67 5B F3 E9 B6 18 D6 25 57 79 36 B7 29 AC 89
0130h: A8 FC 55 20 5B 9A 00 99 D8 06 AD 08 82 BE A8 B1
0140h: 74 6C 30 8B B3 FE 7B 3F 49 1D BA 68 A0 91 31
0150h: 00 F0 5A 31 8E 0A 54 2F 49 9C 1E 4D 03 A3 41 D5
0160h: 99 DA 20 6E 7D FC 9D 65 C3 D2 96 6E 70 DE 6A 07
0170h: 99 20 84 88 D1 20 38 CB 84 CB BE CC 22 E0 D6 0B
0180h: 85 8F 0C 0A 47 B4 C6 97 65 CB 81 68 FA FC F1
0190h: 14 4D 9A C3 14 6F 10 93 01 84 F6 F6 85 C1 5B 66
01A0h: E9 43 5D B5 0C 0B A5 7D 3B 84 05 7C B9 0B 5C DE
01B0h: 44 F9 77 E8 46 5F 79 FF 98 BA A0 F0 66 95 7E 8D
01C0h: EF 7F 5F 53 4E 4E 6A 8E BE 77 21 6F BF 95 7E
01D0h: 2F 0C 89 16 F9 60 BA 57 EC 59 E1 C1 56 2B DD 8D
01E0h: E0 0C 9C 6C 4D 08 AD 24 D3 DF 40 B3 91 9F 35
01F0h: 02 2D 1E 09 3D F5 2D 8D E5 AE 5D 9E E7 4A 85 96
0200h: 00 8F FC 03 D2 E4 03 80 97 A4 55 24 F0 61
0210h: 90 71 E2 03 D8 48 70 82 71 DD 19 55 74 43 6F 6E
0220h: 6E 65 63 74 69 6F 6E 22 7D 0B

```
- Assembly Code:** Shows the encryption loop logic. Key instructions include:
  - `mov rdx, ptr [rax]` - Load 10 bytes from the original file.
  - `xor ecx, ecx` - Initialize XOR register.
  - `mov rax, qword [new_one.exe+48B842470]` - Load encryption key.
  - `lea rcx, qword [804A01]` - Set loop counter.
  - `mov rax, qword [new_one.exe+48B8424A8000000]` - Load another key or constant.
  - `cmp rdx, rcx` - Compare loop counter.
  - `jmp new_one.exe+48B842470` - Loop back.
  - `add rsi, qword [new_one.exe+48B842480000000]` - Advance pointer.
  - `mov rdx, rsi` - Update loop counter.
  - `imul rsi, rcx` - Calculate offset.
  - `add rsi, qword [new_one.exe+48033593490E0]` - Add key to pointer.
  - `mov rdx, ptr [rsi]` - Load encrypted data.
  - `mov rax, qword [new_one.exe+48B8424A0000000]` - Load another key.
  - `mov rdx, rax` - Update loop counter.
  - `mov rax, qword [new_one.exe+48B8424A0000000]` - Load another key.
  - `mov rdx, rdi` - Update loop counter.
  - `mov rax, qword [new_one.exe+48B842480000000]` - Load another key.
  - `cmp rax, rsi` - Compare pointer.
  - `jmp new_one.exe+48B842470` - Loop back.
  - `mov rdx, rax` - Update loop counter.
  - `mov rax, rsi` - Update loop counter.
  - `mov rdx, qword [new_one.exe+48B842490000000]` - Load another key.
  - `mov rsi, qword [new_one.exe+48B842458]` - Load another key.
  - `mov r9, qword [new_one.exe+48B842490000000]` - Load another key.
  - `mov r10, qword [new_one.exe+48B842460]` - Load another key.
  - `mov r11, qword [new_one.exe+48B842458]` - Load another key.
  - `mov rax, qword [new_one.exe+48B8424A0000000]` - Load another key.
  - `mov rdx, ptr [rsi]` - Load encrypted data.
  - `mov rdx, qword [new_one.exe+482CF6FFFF]` - Load another key.
  - `mov rdi, rdx` - Update loop counter.
  - `mov rsi, qword [new_one.exe+48B842480000000]` - Load another key.
  - `mov rdx, rdx` - Update loop counter.
  - `mov rax, rdx` - Update loop counter.
  - `mov rdx, rax` - Update loop counter.
  - `test rdx, rdx` - Check if zero.
  - `jmp new_one.exe+48B842470` - Loop back.
  - `mov rsi, qword [new_one.exe+48B842480000000]` - Load another key.
  - `mov rdx, qword [new_one.exe+48E936FFFFFF]` - Load another key.
  - `jmp new_one.exe+48B842470` - Loop back.
  - `mov rdx, qword [new_one.exe+48B842480000000]` - Load another key.
  - `mov rdx, qword [new_one.exe+48B8508]` - Load another key.
- Memory Dump:** Shows the state of registers and memory. The `rsi` register is highlighted with the value `000000000000143D18`. The memory dump shows the original file content at address `0000000000000000` and the encrypted content at `0000000000000000`.



Figure 7 – Encryption routine and Original/Encrypted file content

In the next step, the malware renames the encrypted files with the “.bianlian” extension and replaces them with the original file using the *MoveFileExW()* API function, as shown below.

```

mov rsi, rsp
mov rcx, qword ptr ds:[rsi]
mov rdx, qword ptr ds:[rsi+8]
mov r8, qword ptr ds:[rsi+10]
mov r9, qword ptr ds:[rsi+18]
movq xmm0, rcx
movq xmm1, rdx
movq xmm2, r8
movq xmm3, r9
call rax
add rsp, 150
    
```

[rsi]:L"C:\\Users\\Admin\\Desktop\\new\_one.exe  
[rsi+8]:L"C:\\Users\\Admin\\Desktop\\new\_one.exe  
MoveFileExW

Figure 8 – MoveFileExW() API

Finally, the ransomware deletes itself using the following command line, leaving only the encrypted files and the ransom note on the victim’s machine.

- `cmd /c del C:\\Users\\Admin\\Desktop\\new_one.exe`

The below figure shows the BianLian ransomware encrypted files and ransom note text file after the successful infection of a victim’s machine.

Name	Type	Size
image1.py.bianlian	BIANLIAN File	20 KB
image2.py.bianlian	BIANLIAN File	49 KB
image3.py.bianlian	BIANLIAN File	4 KB
image4.py.bianlian	BIANLIAN File	26 KB
image5.py.bianlian	BIANLIAN File	44 KB
image6.py.c.bianlian	BIANLIAN File	39 KB
image7.py.bianlian	BIANLIAN File	4 KB
image8.py.bianlian	BIANLIAN File	4 KB
image9.py.bianlian	BIANLIAN File	3 KB
image10.py.bianlian	BIANLIAN File	3 KB
image11.py.bianlian	BIANLIAN File	5 KB
image12.py.c.bianlian	BIANLIAN File	4 KB
image13.py.o.bianlian	BIANLIAN File	4 KB
image14.py.bianlian	BIANLIAN File	103 KB
image15.py.c.bianlian	BIANLIAN File	56 KB
image16.py.bianlian	BIANLIAN File	56 KB
Look at this instruction.txt	Text Document	1 KB
image17.py.bianlian	BIANLIAN File	7 KB
image18.py.bianlian	BIANLIAN File	3 KB

Figure 9 – Files encrypted by BianLian Ransomware

In the dropped ransom note, victims are given instructions on how they can contact the TAs to restore their encrypted files.

The TAs threaten their victims, stating that their important data, such as financial, client, business, technical, and personal files, has been downloaded and will be posted on their leak site if the ransom is not paid within ten days.

The ransom note also contains the ID of TOX Messenger for ransom negotiations and the Onion URL of the leak site page – shown in the figure below.

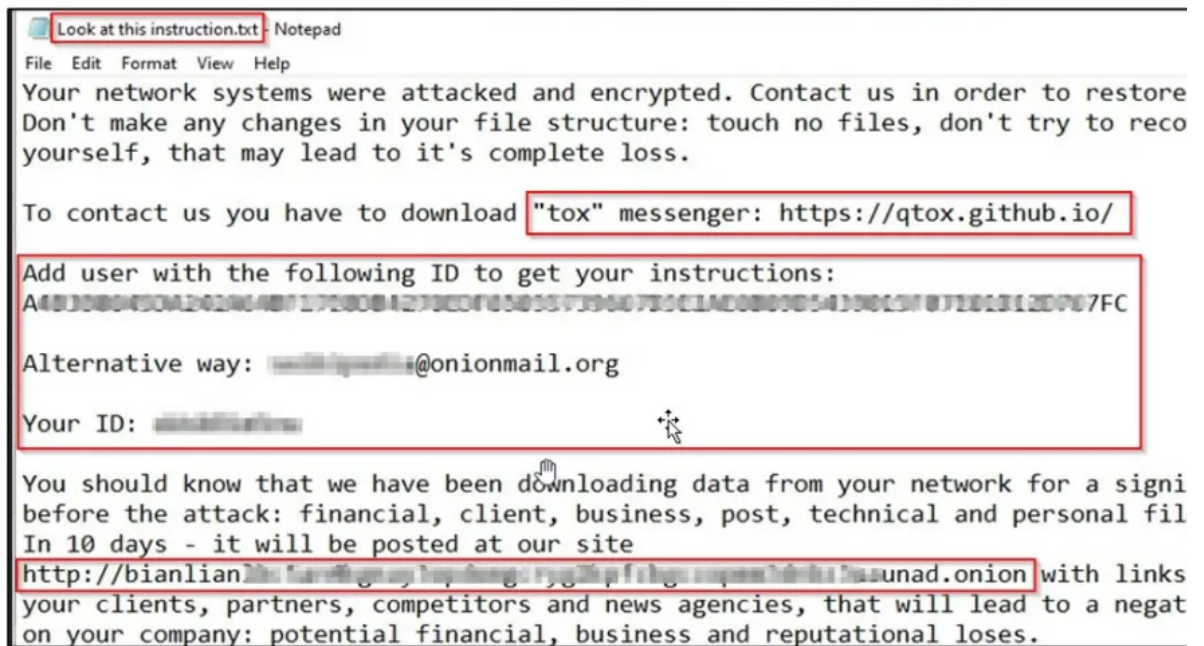


Figure 10 – Ransom note

The figure below shows the BianLian ransomware Onion leak home page and the affected company's extortion objects.

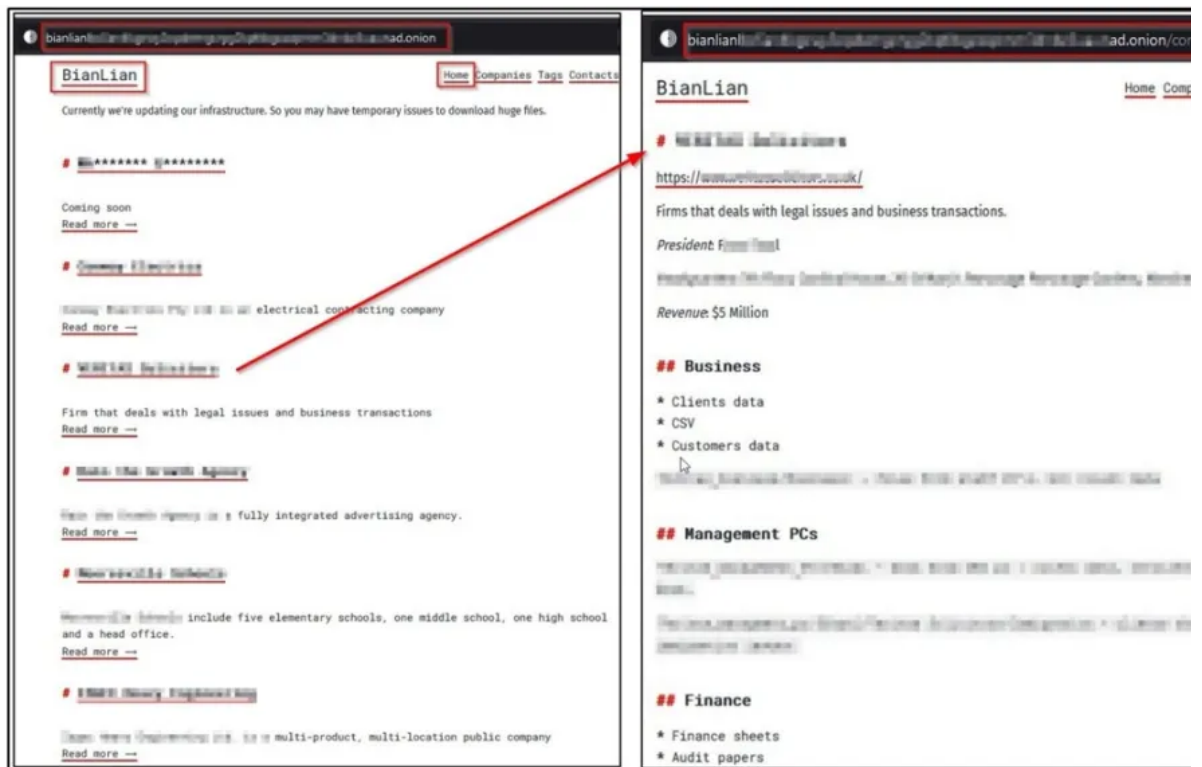


Figure 11 – BianLian Leak site home page

The BianLian Leak site contains the list of all companies affected by the ransomware and the TA's contact details for ransomware data recovery.

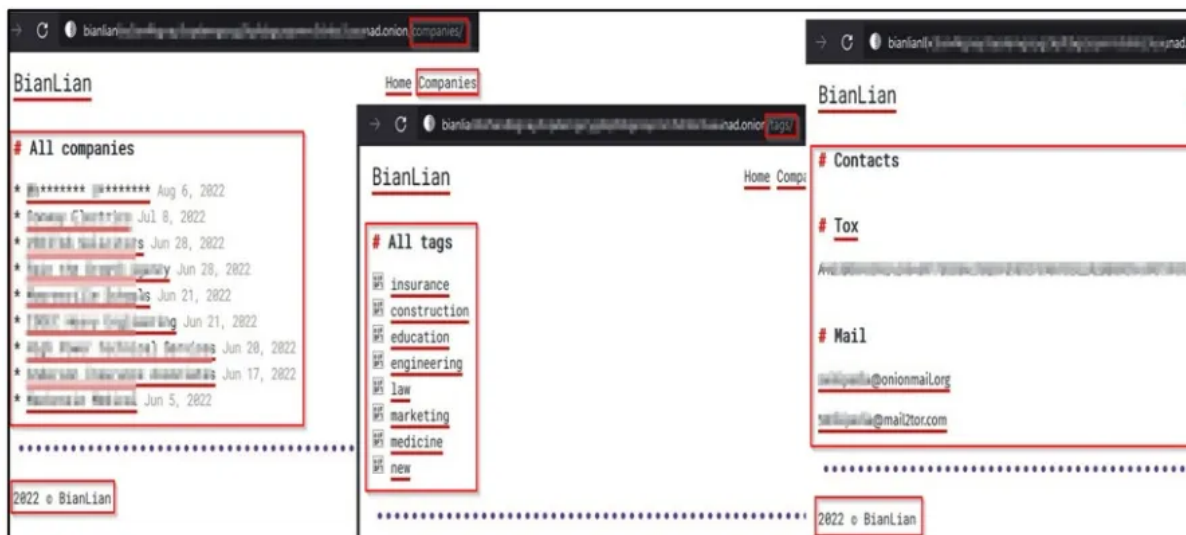


Figure 12 – BianLian Leak site affected companies list & TAs contact details

## Conclusion

Ransomware is becoming an increasingly common and effective attack method that affects organizations and their productivity. BianLian is GoLang-based ransomware that continues to breach several industries and demand large ransom amounts. The TAs also use the double extortion method by stealing an affected organization's files and leaking them online if the ransom is not paid on time.

TAs write their ransomware in GoLang for various reasons; the language enables a single codebase to be compiled into all major operating systems. The TAs behind BianLian are constantly making changes and adding new capabilities to avoid detection.

Cyber Research Labs will continue to monitor BianLian and other similar Ransomware groups' activities and analyze them to better understand their motivations.

## Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the best practices given below:

### Safety Measures Needed to Prevent Ransomware Attacks

- Conduct regular backup practices and keep those backups offline or in a separate network.
- Turn on the automatic software update feature on your computer, mobile, and other connected devices wherever possible and pragmatic.
- Use a reputed anti-virus and Internet security software package on your connected devices, including PC, laptop, and mobile.
- Refrain from opening untrusted links and email attachments without verifying their authenticity.

### Users Should Take the Following Steps After the Ransomware Attack

- Detach infected devices on the same network.
- Disconnect external storage devices if connected.
- Inspect system logs for suspicious events.

### Impact of BianLian Ransomware

- Loss of Valuable data.
- Loss of the organization's reputation and integrity.
- Loss of the organization's sensitive business information.
- Disruption in organization operation.
- Financial loss.

## MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Execution	T1204	User Execution
	T1059	Command and Scripting Interpreter
	T1497	Virtualization/Sandbox Evasion
Defense Evasion	T1027	Software Packing
	T1036	Masquerading
Discovery	T1082	System Information Discovery

	T1083	File and Directory Discovery
	T1518	Security Software Discovery
	T1120	Peripheral Device Discovery
Impact	T1486	Data Encrypted for Impact
Lateral Movement	T1091	Replication Through Removable Media

## Indicator Of Compromise (IOCs)

Indicators	Indicator Type	Description
0c756fc8f34e409650cd910b5e2a3f00	MD5	BianLian
70d1d11e3b295ec6280ab33e7b129c17f40a6d2f	SHA1	Ransomware
eaf5e26c5e73f3db82cd07ea45e4d244ccb3ec3397ab5263a1a74add7bbcb6e2	Sha256	Executable
08e76dd242e64bb31aec09db8464b28f	MD5	BianLian
3f3f62c33030cfd64dba2d4ecb1634a9042ba292	SHA1	Ransomware
1fd07b8d1728e416f897bef4f1471126f9b18ef108eb952f4b75050da22e8e43	Sha256	Executable