Technical Analysis of RiseLoader | ThreatLabz

By ThreatLabz

Published: 2024-12-16 · Archived: 2026-04-05 22:01:58 UTC

The following sections describe some of the features in RiseLoader.

Anti-analysis techniques

Most of the RiseLoader samples analyzed by ThreaLabz are packed with VMProtect. In addition, the malware obfuscates important strings. For example, all RiseLoader samples included the following strings related to malware analysis and debugging:

- `ollydbg.exe`
- `processhacker.exe`
- `tcpview.exe`
- `filemon.exe`
- `procmon.exe`
- `regmon.exe`
- `procexp.exe`
- `ida.exe`
- `ida64.exe`
- `binaryninja.exe`
- `immunitydebugger.exe`
- `wireshark.exe`
- `dumpcap.exe`
- `hookexplorer.exe`
- `importrec.exe`
- `petools.exe`
- `lordpe.exe`
- `sysinspector.exe`
- `proc_analyzer.exe`
- `sysanalyzer.exe`
- `sniff_hit.exe`
- `windbg.exe`
- `joeboxcontrol.exe`
- `joeboxserver.exe`
- `apimonitor.exe`
- `apimonitor-x86.exe`
- `apimonitor-x64.exe`
- `x32dbg.exe`

- `x64dbg.exe`
- `x96dbg.exe`
- `cheatengine.exe`
- `scylla.exe`
- `charles.exe`
- `cheatengine-x86_64.exe`
- `reclass.net.exe`

These strings are defined in a global array, but are not used during execution. This may indicate that anti-analysis features are currently in development and will potentially be implemented in future versions.

Note that RiseLoader does not currently use stack-based string obfuscation, which is present in RisePro and PrivateLoader.

Behavioral analysis

The malware starts by creating a mutex using hardcoded strings for the name. The mutex name will be a combination of three strings such as: `winrar8PROMEMEKGAmaV3_2_8`. The mutex is formed from a prefix (`winrar8`), a `campaign_id` value (`PROMEMEKG`), and a hardcoded suffix (`AmaV3_2_8`). If the mutex exists, RiseLoader will terminate. Samples analyzed by ThreatLabz have lacked a persistence mechanism, although this may be a configurable parameter (similar to other malware loaders).

Next, RiseLoader randomly selects a C2 server from a hardcoded list and opens a TCP connection. This process is repeated up to 10 times until a connection is established. If unsuccessful, RiseLoader terminates. Upon successful communication with the C2 server, a new thread is launched to continuously check for commands, process them, and send system information as requested. Additionally, another thread handles the `PAYLOADS` data from the C2 server, creating a randomly generated folder in the user's temporary directory to process each payload. This thread also creates an infection marker by creating a registry key under certain conditions and prepares the arguments and delays for each payload.

Finally, a new thread is created to download and execute each payload from URLs provided by the C2 server using libcurl. DLL files are launched with `rundll32`, while executables are started by creating a new process. After all payloads are downloaded and executed, RiseLoader terminates.

Network communication

After establishing the TCP three-way handshake with the C2 server, RiseLoader expects the server to respond with a message containing XOR keys used for subsequent communications. If the server does not send this message within a 10-second timeout, the malware will attempt to "wake up" the server by sending a `KEEPALIVE` message. If the server is online, it will respond with a `KEEPALIVE_RES` message, and the malware will reset its timeout. If the server does not respond, the malware will either attempt to reconnect or close the connection, and call `ExitProcess` after 10 failed attempts.

After receiving the XOR keys, the malware sends a `campaign_id` and other information to the server, then waits for the PAYLOADS command. The server can close the connection at any time without notifying the client.

Additionally, a `SEND_SHUTDOWN` command will immediately terminate the malware. The server periodically sends `KEEPALIVE` messages to ensure continuous communication. If the `PAYLOADS` command is received, RiseLoader processes the packet and sends either an `SL_TASKS_EXECUTED` or `PL_TASKS_EXECUTED` message with the task information. Once the task commands are received, the server closes the connection. The message types exchanged in both directions share a common structure, as defined below:

```
 struct message {
   uint32_t magic_bytes;  // Hardcoded to 0x00020001
   uint32_t data_size;
   uint32_t message_type;
   byte data[data_size];
}
```

The `magic_bytes` field may represent a protocol version (i.e., version 1.0.2.0 in little endian byte format), although it is too early to determine the value's exact meaning since this malware family is new.

Not all messages contain data; for these, the data_size will be zero. For messages that contain data, the structure varies. Some messages use a UTF-8 encoded JSON string, while others, like the `SET_XORKEYS` and `SEND_ID` message types, use a byte structure.

Throughout the communication process, the data field will be encoded using one of the XOR keys defined by the C2 server in the `SET_XORKEYS` message.

The RiseLoader message types are shown in the following table:

Message Type	**Message Value**	**Description and Payload**	**Source**
`SEND_VICTIM_INFO`	`0x2B`	Sends information related to cryptocurrency websites, wallets, and web browser extensions.	Sent by the client.
`SYS_INFO`	`0x2F`	Sends information related to the victim's machine in a JSON format: • `ap` : Unused. • `bn` : Windows build number. • `mi` : Minor version. • `mj` : Major version. • `c1` : Indicates if WoW64 process is present. • `tp` : Indicates if this is a workstation.	Sent by the client.

Message Type	Message Value	Description and Payload	Source
`SEND_ID_NEW_VICTIM`	`0x16C64`	Sends the `campaign_id` after checking that there is no former infection.	Sent by the client.
`SEND_ID`	`0x16C63`	Sends the `campaign_id` if RiseLoader previously executed payloads on the victim's system.	Sent by the client.
`SL_FL_TASKS_EXECUTED`	`0x23E9`	Sends a list of task IDs, which were downloaded and executed successfully. This message type is used only for the payload URLs that were included in the JSON keys `sl` and `fl`.	Sent by the client.
`PL_TASKS_EXECUTED`	`0x0b2f09`	Sends a list of task IDs, which were downloaded and executed successfully. This message type is used for the payload URLs that were included in the JSON key `pl`.	Sent by the client.
`SET_XORKEYS`	`0xB6`	The C2 server provides encryption keys that are used for subsequent messages. The first byte encodes message payloads from the infected system and the second byte decodes message payloads coming from the C2 server.	Sent by the server.
`CHANGE_ID`	`0x20C9`	The C2 server sends a new `campaign_id` to the victim.	Sent by the server.
`SEND_SHUTDOWN`	`0x1CCD`	Terminates execution.	Sent by the server.

Message Type	Message Value	Description and Payload	Source
`FORCE_REPORT_SL_FL`	`0x9C04`	Forces sending a `SL_FL_TASKS_EXECUTED` command while executing `pl` tasks.	Sent by the server.
`PAYLOADS`	`0x4DCF`	The C2 server sends a structure with several payloads to download and execute on the victim's system in a JSON format. There are three different arrays for payloads: `pl`, `fl`, and `sl`, each containing a similar structure where: • `pl`: Task ID. • `u`: URL of the payload. • `ag`: The arguments when launching the payload. • `d`: The delay of the download. • `f`: The filename payload on the victim's system. In addition, the `so` parameter will instruct the sample to set an infection marker by creating a registry key, while `sp` will specify the delay in milliseconds between the execution of `fl` and `sl` payloads. The final parameter, `lo`, will contain a URL of an image that is potentially used for tracking purposes.	Sent by the server.
`KEEPALIVE`	`0x6B5`	Requests a response from `KEEPALIVE_RES`.	Sent by both the client and the server.

Message Type	Message Value	Description and Payload	Source
`KEEPALIVE_RES`	`0x4B7C`	The response to the `KEEPALIVE` message.	Sent by both the client and the server.

Table 1: RiseLoader message types exchanged between the client and the server.

The figure below shows a high-level view of RiseLoader’s network communication protocol.

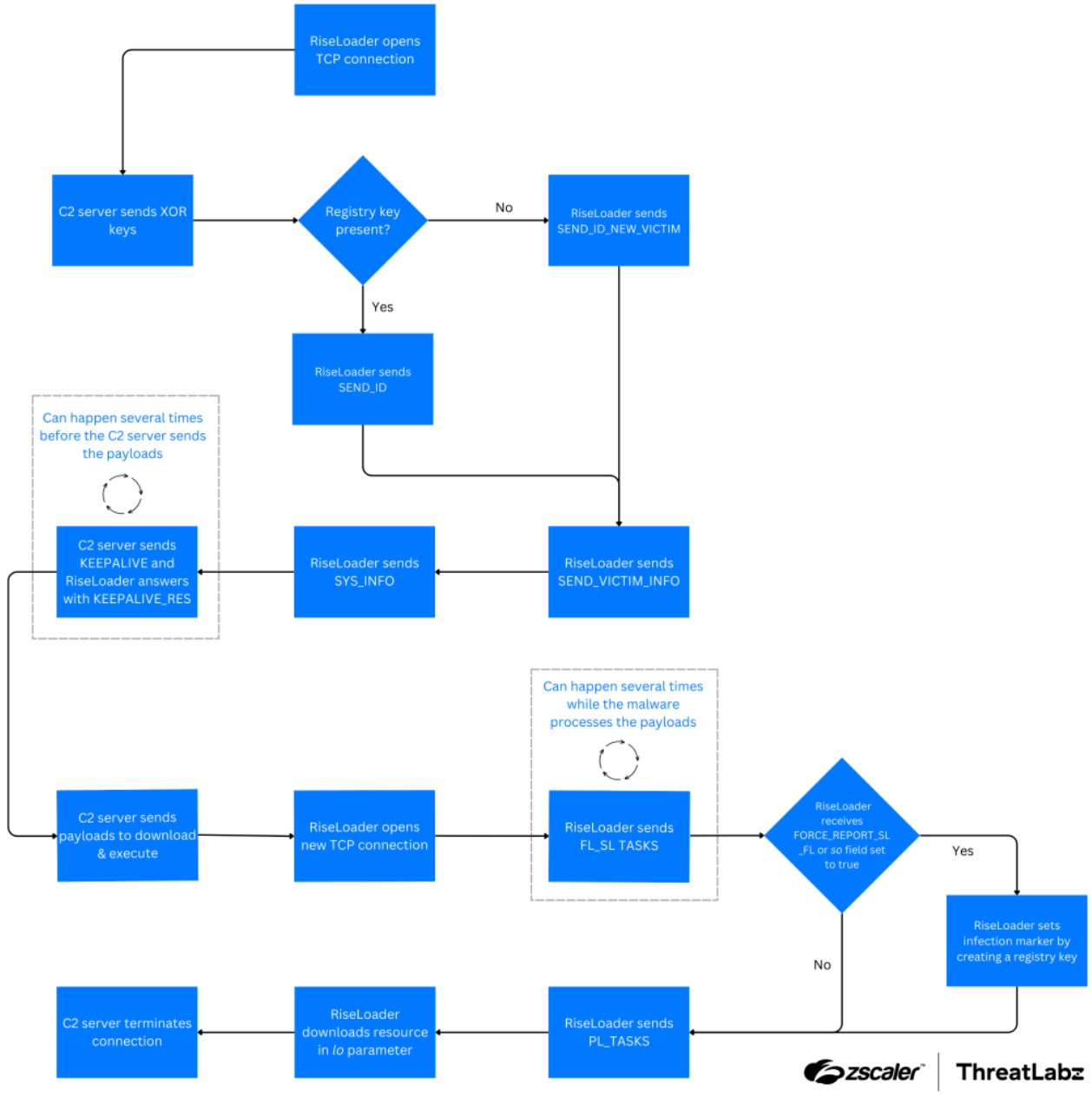

Figure 1: RiseLoader network communication protocol.

Network communication starts with the server sending a `SET_XORKEYS` message that contains two 1-byte XOR keys: the first for encrypting messages from the client, and the second for decrypting messages received from the server. After establishing the encryption keys for the session, the client sends either a `SEND_ID_NEW_VICTIM` or `SEND_ID` message. RiseLoader will determine which of these messages to use based on the existence of a specific registry key (`HKEY_CURRENT_USER\SOFTWARE\dmdsaodgmarksmdkgsa`).

Immediately after sending the `campaign_id` to the C2 server, RiseLoader will scan the victim's file system to gather information about cryptocurrency wallets, extensions, and specific programs (shown in the Appendix). This information will be sent to the server using the `SEND_VICTIM_INFO` message type, followed by a `SYS_INFO` packet. Once the system information is sent, the server will maintain the connection by exchanging `KEEPALIVE` and `KEEPALIVE_RES` messages with RiseLoader.

RiseLoader waits for a `PAYLOADS` command containing a JSON encoded structure with payload URLs to download and execute on the victim's system. The malware will then send `SL_FL_TASKS_EXECUTED` and `PL_TASKS_EXECUTED` messages to report the tasks that were executed.

During the processing phase for payloads, RiseLoader may create a registry key depending on the value specified by the `so` field in the `PAYLOADS` data structure or when receiving the `FORCE_REPORT_SL_FL` command from the C2 server. This registry key appears to serve as an infection marker and is located at `HKEY_CURRENT_USER\SOFTWARE\dmdsaodgmarksmdkgsa`. Under this key the registry name `var1` is created with a hardcoded value set to `0x00b2`. The actual value is not relevant for execution, as RiseLoader only checks for the presence of the registry key when choosing to send either `SEND_ID` or `SEND_ID_NEW_VICTIM` messages.

After processing all payloads, RiseLoader downloads a resource from a URL specified in the `lo` parameter of the `PAYLOADS` structure. Currently, this URL resolves to a 1x1 pixel PNG file, likely serving as a tracking method since the PNG file has no clear purpose in the malware code. After downloading and executing the payloads and downloading the URL from the `lo` parameter, RiseLoader will terminate its execution.

Comparison of the RiseLoader and RisePro communication protocols

The similarities between RiseLoader and RisePro are described below:

- Both use a custom binary TCP-based protocol with encoded JSON messages encrypted by a single byte key.
- The message structure is very similar: magic bytes, followed by a 4-byte payload size, and a 4-byte command type as shown in the figure below.

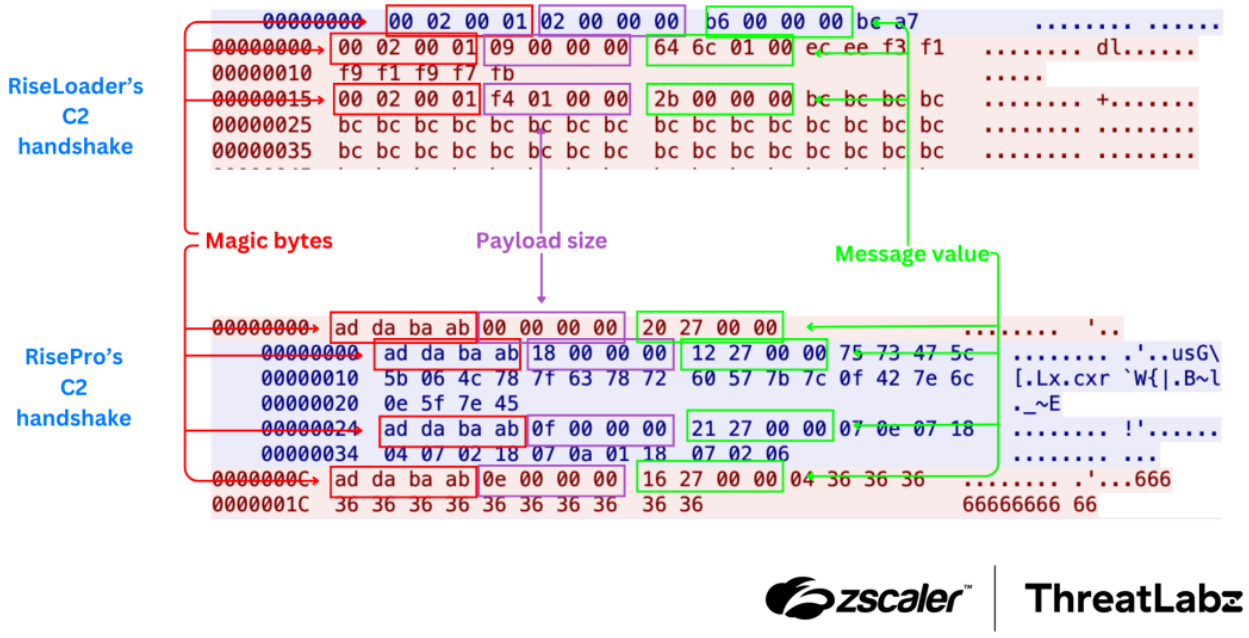

Figure 2: A comparison of RiseLoader's C2 handshake and RisePro's handshake, showing a similar structure.

- The initialization process is similar; however, RiseLoader has been simplified.
