
Tranalyzer2

wechatDecode



WeChat



Tranalyzer Development Team

Contents

1	wechatDecode	1
1.1	Description	1
1.2	Dependencies	1
1.3	Configuration Flags	1
1.4	Flow File Output	1
1.5	Custom File Output	2

1 wechatDecode

1.1 Description

The wechatDecode plugin detects and decodes JCE encoded data found in observed HTTP traffic and writes the results as JSON into a file.

To identify relevant packets to decode, the HTTP payload is searched for the string "TMA SDK_". If this string is found, the HTTP payload is processed using a JCE decoder.

Decryption and decompression of the body are handled automatically.

1.2 Dependencies

1.2.1 External Libraries

This plugin depends on the **zlib** library.

Ubuntu:	sudo apt-get install	zlib1g-dev
Arch:	sudo pacman -S	zlib
Gentoo:	sudo emerge	zlib
openSUSE:	sudo zypper install	zlib-devel
Red Hat/Fedora¹:	sudo dnf install	zlib-devel
macOS²:	brew install	zlib

1.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
WECHAT_JSON_SUFFIX	"_wechat.json"	Suffix appended to the base output file name
WECHAT_JSON_ARRAY	0	0: Output a single JSON array 1: Output a line-delimited JSON
WECHAT_INITIAL_JSON_BUFFER_SIZE	2048	Initial size of output buffer (increased dynamically)
WECHAT_MAX_HTTP_HDR_FIELD_LEN	1024	Maximum length of a HTTP header field
WECHAT_MAX_QUA_MATCH_LEN	255	Max. length of a matching group in the QUA string
WECHAT_VERBOSITY_LEVEL	0	Verbosity level: 0: Quiet mode (only warnings and errors) 1: Debug mode

1.4 Flow File Output

This plugin does not output any columns in the flow file. It writes the decoded data directly to the JSON output file.

¹If the `dnf` command could not be found, try with `yum` instead

²Brew is a packet manager for macOS that can be found here: <https://brew.sh>

1.5 Custom File Output

The wechatDecode plugin produces JSON output in the file `PREFIX_wechat.json`, where `PREFIX` is provided via the `Tranalyzer` option `-w` or `-W` and the `_wechat.json` suffix can be overridden via the configuration flag `WECHAT_JSON_SUFFIX`.

1.5.1 JSON format

For each packet detected to contain JCE data, the HTTP payload is decoded, decrypted and, if necessary, decompressed. The extracted data is then written as one JSON object per packet.

Each JSON object is divided into three sections, i.e., keys: `flow`, `requestHeader` and `body`. The `flow` section contains the six-tuple and the `firstSeen` timestamp of the flow. The `requestHeader` contains any information extracted from the `reqHead` part of the JCE payload, including any optional fields. The `body` section contains the decrypted and decoded information found in the `body` part of the decoded HTTP payload.

Numerical values in the `requestHeader` extracted from the `reqHead` are printed as quoted JSON strings as there are no static guarantees on the bounds of the matched numbers. This avoids integer overflows and unexpected incorrect results.

The following currently understood body types are supported:

- `ReportLog`
- `GetSettings`
- `GetConfig`
- `StatReport`

If the configuration flag `WECHAT_JSON_ARRAY` is set to 0, the JSON objects are output as line-delimited JSON, i.e. one JSON object per line. Note that in this mode, a JSON object may not contain a newline character as it is already used as a delimiter between individual JSON objects.

If `WECHAT_JSON_ARRAY` is 1, the JSON objects are enclosed in a JSON array and separated with a comma to yield valid JSON output. The line-delimited JSON mode is more robust as it does not rely on proper termination of the application to write the closing square bracket.