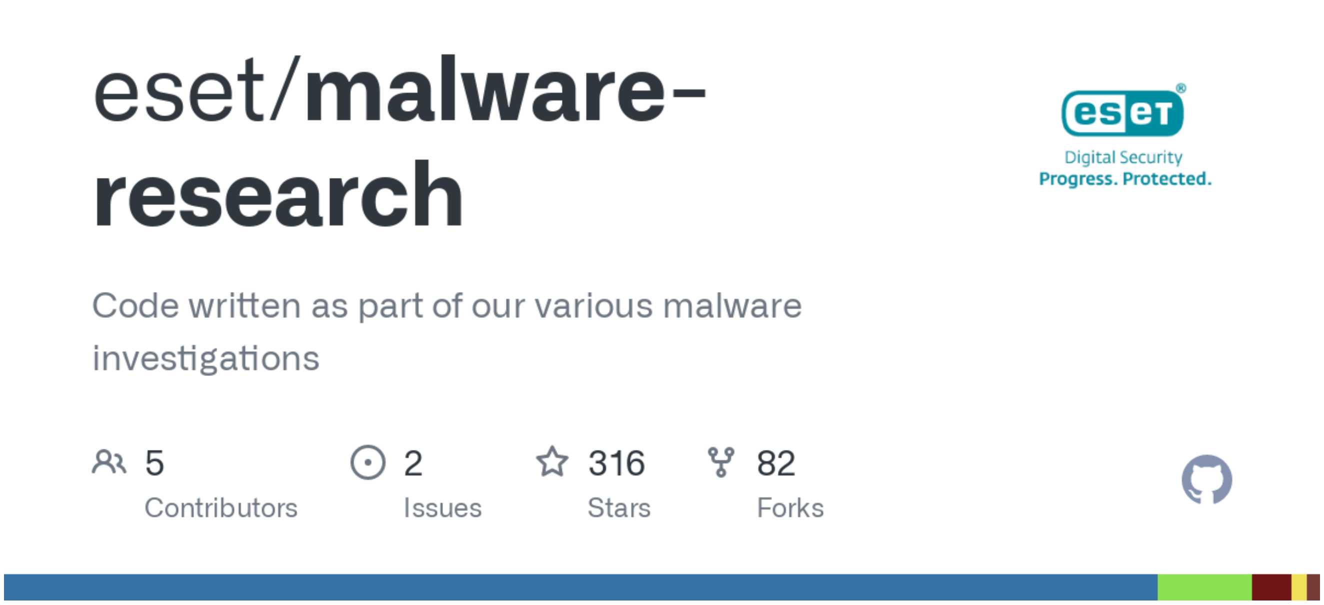
Collection of helper scripts for OceanLotus

github.com/eset/malware-research/tree/master/oceanlotus

eset



eset/malware-research

Code written as part of our various malware investigations

5 Contributors 2 Issues 316 Stars 82 Forks

This repository contains scripts to help analysing OceanLotus' latest campaign using the legitimate "rastls.exe" application for side-loading.

As described in ESET research whitepaper there are two components that has encrypted payloads: the fake document (dropper component) and the backdoor (`rastls.dll`).

`ol_unpack_shellcode_from_decoy.py` will unpack the shellcode embedded in the resource of the (fake) decoy document.

`ol_unpack_files_from_3rd_stage.py` is almost the same script but it is used for the third stage of the dropper part, obtained after the emulation (using `shellcode_emulator`) of shellcode outputted by `ol_unpack_shellcode_from_decoy.py` . The script extracts the encrypted and compressed configuration and parse it. It prints the possible install paths for the backdoor and its persistence mechanism. Finally, the script drops all the backdoor components (e.g. `rastls.exe` , `rastls.dll` and `SyLog.bin`). The Kaitai Struct structure `ol_decoy_dropped_files.ksy` was used to create the Python class.

Both of these scripts uses lief as a PE parser so make sure to install it beforehand. `ol_unpack_files_from_3rd_stage.py` uses Kaitai Struct.

`ol_unpack_shellcode_from_backdoor.py` decrypts the shellcode of an installed backdoor using the key and IV embedded in the `rastls.dll` file and the encrypted `OUTLFLTR.DAT` file (or `SyLog.bin` depending on the version).

The folder `shellcode_emulator` contains a script and its description to run the shellcode emulator. Since the same shellcode is used everywhere during this campaign, it was faster to emulate it instead of using dynamic analysis.

The following flow could be used to obtain the dropped files from the decoy document (the dropper):

`ol_unpack_shellcode_from_decoy.py` ⇒ `ol_shellcode_emulator.py` ⇒ `ol_unpack_files_from_3rd_stage.py`

The following flow could be used to obtain the third stage of the backdoor component:

`ol_unpack_shellcode_from_backdoor.py` ⇒ `ol_shellcode_emulator.py`

Finally, this repository also contains the Kaitai structure for the fifth stage of the backdoor component. It will parse the configuration structure of the decrypted resource. In order to generate a parser class or visualize it, Kaitai Struct should be installed.