Deobfuscating and hunting for OSTAP, Trickbot’s dropper and best friend

By Equipe CERT

Published: 2020-04-14 · Archived: 2026-04-05 22:57:02 UTC

[et_pb_section fb_built=”1″ admin_label=”section” _builder_version=”3.22″][et_pb_row admin_label=”row” _builder_version=”3.25″ background_size=”initial” background_position=”top_left” background_repeat=”repeat” custom_margin=”|auto|-2px|auto||”][et_pb_column type=”4_4″ _builder_version=”3.25″ custom_padding=”|||” custom_padding__hover=”|||”][et_pb_text admin_label=”Text” _builder_version=”4.4.2″ text_font=”Nunito Sans|||||||” text_font_size=”16px” header_2_font=”Poppins|||||||” header_2_text_color=”#071793″ header_2_font_size=”31px” background_size=”initial” background_position=”top_left” background_repeat=”repeat” custom_padding=”||0px|||”]

During **a recent investigation dealing with ransomware attack**, CERT Intrinsec faced OSTAP loader. This loader is **used to deliver other malwares** (such as Trickbot) on an infected system. It uses high obfuscation techniques to prevent the code from being read and to bypass detection processes.

Obfuscated loader

[/et_pb_text][et_pb_image src="https://www.intrinsec.com/wp-content/uploads/2020/04/fig1-1.png" _builder_version="4.4.2"][/et_pb_image][et_pb_text _builder_version="4.4.2"]

Figure 1 : Extract of the loader code (Javascript)

[/et_pb_text][et_pb_text admin_label=”Text” _builder_version=”4.4.2″ text_font=”Nunito Sans|||||||” text_font_size=”16px” header_2_font=”Poppins|||||||” header_2_text_color=”#071793″ header_2_font_size=”31px” background_size=”initial” background_position=”top_left” background_repeat=”repeat”]

The ostap loader (Figure 1) we analysed was about 10 000 lines long. We started the **static analysis** by going through the code by hand to understand its structure. We identified a key part of the code that helped us to deobfuscate the loader (Figure 2).

[/et_pb_text][et_pb_image src="https://www.intrinsec.com/wp-content/uploads/2020/04/fig2-1.png" _builder_version="4.4.2"][/et_pb_image][et_pb_text _builder_version="4.4.2"]

Figure 2 : Key part of the loader code

[/et_pb_text][et_pb_text admin_label=”Text” _builder_version=”4.4.2″ text_font=”Nunito Sans|||||||” text_font_size=”16px” header_2_font=”Poppins|||||||” header_2_text_color=”#071793″ header_2_font_size=”31px” background_size=”initial” background_position=”top_left” background_repeat=”repeat”]

The instructions, on *Figure 2*, aim at executing *String.fromCharCode* function **ewnfBeth8** parameter. There are lots of noise instructions in the program. For example, **ppfhair_3(ewnfBeth8)** instruction in *try* statement will never be triggered because the function does not exist. It is done on purpose to always enter the *catch*. Besides, **etvulike2** parameter is always equal to **'f'**. A large part of the program consists of a concatenation of functions such as the one shown on *Figure 3*.

[/et_pb_text][et_pb_image src="https://www.intrinsec.com/wp-content/uploads/2020/04/fig3-1.png" _builder_version="4.4.2"][/et_pb_image][et_pb_text _builder_version="4.4.2"]

Figure 3 : Repeated function model

[/et_pb_text][et_pb_text admin_label="Text" _builder_version="4.4.2″ text_font="Nunito Sans|||||||" text_font_size="16px" header_2_font="Poppins|||||||" header_2_text_color="#071793″ header_2_font_size="31px" background_size="initial" background_position="top_left" background_repeat="repeat" custom_padding="|||1px||"]

The action of the function above (*Figure 3*) is to apply *String.fromCharCode* to 69, i.e. « E ». **The program uses this method to set all its instructions**. Knowing that, we decided to write a script to extract each obfuscated character.

[/et_pb_text][et_pb_text admin_label="Text" _builder_version="4.4.2″ text_font="Nunito Sans|||||||" text_font_size="16px" header_2_font="Poppins|||||||" header_2_text_color="#071793″ header_2_font_size="31px" background_size="initial" background_position="top_left" background_repeat="repeat" custom_padding="|||1px||"]

Deobfuscation script

The main goal of the script is **to get indicators of compromise from the loader.** It has been developed using Node JS. It first goes through the obfuscated loader, retrieves the targeted numbers and apply *String.fromCharCode* to decode them. Then, it collects the indicators of compromise in the decoded payload using regular expressions. Extracted IOCs are IPs, URLs and User-Agents. The figure below represents the output of the script using a sample hunted on VirusTotal. We can see, at the top of *Figure 4*, a list of file extensions that are targeted (their content will be replaced by the Ostap JS code).

[/et_pb_text][et_pb_image src="https://www.intrinsec.com/wp-content/uploads/2020/04/fig4.png" _builder_version="4.4.2"][/et_pb_image][et_pb_text _builder_version="4.4.2"]

Figure 4 : Script execution output

[/et_pb_text][et_pb_text admin_label="Text" _builder_version="4.4.2″ text_font="Nunito Sans|||||||" text_font_size="16px" header_2_font="Poppins|||||||" header_2_text_color="#071793″ header_2_font_size="31px" background_size="initial" background_position="top_left" background_repeat="repeat"]

Hunting

After deobfuscating the loader as a part of our investigation, **we decided to hunt recent and similar files on VirusTotal, using searches on static code patterns** (*content: "'String')['slice']"*, for instance). We found lots of samples (*Figure 5*) and process them so as to extract as many IOCs as possible.

[/et_pb_text][et_pb_image src="https://www.intrinsec.com/wp-content/uploads/2020/04/fig5-1.png" _builder_version="4.4.2"][/et_pb_image][et_pb_text _builder_version="4.4.2"]

Figure 5 : Ostap samples from VirusTotal

[/et_pb_text][et_pb_text admin_label="Text" _builder_version="4.4.2″ text_font="Nunito Sans|||||||" text_font_size="16px" header_2_font="Poppins|||||||" header_2_text_color="#071793″ header_2_font_size="31px" background_size="initial" background_position="top_left" background_repeat="repeat"]

We collected about **140 samples from VirusTotal using the script**. We analysed them and extracted the indicators of compromise presented in the table below. We can say that at least one of the IP addresses (185[.]234[.]73[.]125) **is related to the Trickbot campaign happening since the Coronavirus appeared such as in Italy, as reported by Sophos** (1).

[/et_pb_text][et_pb_text _builder_version="4.4.2″ hover_enabled="0″]

IP	URL	User-Agent
141[.]98[.]214[.]14	hxxps[://]141[.]98[.]214[.]14/6BcsTO/AGVV5r[.]php	Mozilla/5.0 (Windows NT 6.; Win64; x64; Trident/7.0; rv:11.0) like Gecko
185[.]159[.]82[.]205	hxxps[://]185[.]159[.]82[.]205/2/1[.]php	
185[.]216[.]35[.]10	hxxps[://]185[.]216[.]35[.]10/VYut68/L2KSUN[.]php	
185[.]234[.]73[.]125	hxxps[://]185[.]234[.]73[.]125/wMB03o/Wx9u79[.]php	
194[.]87[.]96[.]100	hxxps[://]194[.]87[.]96[.]100/2/1[.]php	
45[.]128[.]133[.]41	hxxp[://]45[.]128[.]133[.]41/jTlp8P/3OXkud[.]php	
91[.]196[.]70[.]126	hxxps[://]91[.]196[.]70[.]126/2/zsQX9M[.]php	

[/et_pb_text][et_pb_text admin_label="Text" _builder_version="4.4.2″ text_font="Nunito Sans|||||||" text_font_size="16px" header_2_font="Poppins|||||||" header_2_text_color="#071793″ header_2_font_size="31px" background_size="initial" background_position="top_left" background_repeat="repeat"]

References

1. https://news.sophos.com/en-us/2020/03/04/trickbot-campaign-targets-coronavirus-fears-in-italy/

2. https://www.esentire.com/blog/oh-snap-new-ostap-variant-observed-in-the-wild
3. https://www.bromium.com/deobfuscating-ostap-trickbots-javascript-downloader/
4. https://blog.trendmicro.com/trendlabs-security-intelligence/latest-trickbot-campaign-delivered-via-highly-obfuscated-js-file/
5. https://github.com/cryptogramfan/Malware-Analysis-Scripts
6. https://www.cert.pl/en/news/single/ostap-malware-analysis-backswap-dropper/
7. Link to the script on Intrinsec Github : https://github.com/Intrinsec/CERT/tree/master/Scripts/ostap_deobfuscator

[/et_pb_text][/et_pb_column][/et_pb_row][/et_pb_section][et_pb_section fb_built="1" _builder_version="4.4.2" use_background_color_gradient="on" background_color_gradient_start="#071793" background_color_gradient_end="rgba(7,23,147,0.59)" custom_margin="|103px||87px||"][et_pb_row column_structure="3_4,1_4" _builder_version="3.25"][et_pb_column type="3_4" _builder_version="3.25" custom_padding="|||" custom_padding__hover="|||"][et_pb_text _builder_version="4.4.2" text_font="Nunito Sans|||||||" text_text_color="#ffffff" text_font_size="24px" text_line_height="1.4em" header_font="|||||||" width="101.2%" custom_margin="-11px|-42px||31px||" custom_padding="13px||"]

Want to learn more about our Computer Emergency Response Team (CERT) ?

[/et_pb_text][/et_pb_column][et_pb_column type="1_4" _builder_version="3.25" custom_padding="|||" custom_padding__hover="|||"][et_pb_button button_url="https://www.intrinsec.com/cert-intrinsec" url_new_window="on" button_text="Discover" button_alignment="left" _builder_version="4.4.2" custom_button="on" button_text_size="18px" button_text_color="#e02b20" button_bg_color="#ffffff" button_border_color="#ffffff" button_font="Nunito Sans|700|||||||" button_icon="%%86%%" button_icon_color="#ffffff" button_icon_placement="left" custom_margin="14px|-30px|7px|-6px||"][/et_pb_button][/et_pb_column][/et_pb_row][/et_pb_section]

Source: https://www.intrinsec.com/deobfuscating-hunting-ostap/