Contents

Overview

Pro-active threat hunting can take from minute to hours, but with an automation tool such as Scrapper V1, we can ease the search of possible malicious files, IPs, domains, URLs. In this document, we will go through the acquirement of new IoC, how to maintain them, and how to properly filter them for utilization. This document will go through the utilization of the packaged executable [script ready to use].

1. Requirements

Roles:

I assume some sort of editor/admin at tools that allow IoC ingestion

2. Prerequisites

- Access to:
  - Script (Scrapper V1.exe)
  - Config.toml (same path as the script)
  - DataBase reader such as SQLite
  - SharePoint folder where the main DB is held

3. Understanding the Script

The script has a basic protocol to request information from public/private repositories that contain IoCs. Some of these repositories are API (OTX AlienVault, CrowdStrike), TAXII [RESTful APIs] (MS-ISAC, CISA); and a little more complex logic for websites/blogs (guidepointsecurity[.]com).

As of now, it only supports OTX AlienVault – but it has the TAXII skeleton built for implementation. MS-ISAC TAXII has been “deactivated” (waiting for reactivation), and CISA has been requested. Next step as these are getting unblock is to scrap websites.

Next stage is implementing VirusTotal to verify database reports and add a new value (VT score).

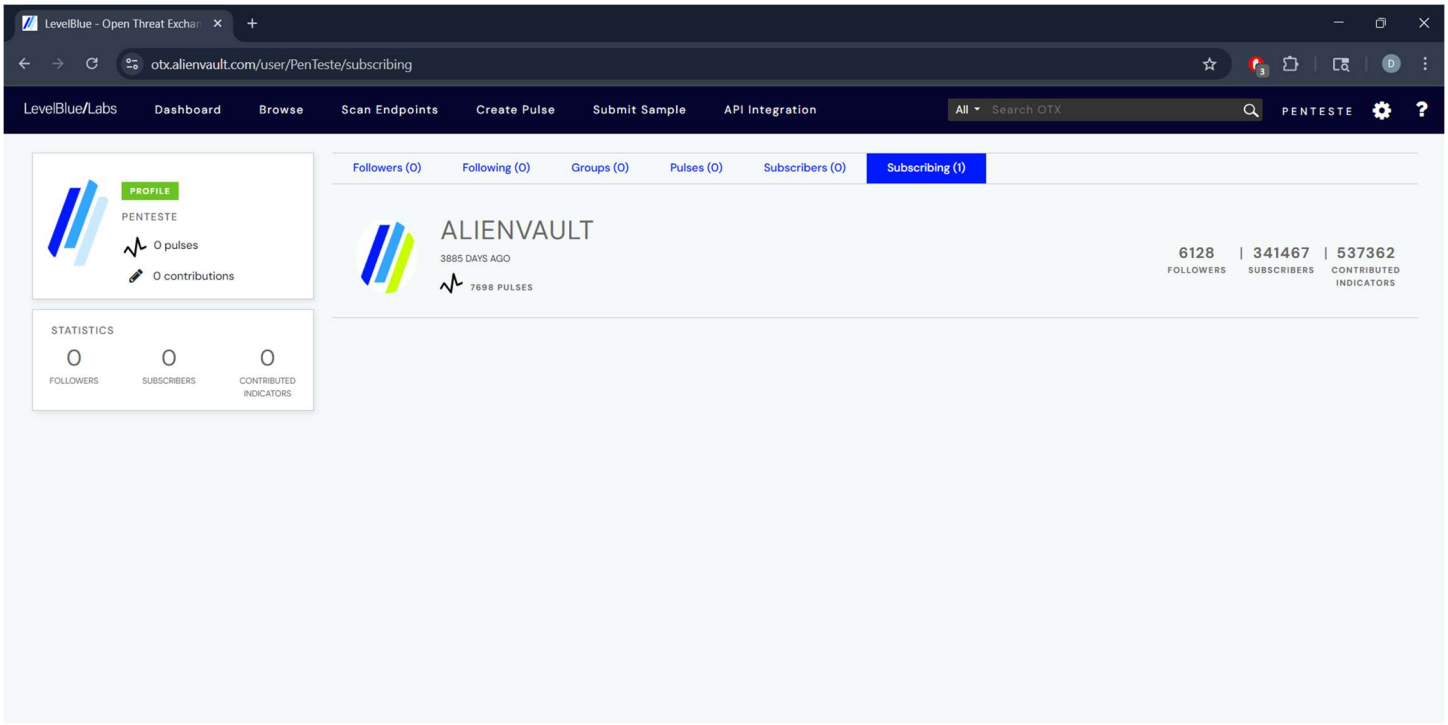
Following that, I would like to implement some GPT API to verify sources from the .db or even read sites that are not yet supported and/or don’t follow a pattern in their HTML.

4. OTX AlienVault

Open Threat eXchange (OTX) AlienVault is part of AT&T cybersecurity; and is a community-driven threat intelligence that collaborates with sharing bundle of IoCs for threat discovery. Their DirectConnect API allows to retrieve “pulses” from their API website, which can be tailored by the API user/holder. As of right now, the API is fetching from the AlienVault vetted/common page/collection – so their IoCs are most likely true positives, but some noise could be found. Thus, some filtering and/or manual revision are recommended.

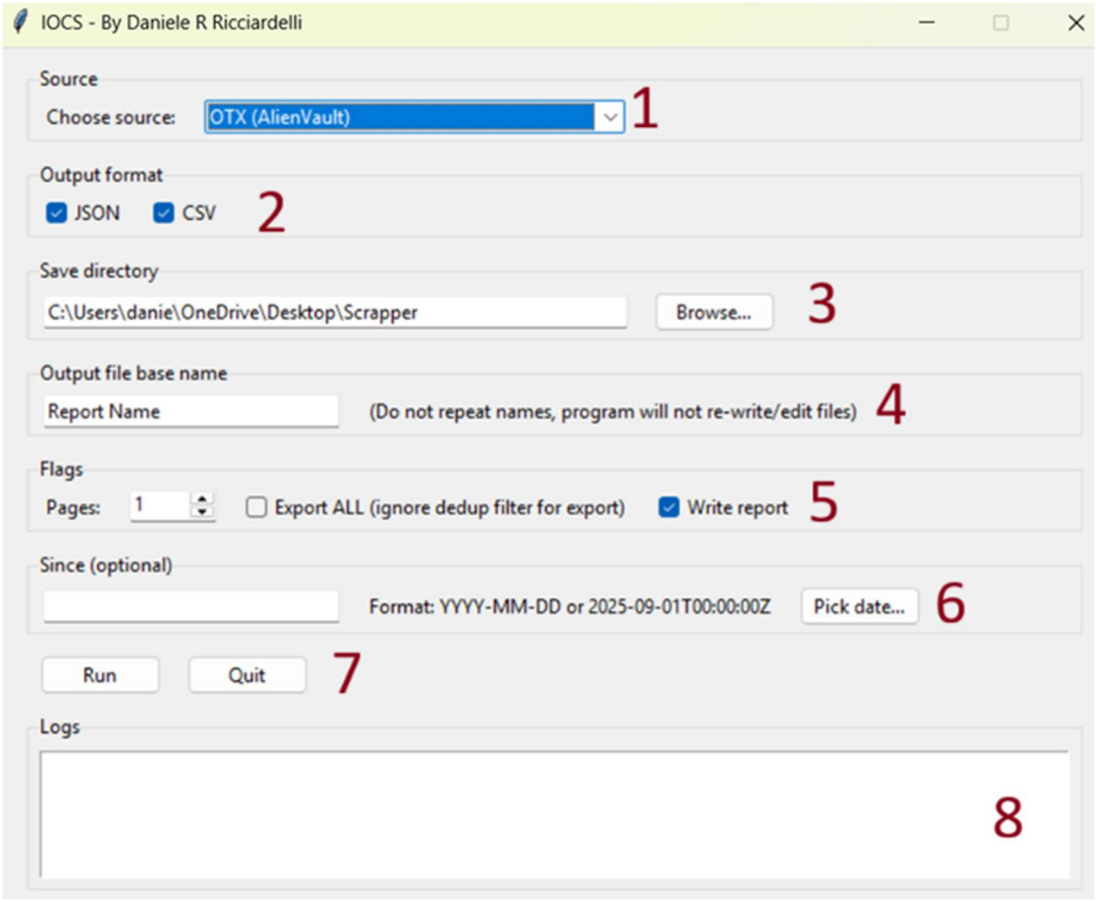
For reference, to access the API website, we can log into https://otx.alienvault.com/api

To check the “subscribed” feeds (collections), we can go to https://otx.alienvault.com/user/PenTeste/subscribing



5. Script usage

The script is quite simple, I will describe what every button/field does.



1. Source: For Scrapper V1, it only supports OTX AlienVault, so whether we want to change it or not, is not offered yet.

2. The output format will print the **NEW** IOCs added into the database. The ones that have been recorded already will not be overwritten nor reported in the JSON/CSV. I recommend ignoring the JSON (unless the user intends to use it to ingest it at some tool), and check the CSV for easy "copy-paste" of IOCS and quick filtering.

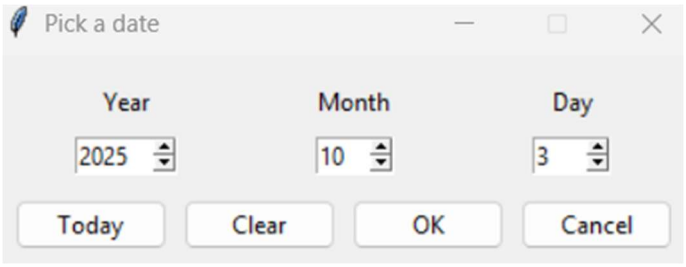
3. Save Directory is the path we decide to save the **reports**, not the database. The database will be stored in the same path as the script. I recommend having reports and the database separated to prevent confusion.

4. Here we choose the report name, personal recommendation is to name each report based on the date created. Moreover, prevent same name reports or will throw an error. The script doesn't have the capability to edit or overwrite files. *Example: Report 10/05/2025 – Swings – Daniele R.*

5. Flags will check the “pages” of collections. Most APIs have “pages” where IOCs or the bundle of IOCs are stored and later on shared. Picture it as when you request from the API/TAXI/Blog, it goes through the different tabs. Below picture for reference:

As it can be seen here, the flag of a google search is always by default at 1, same goes with our script.

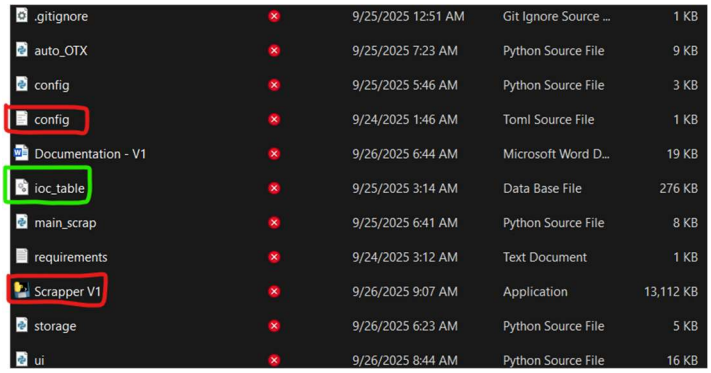
6. Here we can filter by date, format is shared in the script, but it can also be chosen manually through the button. Ideally this is ignored, unless we want to filter by a specific date. Screenshot below



7. Here we can start the script once we have chosen all our options and quit/break the script.
8. Here is the logs box, mainly to analyze possible errors and problems. Moreover, it will also shows us what is doing, but ultimately the messages that we care will be alerts such as “error: ___”, or fortunately, “X IOCS have been created”.

Note: Once the script is run for first time, it will create a database in the same path called “ioc_table”, and it will update with **WRITE** only. New IOCs will be added, previous ones will be ignored, and it will **NOT** edit them.

For reference, as mentioned, the script (.exe) and the configuration document (.toml) must be ran under the same directory to be recognize [files highlighted in red], and the database that will be read/compared will be under the same directory/path as well [highlighted in green]. The database (.db) will be created once the script is lunch for first time.



To idealize this script, we must operate it from a shared folder so the database we work with is updated constantly by the SOC team. As of right now, it is set to be created/modified from the same directory as the script, but as we progress, I will set it up by default on the shared folder of preference.

Last but not least, keep in mind that the ultimate way to verify this is through the created .csv report, but it is crucial to check if the database is being modified correctly as well – thus access to software (or free software) like SQLite is needed to open the database and inspect its content.

If you got this far and still have questions…