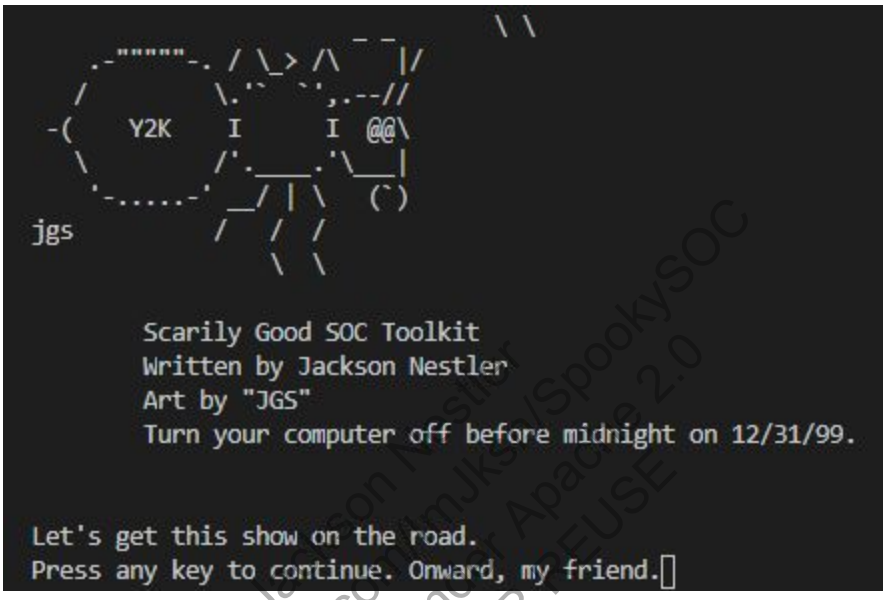
SpookySOC

A Scarily Good SOC Toolkit



Creator:

Jackson Nestler, Network Security and Technology Forensics

Last Updated: April 23, 2020

Technical Field

SpookySOC is heavily involved in the Network Security and Technology Forensics majors.

Background Information

SpookySOC has been in development for the past five years in varying forms and use-cases during that time. I became interested in cybersecurity during high school and spent time discussing with industry professionals "what makes your job fun?" and with that came the necessary followup of "what makes your job less fun?" A common thread was the necessity of performing Open Source Intelligence ("OSINT") on potential Indicators of Compromise ("IOCs"), a time-consuming process of the security analyst's job. This costs the analyst time, the organization resources, and adds a layer of obscurity to an attacker's deck of cards.

Prior Art

Searching on GitHub, considered *the* standard for open source code repositories, there are many security tools available. Most of these projects have one or more shortcomings that SpookySOC aims to improve upon. These shortcomings include, but are not limited to:

- Outdated programming or scripting languages (Python 2.x is popular, but reached End of Life on April 1, 2020).
- Lack of effective communication. End users must read pages and pages of documentation to utilize the tool.
- The "tool" is a "library" -- code that helps simplify the programming process for other developers, but is not an end-user tool.
- Development has ceased.
- Threat intelligence ("Threat intel") repositories utilized are outdated or provide incorrect information too frequently.
- Threat intel has too few sources to back up its conclusions.

The work that most closely resembles SpookySOC is a set of Python scripts developed by "TheresAFewConors" et al, entitled "Sooty."[1] SpookySOC is not intended to be a competitor to Sooty, nor is it an extension of Sooty. Sooty promotes the use of open source methodologies and approaches to problem solving, the same principles that SpookySOC is developed on. SpookySOC in fact has a notice during run-time that suggests utilizing Sooty in parallel with SpookySOC for result validation and support of the Sooty team.

[1] https://github.com/TheresAFewConors/Sooty

Project Description

SpookySOC has been in development for a handful of years, since I was a senior in high school. I always hated how many tabs I had open when I was looking at sketchy websites or possible malware samples. Initially written in Powershell, SpookySOC now only requires Python 3 and Python libraries - no external tools are utilized! A responsive and accurate terminal prompt sends you through all the motions, queries (free) external sources, and returns data in a multitude of usable formats. Most organizations require you sanitize data collected before documenting it or sending it to a customer: “defang” IP addresses/domains, anonymize users/workstation names, etc; SpookySOC makes this significantly easier by taking the manual work out of the equation. For later expansion, plaintext email “templates” can be provided in a configuration file and information will be populated into a template accordingly, ready to be copy and pasted into your ticketing system or documentation platform.

To name a few features beyond automated OSINT:

- Handles encoding/decoding of popular ciphers, primarily Base64.
- Shares threat intel to the community where possible (ex. uploading samples publicly to Virustotal, if permitted).
- Supports a variety of data ingests including hashes, IP addresses, domains, and email addresses.

More functionality can be expected as development continues, however this is the currently planned list of features for which modules are written or charted.

Innovation Claim

This project is innovative as it fills a hole that tools on-the-market don't provide. CarbonBlack links you to VirusTotal, but doesn't submit samples or show VirusTotal results in the same browser tab. CrowdStrike has a built-in IOC list that activity is compared against, however that list is maintained by CrowdStrike only. SpookySOC adds reputability and ease of use to the analyst's toolbox.

Usage Scenario

SpookySOC is best described in action with a practical example: a SOC analyst receives an alert for a malware event detected on a client's domain controller. The analyst's logs show a previously unexecuted file entitled "softwarefix.exe" made DNS requests to IP 1.1.1.1 as a resolver, attempting to resolve "example.com." SpookySOC allows the analyst to simply specify the IP address "1.1.1.1" and receive immediate results that indicate the IP address is a public DNS resolver maintained by Cloudflare and is trusted. In the same run, the analyst can include the domain "example.com" and retrieve information regarding the domain. This information is presented in an easy-to-read and ready-for-report format.

Evaluation Criteria

The following questions will identify the successful completion of the project:

- Can a user execute SpookySOC on the most recent versions of the three popular desktop OS (Windows, OS X Catalina, and Ubuntu 18.04, by market share) with only the Python 3 standard library and free Python packages?
- Does the user's input get parsed properly?
- Do domains, IPs, and hashes all get analyzed?
- Are special characters in domain names handled in such a way that does not cause the application to crash?
- Were results returned from threat intelligence sources?
- Is the data able to be pasted into a report with minor formatting required?

Objectives and Tasks Associated with the Project

Goal - Provide a toolkit that assists SOC analysts in their day-to-day duties while maintaining high reputability and ensuring accuracy of results through peer validation.

1. Objective 1 - Accept command-line arguments for IP addresses, domains, and hashes.
    a. Use Python's *argparse* module to accept command line arguments for IPs.
        i. Properly pass the IP address to lookup functions.

    b. Use *argparse* to accept arguments for hostnames.
        i. Properly pass the domain to lookup functions.
        ii. Resolve IP address of domain name at the time of lookup and perform IP lookups on the address that is resolved.
    c. Use *argparse* to accept arguments for hashes.
        i. Properly pass the hash to lookup functions.
        ii. If samples are found, attempt to gain further information from them and lookup IPs or domains that are associated.
2. Objective 2 - Lookup IP addresses against multiple threat intelligence sources.
    a. Use APIs from each source to send information about the IP.
    b. Accept data returned ("response") as JSON and store it in a file.
    c. Parse JSON response and display relevant information to the terminal.
3. Objective 3 - Lookup domains against multiple threat intelligence sources.
    a. Use APIs from each source to send information about the domain.
    b. Accept data returned ("response") as JSON and store it in a file.
    c. Parse JSON response and display relevant information to the terminal.
    d. Perform IP lookups (objective 2) against any IPs associated with this domain.
4. Objective 4 - Lookup hashes against multiple threat intelligence sources.
    a. Use APIs from each source to send information about the hash.
    b. Accept data returned ("response") as JSON and store it in a file.
    c. Parse JSON response and display relevant information to the terminal.
    d. Perform IP lookups (objective 2) and domain lookups (objective 3) against IPs and domains associated with this hash value.
5. Objective 5 - Store a copy of all output to the running location.

Description of Design Prototype

SpookySOC is written in Python 3.8 and is expected to continue working with all future Python 3.x versions. As Python is cross-platform by nature, SpookySOC can run on any operating system that has Python 3.x installed, and only requires that the user provide their free API keys in a YAML document. This install is entirely portable and can be moved from system-to-system with ease; all paths are referenced relatively (opposed to absolutely), allowing SpookySOC to be completely isolated to its operating folder and the modules that SpookySOC calls on to perform the lookup actions.

Evaluation Plan

SpookySOC will be evaluated throughout testing, and each feature added will be compared against the evaluation questions. As time goes on and a near-production-ready prototype is ready, analysts from the industry will be asked to include it in their daily use as much as possible. While one-off checks and demonstrations are easy to prove functionality, how does it stack up to constant use?

Project Completion Assessment

Note: This section must completed prior to SIP403.

Provide an in-depth description of the completion assessment of your project. Describe how well the completed components function and highlight the innovative facets of your design. This is sometimes known as a "Post-Mortem" or "Lessons-Learned Report". A good approach for this section is to answer the following 4 questions: "What went right? What went wrong? What was learned throughout the process? What would be done differently if you had to do it again?

Appendices

Note: While students are encouraged to start citing their sources as soon as SIP311, this section must completed prior to SIP403.

Include as appendices any supporting material for this project, including charts, graphs, and other data; images associated with the project; or other documentation (e.g., a game design document or read-me file). Include any prior art that was used such as U.S. Patent Documents, Foreign Patent Documents, or other sources. Remember that this section should only be a list of additional files, not the actual data of the files!

Use the following format:

Appendix letter: description of item – file name

Example…

Appendix A: Game design document – myGameDoc.docx

Appendix B: 3D render of primary character – mainCharacter.jpg

Appendix C: References

Author. (date). etc, following APA style.