PyFunnels: Data Normalization for InfoSec Workflows

TJ Nicholls
PyFunnels:

Data Normalization for InfoSec Workflows

Author: Student TJ Nicholls, Packetvitality@gmail.com

GNFA Gold

Advisor: Sally Vandeven

Accepted: January 2019

Abstract

Information security professionals cannot afford delays in their workflow due to the challenge of integrating data. For example, when data is gathered using multiple tools, the varying output formats must be normalized before automation can occur. This research details a Python library to normalize output from industry standard tools and act as a consolidation point for that functionality. Information security professionals should collaborate using a centralized resource that facilitates easy access to output data. Doing so will bypass extraneous tasks and jump straight to the application of practical data.
1. Introduction

There are various security tools and frameworks that efficiently collect and parse useful open-source intelligence (OSINT) data. Websites like OSINT Framework (Nordine, n.d.) and Pentester's Framework (Hatton & Robb, n.d.) are dedicated to cataloging the large breadth of information gathering tools. However, in many cases the various output formats from these tools makes data integration into workflows difficult. The challenge of accessing output is supported by observing prevalent OSINT tools or frameworks like SpiderFoot ("SpiderFoot: OSINT Automation," n.d.), Recon-ng (Tomes, 2018), and theHarvester (Martorella, 2018). SpiderFoot stores OSINT data in a SQLite database ("SpiderFoot: OSINT Automation," n.d.) and facilitates access to the data through a web interface, command line interface (CLI) and comma-separated value (CSV) file export. Using SpiderFoot’s CLI, the data can be accessed using various built-in commands or by directly querying the results database with Structured Query Language (SQL). Similarly, Recon-ng stores OSINT data in a SQLite database and the data is accessed through the CLI by using several built-in commands or by querying the results database with SQL (Helwig, 2016). Lastly, theHarvester provides OSINT data output in the form of XML and HTML files (Martorella, 2018).

A common requirement after the data collection process is the step of data normalization, which is essential before it can be integrated into workflows. This step is prevalent due to the fact that tools across the industry are not standardized and often use proprietary formats and terminology for their security content ("Part I Security Automation Essentials," n.d.). The problem isn’t that the data is inaccessible. In fact, in the case of SpiderFoot and Recon-ng, the data is well formatted in a SQLite database. The problem is when information security professionals work with disparate tools and try to incorporate a large amount of collected data into workflows; the presence of inconsistencies within the data can cause delays in assessments and remediation ("Part I Security Automation Essentials," n.d.).
This research proposes a data normalization solution by leveraging a custom built Python library to massage data points from OSINT tools and make the output readily available. To be clear, the solution is not another framework or tool but rather a library that makes the output easier to utilize. While users would still leverage tools in their normal capacity, when a pragmatic use case for the data is needed the library presented in this research would facilitate a simple way to access the gathered data.

1.1. Amplifying the Problem

The workforce for information security professionals is chronically understaffed. Frost & Sullivan (2017) forecast that by 2022 there will be a 1.8 million shortage of cyber security professionals. Many organizations struggle to staff and retain security operations teams due to the limited pool of skilled cybersecurity professionals (Blackshaw, 2017). Additionally, information security professionals often feel ill-equipped to manage the ever-evolving threat landscape and the vast amounts of data associated (Frost & Sullivan, 2017). Figure 1 below depicts the gap between employer demand and job seeker interest across various countries (Indeed, 2018). To address this issue, many organizations and professionals often strive to automate workflows to improve upon the efficiencies of everyday work.

![Figure 1: Gap between Employer Demand and Interest](image-url)

The Python library proposed in this research is intended to improve the efficiency of information security professionals by aggregating the output of tools and returning data in
a standardized format. Normalizing output data allows for easier integration into automated workflows. The library is essentially a catalog of code designed to capture the functionality of isolating data points from industry tools. The specifics of how the library is designed are covered in Section 1.4.

1.2. Benefits of Automation

It is beneficial for security professionals to identify what parts of a workflow can become more efficient as well as which can become automated. Martin Roesch, a Chief Architect at Cisco, suggested that one must make use of a fair amount of automation if one is to integrate a lot of security data quickly (Grachis, 2018). Most organizations can't properly manage security with their current resources because it has become increasingly complex and dynamic ("Part I Security Automation Essentials," n.d.). In some cases, to be successful as an information security professional one has to know how to automate tasks.

Automation improves the overall efficiency of information security professionals. In the current landscape where skilled information security professionals are difficult to recruit and retain, organizations require a force multiplier. Lamar (2016) suggested that automation can be a “force multiplier” to quickly, easily, and intelligently take action. For example, it’s common for security operation center (SOC) analysts to spend upwards of 90 minutes manually investigating suspected phishing emails (Morey, 2018). With automation, the same process can be completed in under a minute (Morey, 2018). Without the means for automation, information security professionals would not be able to keep up with daily requirements while also defending against emerging threats.

Particular tasks lend well to automation such as log analysis, asset management, and vulnerability enumeration. A 2018 survey of more than 600 IT and IT security practitioners by the Ponemon Institute discovered that seventy-two percent of respondents said their organizations use automation (Ponemon Institute LLC, 2018). Automation allows information security professionals to shift their focus to more serious threats and vulnerabilities (Ponemon Institute LLC, 2018). Enabling easy access to data is

TJ Nicholls, Packetvitality@gmail.com
a catalyst for automation because it removes hurdles and frees information security professionals to focus on utilizing the data.

1.3. Goals

The goal of this research is to create a collaborative code library which makes integrating data into automated workflows easier for information security professionals. The library acts as a centralized location where professionals can contribute and use code. This type of collaboration is important and beneficial to the community because it allows organizations to enhance their security processes through the work of others. In his Black Hat USA 2016 keynote Dan Kaminsky said, “We don’t compete on security. If one of us gets hit, we’re all going down so we should probably share our information.” Even though organizations may compete with products and services, it is crucial to collaborate when it pertains to information security.

1.4. Approach

This library will first be implemented by prioritizing established information security tools with an emphasis on OSINT. The library has the potential to be extensible and accommodate additional information security focuses such as penetration testing, network forensics, and auditing. The library was planned accordingly:

- Determine the tools with which to begin building integrations.
- Determine the data points for which to build integrations. For example, OSINT tools will likely store data points such as IP addresses, URLs, and email addresses. Data points which have the most overlap between tools will be prioritized.
- Build a class for each tool which contains methods to isolate data points, normalize the format, deduplicate the data, and return the output.

Figure 2 below conceptualizes how the library will function. To summarize, the library will support tools and more specifically data points within those tools. A class is used for each tool and contains methods for each data point it supports. The methods extract and return data in a standardized format. The methods from multiple tools can be

TJ Nicholls, Packetvitality@gmail.com
called to return aggregated data. This aggregation process normalizes and deduplicates the data for easy access. Lastly, users of the library will integrate that data into their specific use cases.

The library is currently being developed using Python3 which has become a standard programming language for information security tools. For example, Imperva (2018) stated, “more than 20% of GitHub repositories that implement an attack tool / exploit PoC are written in Python.” Using Python will allow for easy integrations with already established information security tools and workflows.

TJ Nicholls, Packetvitality@gmail.com
2. The Library

The library outlined in this research has been named PyFunnels due to data points from one or more tools being "funneled" together. The GitHub repository can be found at https://github.com/packetvitality/PyFunnels. PyFunnels consists of multiple classes structured modularly so that additional tools and data points can be easily added and work independently of one another.

Five tools have been selected to begin this project. Each tool is listed in Table 1 below and includes a description of its function.

<table>
<thead>
<tr>
<th>Tools</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpiderFoot</td>
<td>SpiderFoot is a reconnaissance tool that automatically queries over 100 public data sources (OSINT) to gather intelligence on IP addresses, domain names, e-mail addresses, names and more (&quot;SpiderFoot: OSINT Automation,&quot; n.d.).</td>
</tr>
<tr>
<td>theHarvester</td>
<td>theHarvester is a tool for gathering subdomain names, e-mail addresses, virtual hosts, open ports/ banners, and employee names (Martorella, 2018).</td>
</tr>
<tr>
<td>Photon</td>
<td>Photon is a crawler designed for OSINT. It can extract information such as URLs, emails, social media accounts, files, secret, Subdomains, and DNS-related data (Sangwan, 2018).</td>
</tr>
<tr>
<td>Nmap</td>
<td>Nmap is a utility for network discovery and security auditing (Fyodor, n.d.).</td>
</tr>
<tr>
<td>Recon-ng</td>
<td>Recon-ng is a full-featured Web Reconnaissance framework. Recon-ng provides a powerful environment in which open source web-based reconnaissance can be conducted quickly and thoroughly (Tomes, 2018).</td>
</tr>
</tbody>
</table>

Table 1: Selected Tools

Next, ten data points were selected to begin the project. The data points are listed in Table 2 below.

<table>
<thead>
<tr>
<th>Data Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
</tr>
<tr>
<td>IPv4 Address</td>
</tr>
</tbody>
</table>

Table 2: Selected Data Points

TJ Nicholls, Packetvitality@gmail.com
The classes within the library can be thought of as a catalog of tools and methods to retrieve data. Not all data point methods are required for each tool, meaning a new tool can be added with only a single method. Ideally, all data points would be supported for each tool but this structure allows the functionality to grow organically and makes it easy to contribute code to the project.

2.1. General Usage

The library reduces the time it takes information security professionals to utilize output from tools. The general workflow for using PyFunnels is described below, and a comparative use case is detailed in Section 2.2. For example, consider the following workflow:

1. Collect data with tool one.
2. Collect data with tool two.
3. Write code to isolate the data for tool one.
4. Write code to isolate and data for tool two.
5. Merge data into a standard format.
6. Remove duplicated data.
7. Expose normalized data.

To summarize, this workflow can be reduced to the following using PyFunnels:

1. Specify output files
2. Initiate an object.
3. Use method on the object.
4. Expose normalized data.

To demonstrate with more detail, a user creates an object and specifies output files for the tools they have used. For example, using the output of three tools, the workflow below could be used.

1. Identify the location of each output file and initiate a PyFunnels object. See Figure 3 below.
2. After creating the object, the user can call a method of the object to retrieve data points from the output files. The method to retrieve data points is named “funnel” and its purpose is to aggregate a data point from the output files. The aggregation process deduplicates and returns the data point in an expected format as an array. For example, calling the “funnel” method with “domains” as a parameter returns an array of domains from the provided tool output files. See Figure 4 below.

```
#Specify the output file for each tool in a dictionary.
source_files = {
    "tool1":"/myfilepath/tool1/tool1.db",
    "tool2":"/myfilepath/tool2/tool2.txt",
    "tool3":"/myfilepath/tool3/tool3.xml",
}

#Create a PyFunnels object.
PyF = PyFunnels(source_files)
```

**Figure 3: Initiating a PyFunnels Object**

```
print(PyF.funnel("domains"))
['example.com', 'partner.com', 'related.com']
```

**Figure 4: Calling PyFunnels Method**

3. The user can then integrate the data into their workflow as needed. The use cases are decided on by the end user; an example is provided in Section 2.2 below to demonstrate how it can be valuable for information security professionals.

### 2.2. Use Case

The use case in this section serves to demonstrate how an information security professional’s workflow could benefit from the ability to easily access the output of tools. It is beneficial to use multiple tools when performing reconnaissance. However, it can be cumbersome to aggregate the data from various tools into a unified dataset. The use case is presented from the perspective of performing the tasks with and without PyFunnels. The specifics of how the data points are isolated are not critical since there are always

TJ Nicholls, Packetvitality@gmail.com
multiple programmatic ways to accomplish the same task. The intention is to demonstrate the hurdle an information security professional must overcome prior to working with the data. PyFunnels is designed to alleviate this hurdle.

The use case is to identify services accessible via the Internet using OSINT tools for the purpose of comparing the identified services against a list of known services. It is common for an organization to unknowingly expose servers online due to forgotten servers which were never powered off, a disconnect between different departments who fail to communicate a server has been brought online, or testing environments that were not meant to be exposed. The unknown services may present additional risk to the organization since they may not be appropriately maintained or secured.

The tools leveraged for this use case are SpiderFoot, Recon-ng, theHarvester, Nmap, and Photon. The data point used in our example is "domains". When trying to identify unaccounted for public services it is also advisable to check for IP addresses since a host may be exposed without a viable DNS entry. However, to avoid being repetitive in the example below only the domains were used.

2.2.1. Working without PyFunnels

The following steps show how the data points could be collected without PyFunnels.

- The domain names will be collected from each tool and saved to an array. Figure 5 below contains the Python code used to complete this step.

```
domains = []
```

**Figure 5: Creating Array**

For SpiderFoot, the data will be pulled directly from the database, which is created after running the tool. This data is isolated using the steps outlined below. Figure 6 below contains the Python code used to complete these steps.

TJ Nicholls, Packetvitality@gmail.com
- SpiderFoot stores the results in the ‘data’ column. As a result, it is necessary to designate which database columns indicate that domains exist in the ‘data’ column.
- Open connection to the database.
- Extract the information from the ‘data’ column when appropriate.
- Add the domains to the array.
- Close the database connection.

```python
# Designate which database columns contain the relevant domain names.
domain_col_names = [
    "INTERNET_NAME",
    "SIMILARDOMAIN"
]

# Open connection to the database.
conn = sqlite3.connect("/path/to/file")
# Extract the information from the 'data' when appropriate.
for col in domain_col_names:
    cursor = conn.execute("SELECT data from tbl_scan_results WHERE 'type' = '{0}".format(col))
    for row in cursor:
        if row[0] not in domains:
            # Add the domains to the array.
            domains.append(row[0])

conn.close()
```

**Figure 6:** Extracting Domain information from SpiderFoot DB

The process is similar for Recon-ng. In this case, there wasn't a need to designate multiple column names because only domain name data is stored within the ‘host’ column of the ‘hosts’ table. Figure 7 below contains the Python code used to complete these steps.

```python
conn = sqlite3.connect("/path/to/file")
cursor = conn.execute("SELECT host from hosts")
for row in cursor:
    if row[0] not in domains:
        domains.append(row[0])

conn.close()
```

**Figure 7:** Extracting Domain information from Recon-ng DB

The next tool to collect data from is theHarvester. This tool provides results in HTML and XML format. The XML formatted output was chosen for easier parsing. This data is

TJ Nicholls, Packetvitality@gmail.com
isolated using the steps below. Figure 8 below contains the Python code used to complete these steps.

- Create the element tree object and get the root element.
- Extract the information from the tagged information when appropriate.
- Add the domains to the array.

```python
# Create the element tree object and get the root element
tree = ET.parse("/path/to/file")
root = tree.getroot()

# Extract the information from the tagged information when appropriate.
for child in root:
    if child.tag == "host":
        if child[1].tag == "hostname":
            if child[1].text not in domains:
                # Add the domains to the array.
                domains.append(child[1].text)
```

**Figure 8: Extracting Domain information from theHarvester XML Output**

The process is similar for Nmap’s XML formatted output. Figure 9 below contains the Python code used to complete these steps.

```python
# Create the element tree object and get the root element
tree = ET.parse("/path/to/file")
root = tree.getroot()

# Extract the information from the tagged information when appropriate.
for child in root:
    if child.tag == "host":
        for child2 in child[2]:
            if child2.attrib.get("name") not in domains:
                # Add the domains to the array.
                domains.append(child2.attrib.get("name"))
```

**Figure 9: Extracting Domain information from Nmap XML Output**

The last tool to collect data from is Photon. This tool provides results as individual files within a directory. This data is isolated using the steps below. Figure 10 below contains the Python code used to complete these steps.

- Find and open the appropriate file.

TJ Nicholls, Packetvitality@gmail.com
Add the domains to the array.

![Code snippet]

Figure 10: Extracting Domain information from Photon result file

At this point, all of the domains have been extracted from the output of the utilized tools and saved to an array. The user can now perform the intended action, comparing the discovered domains against a list of expected domains.

2.2.2. Working with PyFunnels

The following steps show how the data points could be collected with PyFunnels. Figure 11 below contains the Python code used to complete these steps.

- Specify the tool output files in a dictionary.
- Create a PyFunnels object.
- Use method on the object.

![Code snippet]

Figure 11: Aggregating Data with PyFunnels

One of the advantages of the library is removing extraneous tasks and jumping straight to the utilization of the data. At this point, all of the domains have been extracted.

TJ Nicholls, Packetvitality@gmail.com
from the output of the specified tools. The called upon method will return the domains as an array. The user can now perform the intended action, comparing the discovered domains against a list of expected domains.

2.3. **Expansion of the code library**

PyFunnels started with five tools and ten data points with a focus on OSINT data collection. The library has been purposely structured for ease of use and extensibility to new tools and data points. Users of the library are encouraged to contribute code for new tools and data points they find useful. Whenever a user creates Python3 code to isolate data from the output of a tool, he or she is encouraged to commit that code to PyFunnels so others in the community can use it as well. As a community driven project, PyFunnels could quickly grow to accommodate a wide range of tools and data points.
3. Conclusion

Many information security professionals utilize industry standard tools for day-to-day tasks. Furthermore, many professionals leverage multiple tools to do the same job due to the fact that different tools can provide different results. The end goal is not to derive data from these tools but rather to put that data to use. Since tools across the industry are not standardized, the information security community would benefit from a simple and reliable way to access a standardized output. This would reduce the overall time of an assessment and help streamline the process of remediation.

Delays that occur in an information security professional’s workflow are intolerable when considering the current shortage of qualified personnel in the information security workforce. Leveraging community support through open source projects allows information security professionals to benefit from each other's work. This collaboration reinforces the principle that organizations and professionals do not need to compete but support each other within the security field.

A solution such as PyFunnels would improve the efficiencies of information security professional’s data collection process. PyFunnels is a Python3 library containing methods to normalize data points from tools and make them easily accessible. PyFunnels has begun with a select number of OSINT related tools and data points but can easily be expanded on. The library is purposefully modular and allows programmers with varying skillsets to contribute code for new tools and data points. The PyFunnels library takes the functionality information security professionals often build in isolated environments and consolidates it into a centralized and free resource.

Providing easy access to non-standardized outputs helps facilitate the integration of that data into automated workflows. Automation can be a force multiplier for organizations but specifically those operating in information security related positions. With the ever changing digital threat landscape, the ability to automate tasks is critical for staying ahead of tomorrow's threats.

TJ Nicholls, Packetvitality@gmail.com
References


TJ Nicholls, Packetvitality@gmail.com
PyFunnels: Data Normalization for InfoSec Workflows


TJ Nicholls, Packetvitality@gmail.com

© 2019 The SANS Institute  Author retains full rights.
# Upcoming SANS Training

Click here to view a list of all SANS Courses

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Dates</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Chicago Spring 2020</td>
<td>Chicago, ILUS</td>
<td>Jun 01, 2020 - Jun 06, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS ICS Europe Summit &amp; Training 2020</td>
<td>Munich, DE</td>
<td>Jun 08, 2020 - Jun 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Budapest June 2020</td>
<td>Budapest, HU</td>
<td>Jun 08, 2020 - Jun 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Las Vegas Summer 2020</td>
<td>Las Vegas, NVUS</td>
<td>Jun 08, 2020 - Jun 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Zurich June 2020</td>
<td>Zurich, CH</td>
<td>Jun 15, 2020 - Jun 20, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>