2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Project Umbra

Timothy Wright, Shaun Whitfield, Sean Cahill, John Duffy
twright@nd.edu, swhitfie@nd.edu, scahill3@nd.edu, jduffy8@nd.edu
Center for Research Computing, University of Notre Dame, Notre Dame, IN 46556, USA
+1 574-631-2400

***Abstract*—Open Source Intelligence (OSInt) is the mining of public data sources to achieve some intelligence goal. Although activities range widely, the problems faced by OSInt analysts remain similar and include: managing the information flood of public data sources, establishing the veracity of information found, documenting the provenance of information found, and structuring results to enable sensible analysis. We intend to address each of these through a novel project called Umbra. Aimed at supporting human subject investigations, Umbra will help an analyst to: 1) filter out noise from relevant data, 2) track similar data from multiple sources as a form of corroboration, 3) annotate and time-stamp information found, and 4) tag/classify data.**

Keywords—Open Source Intelligence, OSInt, Umbra, human subjects, missing people, background search

I. Introduction

The province of Open Source Intelligence (OSInt) is the finding and transforming of public data to achieve some intelligence goal [1]–[6]. Regardless of the activity, be it locating a subject [7]–[9], monitoring open source code repositories [10] or Twitter for threats [11], or combing through multilingual news streams in real time [12], the OSInt domain is susceptible to an important set of common challenges [1], [3]–[5], [12], [13]:

- Managing the information flood produced by public data sources,
- Establishing the veracity of information found,
- Documenting the provenance of information found, and
- Structuring results to enable sensible analysis.

Our goal is to help the analyst address each of these through a novel project called Umbra. Designed for human subject investigations (e.g., background checks, finding information on a terrorism suspect, locating missing persons), Umbra supports an analyst's workflow as they: 1) filter out noise from relevant data, 2) track similar data from multiple sources as a form of corroboration, 3) annotate and time-stamp information, and 4) tag/classify data.

The Umbra system, currently being prototyped, employs a classic, tiered approach and is written largely in Python3. The server tier includes a Web application that incorporates a database, text search, and background job modules referred to as *OSInt processors*, which enable retrieval or analysis of data and other material. Umbra's client tier leverages an optional Mozilla Firefox add-on to streamline the capture of data by an analyst.

The remainder of this paper is organized as follows. Section 2 briefly examines other systems that share some of Umbra's feature set. In Section 3, we take a more detailed look at the Umbra prototype's architecture, capabilities, and use cases. Finally, in Section 4, we discuss why we believe Umbra is a compelling option for human subject OSInt activities, and make some concluding remarks.

II. Related Works

For expedience, we classify OSInt tools by their accessibility through a command line interface (CLI) or graphical user interface (GUI). Dividing this territory further, we also find tools whose purpose is to automate the search and retrieval of OSInt data, and others that offer a mix of manual and automated capabilities. Umbra is a GUI-based tool intended to be a hybrid of manual and automated aspects (though the prototype is largely manual).

Because of Umbra's human subject focus, its features and database are structured to support the gathering and filtering of information exclusively about people. Open source tools compatible with Umbra's code base and focus may be considered for integration through Umbra's OSInt processor modules. This can be accommodated by either adding modules to run and retrieve results from the tools in question, or assimilating tools into modules.

A. CLI Tools

Photon is a Python-based crawler that scrapes the Web for relevant OSInt data [14]. Capable of exploring existing websites or those archived through services such as the WaybackMachine (https://web.archive.org), Photon is able to recover data such as JavaScript files, email addresses, social media data, passwords, and API keys related to a target. The simplistic, customizable nature of Photon makes it a powerful automated tool for OSInt analysts.

IEEE/ACM ASONAM 2020, December 7-10, 2020
978-1-7281-1056-1/20/$31.00 © 2020 IEEE

PhoneInfoga aims to locate data on targets associated with a provided phone number. Written in Go, this tool collects and reports on a given phone number's country, line type, carrier, and legitimacy. It then automatically employs search engines to identify the owner, VoIP provider/line type, associated social media, and reputation reports from reverse phone lookups [15]. PhoneInfoga includes functionality to store results as CSV files.

Omnibus, coded in Python, investigates information "artifacts" about IP addresses, Bitcoin, email addresses, and file hashes [16]. Inspired by SpiderFoot (see GUI Tools, below) and other related projects, Omnibus uses an interactive CLI to access and configure modular plugins. Artifacts and discovered information are stored in MongoDB and Redis databases.

OSRFramework is composed of Python libraries that leverage separate OSInt applications for tasks such as username checking, DNS lookups, information leaks research, and deep web search [17]. Some Maltego transforms (see GUI Tools, below) are used to display information graphically. OSRFramework packages include tools that generate/guess nicknames, emails, and domains based on available information collected for a given target.

Other key CLI tools include ReconDog and OSIntSpy. ReconDog, developed in Python by the same team responsible for Photon, gathers basic information related to IP address and domain name data. This tool can accept the output of third party tools (piped through a command shell's STDIN), execute port scans, server name/IP lookups, and carry out a variety of other footprinting activities [18]. OSIntSpy, written in Python, uses external services to collect information about devices/domains, Bitcoin, email addresses and malicious files [9].

B. GUI Tools

Maltego is an automated, database-integrated, GUI tool used to dynamically collect intelligence for various targets. Written in Java, Maltego utilizes built-in and customizable "transforms," via the Maltego Transform Hub, to selectively explore data from public sources, commercial vendors, and internal data [19]. These programmatic mechanisms work to scour open-source data, compile relevant information on a specified target, and present results in a visualized graph format. Maltego is offered in a limited community edition or as a full-featured commercial platform.

SpiderFoot, developed in Python, offers a Web interface to query numerous public databases for target data such as, but not limited to, IP addresses, domain names, e-mail addresses, and usernames [20]. Although access to SpiderFoot is primarily through a browser, it also includes a command-line interface. Leveraging various modules to perform port scans, language identification, and screenshotting, SpiderFoot presents collected data in a simplified visual manner. As with Maltego, SpiderFoot includes both a limited community edition and a robust, commercial version.

III. The Umbra Prototype's Features and Architecture

Here, we discuss Umbra's purpose, paradigm of operation, software architecture, and how it aids the OSInt endeavor.

A. Umbra's Purpose as an OSInt Tool

At its core, Umbra supports an OSInt analyst's workflow to gather and process information. It does this by enabling a simple, coherent means of ingesting and organizing data and materials obtained during an investigation. Moreover, Umbra is designed to be deployed as a tool used by a single analyst on a laptop, or as a shared system with any number of simultaneous analysts operating remotely.

Umbra organizes OSInt data along two axes: *portfolios* and *targets*. A portfolio equates to a case; it is primarily a containing artifice, within which the information for one or more targets is tracked. Portfolios also gather high-level information, such as triage data, data about potential leads, and general scratchpad information. A target, meanwhile, represents a human subject and their wide variety of related information (biographical data, text, images, video, computer addresses, and so forth). To make OSInt activities more flexible, any given target can be associated with more than one portfolio simultaneously.

Umbra supports OSInt workflow phases that include: triage (the initial evaluation of a case and how investigative activities will proceed), collection (ingesting, annotating, and categorizing data), analysis (visualizing target data for the analyst to review, also the use of tools such as language translation, natural language processing, and image processing), and reporting (the presentable, organized output of case information) [21].

B. Paradigm of Operation

Umbra deployments are intended to easily scale from that of a single-user laptop, to a dedicated server for any number of remote analysts. Regardless of the deployment, Umbra's feature set remains the same--though performance will naturally be commensurate with the CPU, memory, and storage resources on hand. To help expedite an Umbra installation, the option to use Docker containers will eventually be available.

An interesting characteristic of Umbra is its use of access controls. To begin with, there is no administrative role; all Umbra users have the same level of privilege. In part, this is because, when used by multiple analysts, Umbra presumes the team of users knows and trusts one another. We feel this is a very modest presumption, given the sort of OSInt work relevant to Umbra, e.g., [7]–[9]. Also, because OSInt data are public by nature, the need for protection based on sensitivity is limited. The net effect of Umbra's approach is the easy sharing of portfolios and targets among analysts: any analyst may assign or remove themselves to/from a portfolio, and any target may be tracked by, or removed from a portfolio. That said, if an analyst is not assigned to a given portfolio, they cannot view or edit that portfolio or its targets (unless those targets are tracked by another portfolio to which the analyst is assigned). To access Umbra (or its API mentioned below), an

analyst provides a unique user ID and password, which are created when the analyst first interacts with Umbra. If a password is forgotten, a secure reset is handled through email and a temporary token. All communications with Umbra are protected by TLS.

C. Architecture

Umbra's major software components include a Python Web application (written using the Flask library), a relational database (MariaDB), the Mozilla Firefox Web browser, and an optional Firefox add-on. Supporting server elements include Gunicorn and nginx, as well as Elasticsearch and Redis Queue.

The Umbra Flask application provides a standard Web service and also implements a REST API. The Web service handles all interactive work with Umbra (e.g., logging into Umbra, creating and managing portfolios and targets, running reports). The API forms an interface for the optional Firefox add-on, which permits an analyst to easily select and store text, images, video, and audio data in Umbra. Of course, any tool able to communicate with the API can access Umbra in the same manner as the Firefox add-on.

Gunicorn provides a Python-based Web Server Gateway Interface (WSGI) service. This handles kicking off the Flask Umbra application to handle requests and facilitates communication between the nginx Web server and Umbra. Nginx, then, provides the outward-facing Web service to which Umbra users connect via HTTPS.

Elasticsearch enables the ability to search text data stored in Umbra. This permits an analyst to locate portfolios and targets through natural language searches.

Finally, Redis Queue, encompassing the Redis database and the Redis Queue Python library, enables the execution of background jobs. Implemented through Umbra's OSInt processors, these jobs are intended to handle, for example, text translation, searching media files for patterns, obtaining data from third party services, or report generation.

A typical Umbra use-case entails the following.

1. An analyst browses Web resources during an investigation, selecting and transmitting to Umbra text/images/video/sound data that are relevant
2. Data received by Umbra through its API are stored in the Umbra database
3. OSInt processor tasks run asynchronously in the background to further evaluate/analyze or add to Umbra data
4. The analyst can interact directly with the Umbra Web application to view, manage, and report on portfolios and targets

Umbra's typical usage, server-side elements, and software stack are all depicted in Figure 1.

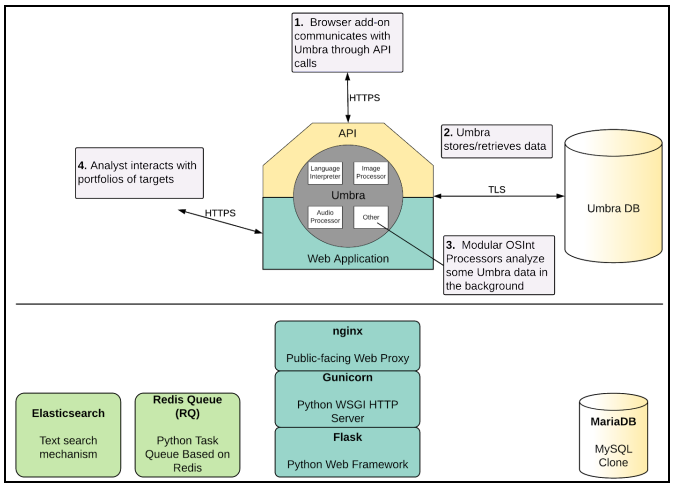

Fig. 1. Umbra usage and software stack.

D. How Umbra Helps

Umbra's stated goal is to address the OSInt domain challenges noted in the introduction. This is achieved through Umbra's workflow support as outlined below.

- Information Flood -- The Umbra Firefox add-on enables an analyst to select, categorize, and transmit data directly to Umbra for targets being investigated. In addition, Umbra's OSInt processors can automate some data evaluation and discovery activities in the background.
- Information Veracity and Provenance -- Veracity and provenance are obtained by categorizing and structuring information (helping to reveal corroborating elements), as well as tracking source URLs, timestamps, and other metadata.
- Sensible Information Structuring -- Effective grouping/managing of OSInt information is enabled through the use of portfolios and targets, along with access controls conducive to appropriate, streamlined information sharing.

IV. Conclusion

The domain of OSInt is focused on "information that is publicly available material that anyone can lawfully obtain by request, purchase, or observation" [1]. Collecting and converting such information into actionable intelligence is frustrated by issues such as information flood, determining veracity and provenance, and structuring data in a coherent, useful way. Umbra, currently being prototyped, is a Web application system designed to address each of these important issues by supporting the OSInt analyst's workflow activities. Through its tiered client/server approach, Umbra directly supports workflow phases that include triage, collection, analysis, and reporting. This is enabled through several helpful features, such as an optional Firefox add-on for easy collection of information, and a Web service that coherently categorizes, stores, and can process/report on information. Umbra is designed for deployments of any size, from that of a

single analyst on a laptop, to a team remotely working on a server.

Umbra is well-positioned for future expansion. Subsequent versions of the Umbra system will build upon the existing OSInt processor module capability. This will facilitate reporting, integrating external tools, and the incorporation of features such as artificial intelligence for tasks like facial recognition and pattern discovery. Additional plans are to containerize Umbra for ease of deployment and improve on Umbra's various built-in tools to better aid the analyst in interpreting data.

Acknowledgment

The authors wish to acknowledge the support of the Office of Naval Research grant N00014-20-1-2061 for student contributions made to this paper.

References

[1] R. A. Best and A. Cumming, *Open Source Intelligence (OSINT): Issues for Congress*. Library of Congress. Congressional Research Service., 2007.

[2] N. Hassan, "An Introduction To Open Source Intelligence (OSINT) Gathering," *Secjuice*, Aug. 12, 2018. https://www.secjuice.com/introduction-to-open-source-intelligence-osint/ (accessed Sep. 02, 2019).

[3] G. Hribar, I. Podbregar, and T. Ivanuša, "OSINT: A 'Grey Zone'?," *Int. J. Intell. CounterIntelligence*, vol. 27, no. 3, pp. 529–549, 2014.

[4] J. Richardson, "The challenges of Open Source Intelligence (OSINT) in an increasingly connected age," *Global Results Communications*, Apr. 14, 2017. https://www.globalresultspr.net/blog/the-challenges-of-open-source-intelligence-osint-in-an-increasingly-connected-age/ (accessed Sep. 02, 2019).

[5] F. Sampson, "Intelligent evidence: Using open source intelligence (OSINT) in criminal proceedings," *Police J. Theory Pract. Princ.*, vol. 90, no. 1, pp. 55–69, 2017.

[6] S. A. Stottlemyre, "HUMINT, OSINT, or Something New? Defining Crowdsourced Intelligence," *Int. J. Intell. CounterIntelligence*, vol. 28, no. 3, pp. 578–589, 2015.

[7] AccessOSINT, "Tracelabs--Searching for our missing persons with OSINT!," *Medium*, Aug. 27, 2018. https://medium.com/@osint/tracelabs-searching-for-our-missing-persons-with-osint-da6e9ea68daa (accessed Sep. 02, 2019).

[8] J. Cox, "The Hackers Hunting Down Missing People," *Motherboard Tech by VICE*, Aug. 15, 2018. https://www.vice.com/en_us/article/qvmm3x/hackers-hunting-missing-people-osint-defcon-tracelabs? (accessed Sep. 02, 2019).

[9] B. Stark, "Interview Series: Learn How Trace Labs Use Crowdsourced OSINT to Find Missing Persons with Robert Sell," *Intelligence101.com*, Apr. 16, 2019. https://www.intelligence101.com/tracelabs/ (accessed Sep. 02, 2019).

[10] L. Neil, S. Mittal, and A. Joshi, "Mining Threat Intelligence about Open-Source Projects and Libraries from Code Repository Issues and Bug Reports," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Nov. 2018, pp. 7–12, doi: 10.1109/ISI.2018.8587375.

[11] A. Bose, V. Behzadan, C. Aguirre, and W. H. Hsu, "A Novel Approach for Detection and Ranking of Trendy and Emerging Cyber Threat Events in Twitter Streams," *CoRR*, vol. abs/1907.07768, 2019, [Online]. Available: http://arxiv.org/abs/1907.07768.

[12] C. Best, "Challenges in Open Source Intelligence," in *2011 European Intelligence and Security Informatics Conference*, Sep. 2011, pp. 58–62, doi: 10.1109/EISIC.2011.41.

[13] G. Backfried, C. Schmidt, M. Pfeiffer, G. Quirchmayr, M. Glanzer, and K. Rainer, "Open Source Intelligence in Disaster Management," in *2012 European Intelligence and Security Informatics Conference*, 2012, pp. 254–258.

[14] S. Sangwan, "Photon GitHub Project Page," *Photon*, Dec. 06, 2019. https://github.com/s0md3v/Photon (accessed Apr. 18, 2020).

[15] R. C., "PhoneInfoga GitHub Project Page," *PhoneInfoga*, Apr. 17, 2020. https://github.com/sundowndev/PhoneInfoga (accessed Apr. 18, 2020).

[16] C. Morrow and P. Amini, "OmniBus GitHub Project Page," *Omnibus*, Nov. 07, 2019. https://github.com/InQuest/omnibus (accessed Apr. 18, 2020).

[17] Y. Rubio and F. Brezo, "OSRFramework GitHub Project Page," *OSRFramework*, Apr. 10, 2020. https://github.com/i3visio/osrframework (accessed Apr. 18, 2020).

[18] S. Sangwan, "ReconDog GitHub Project Page," *ReconDog*, May 05, 2019. https://github.com/s0md3v/ReconDog (accessed Apr. 18, 2020).

[19] Maltego Technologies, "Maltego and its products: Basics," *Maltego*, Jan. 23, 2019. https://docs.maltego.com/support/solutions/articles/15000020188-what-can-i-use-maltego-for- (accessed Apr. 08, 2020).

[20] SM7 Software OÜ, "SpiderFoot Homepage," *SpiderFoot*, 2020. https://www.spiderfoot.net/ (accessed Apr. 18, 2020).

[21] M. Bazzell, "Methodology and Workflow," in *Open Source Intelligence Techniques: Resources for Searching and Analyzing Online Information*, 7th Ed., IntelTechniques.com, 2019, pp. 503–529.