



University of West Attica

Faculty of Engineering

Department of Informatics and Computer Engineering

Διπλωματική Εργασία

Tracking Cyber-Adversaries and Constructing their Digital Profile

Καβαλιέρου Θάλεια - Ελπίς

AM: 711171009

Εισηγήτρια: Ιωάννα Καντζάβελου

Tracking Cyber-Adversaries and Constructing their Digital Profile

Tracking Cyber-Adversaries and Constructing their Digital Profile

Diploma Thesis

Tracking Cyber-Adversaries and Constructing their Digital Profile

Kavalierou Thaleia-Elpis

Registration Number: 711171009

Εισηγήτρια:

Ιωάννα Καντζάβελου, Επίκουρη Καθηγήτρια

Εξεταστική επιτροπή

Αντώνιος Μπόγρης, Καθηγητής

Χριστίνα Γεωργουλάκη, ΕΔΙΠ

Ημερομηνία εξέτασης: 21/03/2024

Tracking Cyber-Adversaries and Constructing their Digital Profile

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

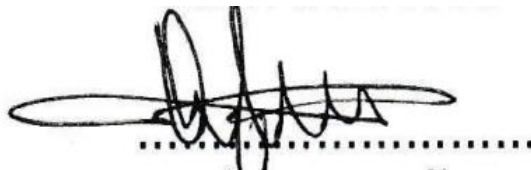
Η κάτωθι υπογεγραμμένη Καβαλιέρου Θάλεια-Ελπίς του Ελευθερίου, με αριθμό μητρώου 711171009 φοιτήτρια του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

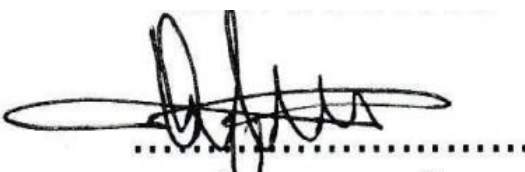
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

«Βεβαιώνω ότι είμαι συγγραφέας της παρούσας διπλωματικής εργασίας και ότι έχω αναφέρει ή παραπέμψει σε αυτή, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για την συγκεκριμένη διπλωματική εργασία»

Η Δηλούσα



Tracking Cyber-Adversaries and Constructing their Digital Profile



Καβαλιέρου Θάλεια-Ελπίς

Σχολή Μηχανικών

Τμήμα Μηχανικών πληροφορικής και Υπολογιστών

Πανεπιστήμιο Δυτικής Αττικής

Copyright © Καβαλιέρου Θάλεια-Ελπίς, 2024

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευτεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Δυτικής Αττικής.

Tracking Cyber-Adversaries and Constructing their Digital Profile

ACKNOWLEDGEMENTS

I would like to extend my sincerest gratitude to Professor Kantzavelou for her invaluable assistance and guidance throughout the process of crafting my diploma thesis. Her unwavering commitment to academic excellence and her willingness to invest her time and expertise in mentoring me have been instrumental in shaping the trajectory of my research. Professor Kantzavelou's insightful feedback, constructive criticism, and encouragement have not only refined the quality of my thesis but also enriched my understanding of the subject matter.

Moreover, I am deeply appreciative of Professor Kantzavelou's accessibility and responsiveness, as she consistently made herself available to address my queries and concerns, demonstrating a genuine dedication to my academic growth. Her profound knowledge, coupled with her approachable demeanor, created an environment conducive to learning and exploration. I am indebted to her for instilling in me a sense of confidence and scholarly rigor, which I will carry forward into my future endeavors. Professor Kantzavelou's mentorship has been a cornerstone of my academic journey, and for that, I extend my heartfelt gratitude.

Tracking Cyber-Adversaries and Constructing their Digital Profile

ΠΕΡΙΛΗΨΗ

Μέσα από την ανάλυση των πιο εξελιγμένων, σε επίπεδο εκτέλεσης και αντίκτυπου, επιθέσεων είναι εύκολο να διακρίνουμε πως, η μεθοδολογία των ορισμένων επιτιθέμενων χαρακτηρίζεται, έντονα, από αυστηρή μεθοδικότητα και απόλυτη ακρίβεια κατά τη διεκπεραίωση των βημάτων τους.

Καθίσταται, λοιπόν, λογικό να υποθέσουμε ότι, ο επιτιθέμενος έχει εμπλουτίσει το γνωσιακό του υπόβαθρο μέσα από την εμπειρία του σε παρόμοιες επιθέσεις και την αλληλεπίδραση του με άτομα που απασχολούνται, επαγγελματικά και ερασιτεχνικά, με τον κλάδο της Πληροφορικής. Συνεπώς, μπορούμε να συμπεράνουμε πως, προτού φτάσει στον τελικό του στόχο, έχει ήδη αποπειραθεί να εκτελέσει την επίθεση σε άλλους οργανισμούς και έχει κατακτήσει την απαραίτητη τεχνογνωσία μέσω διαδικασίας σφάλματος-ανάκαμψης.

Στόχος αυτής της διπλωματικής εργασίας είναι η πρόταση μεθόδου σύστασης του προφίλ ενός επιτιθέμενου, όπου με τον όρο «επιτιθέμενος» στην προκειμένη περίπτωση μπορεί να προσδιορίζεται ένα άτομο ή μία APT (Advanced Persistent Threat) ομάδα, βασιζόμενη στα εσφαλμένα/παράτολμα βήματα του κατά τα δοκιμαστικά στάδια που προηγήθηκαν της επίθεσης. Τα δεδομένα που αφορούν τους εκάστοτε επιτιθέμενους, θα συγκεντρώνονται σε μία κεντρική Βάση Δεδομένων, η οποία θα εξυπηρετεί Threat Intelligence σκοπούς, θα αξιολογούνται με αλγόριθμους Μη Επιβλεπόμενης Μηχανικής Μάθησης και θα ταξινομούνται μέσω διαδικασίας Social Network Analysis σε ανάλογα δίκτυα. Το πρόβλημα που θα επιλύουν οι αλγόριθμοι είναι η αντιστοίχιση των δεδομένων υπό το προφίλ πιθανού επιτιθέμενου με άλλων προφίλ ήδη καταγεγραμμένων επιτιθέμενων.

Λέξεις- Κλειδιά: Κυβερνοασφάλεια, ιχνηλάτηση κυβερνο-επιτιθέμενων, μηχανική μάθηση, νέφος

Tracking Cyber-Adversaries and Constructing their Digital Profile

ABSTRACT

Through the analysis of the most advanced attacks in terms of execution and impact, it is easy to discern that the methodology of certain adversaries is characterized strongly by strict orderliness and absolute precision in carrying out their steps.

Therefore, it becomes reasonable to assume that the adversary has enriched their knowledge base through experience in similar attacks and interaction with individuals professionally and amateurishly engaged in the field of Information Technology. Consequently, we can infer that before reaching their ultimate goal, they have already attempted to execute the attack on other organizations and have acquired the necessary expertise through a trial-and-error process.

The objective of this thesis is to propose a method for establishing the profile of an adversary, where the term “adversary” in this case can refer to an individual or an APT (Advanced Persistent Threat) group, based on the erroneous/bold steps taken during the testing stages preceding the attack. Data concerning the respective attackers will be collected in a central Database, which will serve Threat Intelligence purposes, will be evaluated using Unsupervised Machine Learning algorithms, and will be classified through a Social Network Analysis process into relevant networks. The problem that the algorithms will solve is matching the data under the profile of a potential threat actor with other profiles of already recorded actors.

Keywords: Cybersecurity, threat actor attribution, unsupervised machine learning, social network analysis, cloud

Tracking Cyber-Adversaries and Constructing their Digital Profile

CONTENTS

Acknowledgements.....	8
Περίληψη.....	10
Abstract.....	12
Table of Figures	16
1. Introduction	16
1.1 The goal and the necessity	19
1.2 Machine Learning for the rescue.....	20
1.3 SNA.....	21
2. Building a cyber profile	22
2.1 Human hiding in logs.....	25
2.2 Indication of Compromise becomes indication of Compromiser	27
2.3 The Reverse Pyramids of Goals	29
2.4 An Overview of a Threat Actor's Profile	32
3. The Framework Architecture.....	35
4. Initial Stage of the Framework - Gathering logs from diverse sources	38
4.1 Gathering logs from the web application	40
4.2 Gathering logs from the Intrusion Detection System	49
4.3 Collecting public information from open-source knowledge bases	55
5. Middle stage of the framework - Assembling distinct technical artifacts under individual profiles.....	58
5.1 Building the Web Logs part	59
5.2 Building the Networking Data part.....	60
5.3 Building the Attack Methods part.....	61
5.4 Building the Miscellaneous part	63
5.5 Building the final profile	64
6. Third stage of the framework – Implementation and Testing.....	69
6.1 Data vectorization with TF-IDF	69
6.2 Calculating affinity between cyber profiles with Cosine similarity	71
6.3 Utilizing Social Network Analysis for networking graph generation	72
6.4 Enhancing the plotting feature for web application integration.....	82
7. Leveraging the Framework: Future Application and Implementation	85
7.1 Security-as-a-Service model.....	86

Tracking Cyber-Adversaries and Constructing their Digital Profile

7.2 Transition to supervised machine learning89

8. Conclusions.....92

References.....95

Appendix 198

Appendix 299

Appendix 3100

Appendix 4101

Appendix 5102

Appendix 6103

TABLE OF FIGURES

Figure 1: Threat actors represented as a subcluster of network users.17

Figure 2: A threat actor's process of collecting experience and knowledge before reaching the "zero-day".
..... 18

Figure 3: The visual comparison of a threat actor's level and their targeted systems. 31

Figure 4: Visual representation of a threat actor's cyber profile. 34

Figure 5: Framework's architecture..... 36

Figure 6: An example of Apache web server access log structure. 40

Figure 7: An example of Apache web server error log structure. 43

Figure 8: Corresponding Apache web access and error logs for the user with IP address 10.2.25.83..... 47

Figure 9: Corresponding Apache web access and error logs for the user with IP address 10.2.23.136.... 48

Figure 10: An example of Snort IDS log structure. 49

Figure 11: Corresponding Snort IDS logs for the user with IP address 10.2.23.136..... 54

Figure 12: The complete profile of the user with IP address 10.2.25.83. 67

Figure 13: The schemas of the "web_access", "snort_logs" and the, concluded, "profiles" databases. 68

Figure 14: User's "user_10.2.25.83" position in the generated network graph..... 80

Figure 15: User's "user_10.2.23.136" position in the generated network graph..... 81

Figure 16: A node representing a user's profile and edges connecting profiles based on their similarities.
..... 93

Figure 17: Overview of the framework as a web application. 94

1. INTRODUCTION

Tracking Cyber-Adversaries and Constructing their Digital Profile

In the context of this research, we commence with a simple hypothesis. Humans, acting as users-nodes, leave traces and footprints of their presence in the digital landscape, in a similar way with the real world. Having this hypothesis as base, along with the assumption that behind a user's endpoint we can find a human actor, we aim to identify the affiliation between digital footprints that will lead to a "digital trail" or, more specifically, a "digital profile".

Knowledge level, learning pace, skills, methodology, tactics, even intentions among users on the internet differ in various ways. Malicious actors/adversaries constitute just a subset of this set, which inherits the same attribute. However, when we face attacks of the most advanced kind, where we observe infallible maneuvers, able to bypass every security countermeasure, most of the times undetected, we tend to forget that behind this activity there is a human, as well.

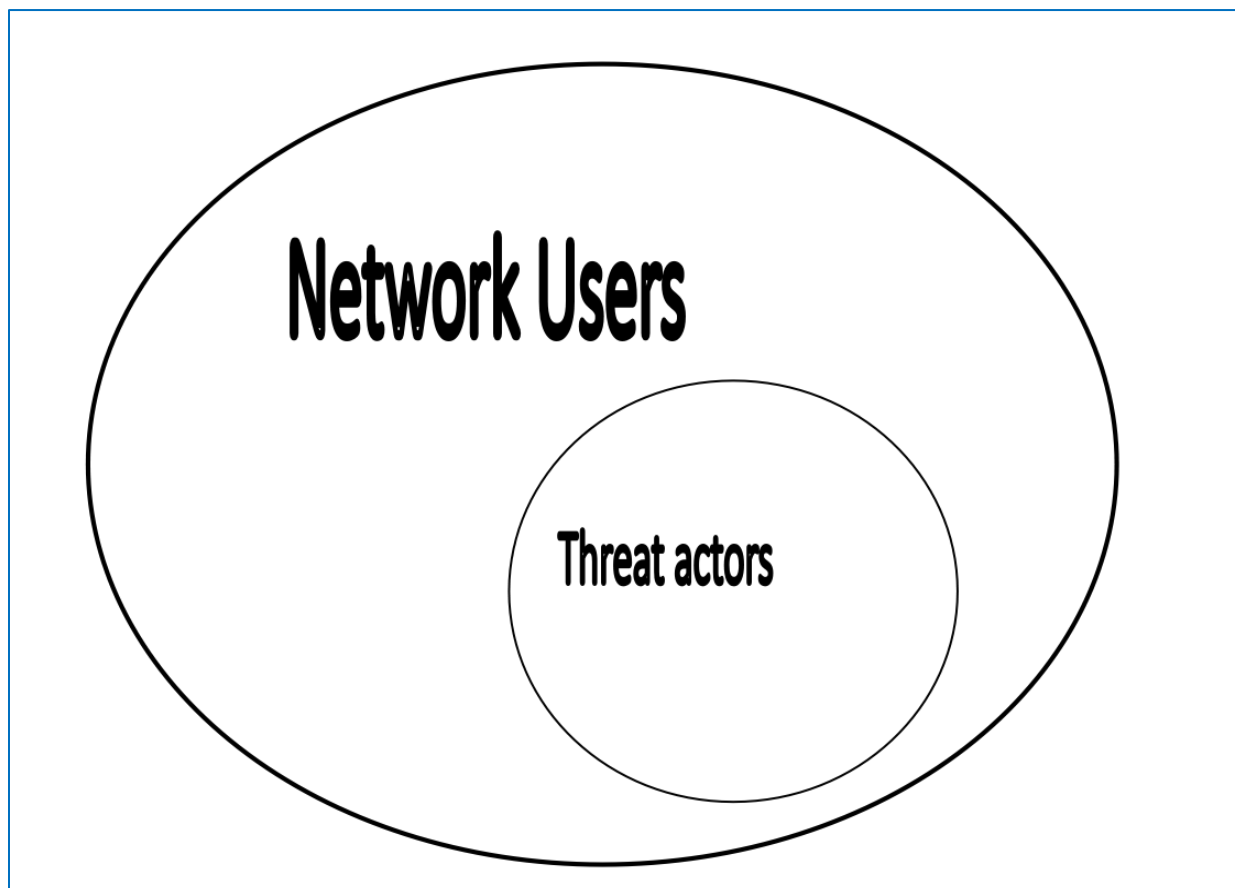


Figure 1: Threat actors represented as a subcluster of network users.

Having this in mind, we can assume that this person, our adversary, reached to the day of the attack, which we will define as "zero-day", by collecting experience and

Tracking Cyber-Adversaries and Constructing their Digital Profile

knowledge from several attempted attacks, and has conquered the required expertise through an error-recovery process. That timespan that preceded the zero day will be examined in this research.

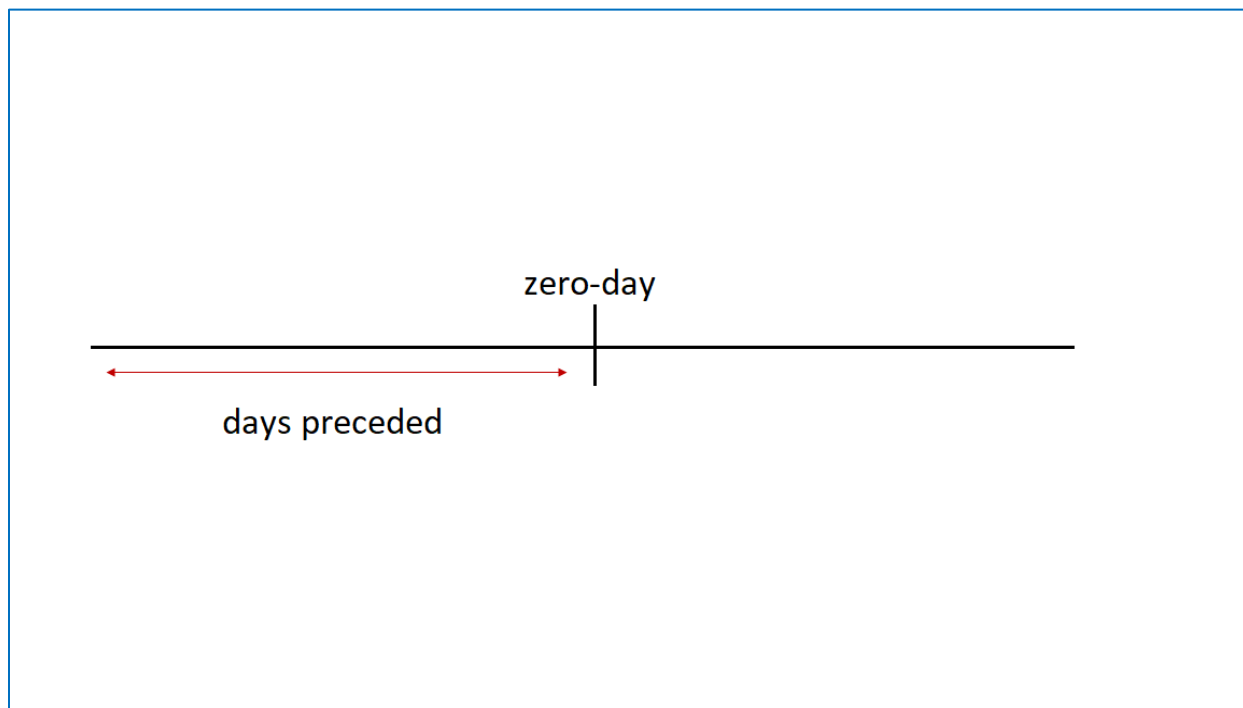


Figure 2: A threat actor's process of collecting experience and knowledge before reaching the "zero-day".

First goal is to identify suitable *technical artifacts/criteria* to match a real-time attacker's actions with others, significant of previous attacks. Research of experts in the cybersecurity field and after-attack system analysis will be a starting point for finding the right criteria. This empirically knowledge already form a robust database and foundation for the current study.

The criteria will be applied to unsupervised machine learning and SNA algorithms, with the objective of evaluating the affiliation score between the records of a real-time attack and those of prior recorded, succeeded or not, attacks.

Furthermore, we will introduce a cloud-based distribution approach. For this application, our model can be distributed as a Security-as-a-Service [1] within a cloud infrastructure, gathering data from various sources into a central Data Collector. This will allow us to analyze the data for potential signs of adversaries and their connections across the network.

1.1 THE GOAL AND THE NECESSITY

Currently, in the cybersecurity landscape, the predominant method for detecting cyberattacks relies heavily on rule-based approaches. Experts create rules to match indications within logs of Intrusion Detection Systems, Firewalls and Web Firewalls, Endpoint Detection and Response systems, Security Information and Management systems, etc. On opposite, what we try to achieve is a criteria-based methodology to detect signs of attackers based on their behavior. We could describe this as a popular board game called “Guess Who”. Imagine playing this game with a board of users. Characteristics connected with their cyber presence would differentiate one from another. With the right criteria, we would eliminate them down to one, our cyber adversary.

Much like in the game, many users, even in the subset of cyber adversaries, share similar characteristics. The most popular tools, like “nmap” [2], “gobuster” [3], “sqlmap” [4], etc., common wordlists, like “rockyou.txt” and the “seclists” collection, etc., even strings, e.g. “admin” or “test”, have a prevalent presence in logs originating from numerous diverse sources and are possible to, mistakenly, define users completely unrelated. Consequently, when attempting to reach our sole user, who exhibits the strongest connection with our targeted profile, we need to prune a decision tree, somehow like removing branches per level.

But why do we need to shift from a rule-based framework of a “Guess what?” game to a criteria-based one of a “Guess who?” game? What the criteria-based detection introduces in this framework is the capability to attribute an attack and establish a knowledge base that aids in preparing for future attacks from the same or affiliated actors.

In K. Poireault’s article “*Threat Intelligence: Why Attributing Cyber-Attacks Matters*” [5] experts in the field comment on the significance of collecting and connecting technical artifacts in order to attribute an attack to an attacker. A. Leslie from *Recorded Future* stated that “Attribution really matters because it allows you to think about how you can best strategize and predict future attacks”. In a Trend Micro’s blog post, it was mentioned that “that attribution can help identify if victims are a target or collateral damage, better understand the tactics, techniques and procedures (TTPs) used during an attack to enhance detection and response, and help the board see the investment value in new security tools”, while their senior threat researcher F. Hacquebord noted that “When we talk about attribution at Trend Micro, we’re only referring to technical attribution, devoid of legal or political purposes”. S. Ronis from Mandiant raised the concerns on future attacks and the value of gathering knowledge, “Let’s say you’ve blocked an attack and remediated it or contained it – how do you guarantee that you’ll be able to block them

again if they come after you in a different way? The best way to do that is to get some degree of knowledge”.

Finally, F. He, of Group-IB, provided some information on their workaround of attributing attacks to adversaries, describing how “CTI teams cross-check data from various attacks and operations to unveil similarities and patterns” which they “will organize such discoveries under a given profile, that will be given a code name”.

Concluding this paragraph, it is crucial to highlight the challenges posed by the complex process of cyberattack attribution. Trend Micro’s experts stated on that [6], “Another challenge is that attribution can take a long time and may not provide immediate value during a cyber incident”. As a matter of fact, the amount of data that needs evaluation imposes a substantial burden on analysts engaged in the attribution process. Hence, arises the necessity for a framework capable of implementing their logic implementation and decision-making while expediting the process to achieve a more suitable timing for real-time detection and response.

1.2 MACHINE LEARNING FOR THE RESCUE

The data required for accurate threat actor attribution has the potential to expand and encompass an excessive amount of information, making it challenging to manage effectively. Logs originating from web applications, underlying servers supporting them, and security monitoring software contain, in a substantial quantity, legitimate activity of desired users for the exposed product, service or organization. However, within these logs, evidence of undesirable and malicious activities will also be documented.

In the context of this research, machine learning will play a pivotal role in processing technical artifacts hidden in logs and finding similarities that could lead to attributing threat actors for specific attack vectors and patterns. Despite considering logs generated by pre-existing security software installed on the systems during profile creation, we will not utilize their logic for identifying the malicious activity. Unsupervised machine learning algorithms will perform this task as traditional rule-based approaches could fall short in capturing subtle patterns and anomalies. Furthermore, machine learning excels at recognizing intricate correlations and deviations within these datasets.

Cosine similarity [7] is a mathematical technique used to measure the similarity between two vectors in a multi-dimensional space. It quantifies the directional relationship between these vectors, indicating the extent to which they align in direction, irrespective of their magnitude. Cosine similarity is widely employed in various fields, including natural

Tracking Cyber-Adversaries and Constructing their Digital Profile

language processing, information retrieval, and recommendation systems. In the context of cybersecurity and threat analysis, cosine similarity provides a valuable means of comparing data vectors to identify patterns, anomalies, and similarities, helping in the detection of potential threats and the organization of large datasets for analysis. Its efficiency and simplicity make it a popular tool for similarity measurement in diverse applications.

Within our framework, the cosine similarity will function as a determining method, augmented by the integration of Social Network Analysis (SNA). Here, textual data from user profiles is leveraged to reveal concealed relationships and associations. By leveraging the principles of Term Frequency-Inverse Document Frequency (TF-IDF) vectorization and cosine similarity, the code evaluates the similarity between pairs of user profiles based on their textual attributes. This quantification of similarity allows for the creation of a network graph that visualizes mutual connections between profiles, offering valuable insights into underlying patterns and associations within the dataset.

1.3 SNA

Social Network Analysis (SNA) [8] is a methodological approach that examines and interprets the relationships between individuals, groups, or entities within a social system. Its objective is to visualize, quantify, and comprehend the patterns of connections that shape networks similar to those extracted from datasets and examined during the process of attributing cyber attackers. By depicting these connections as nodes (representing individuals or entities) and edges (indicating relationships), SNA allows for the visualization of intricate patterns, revealing the structure of threat actor networks and their behaviors. This visualization aids in identifying potential attack vectors, understanding the dynamics of security incidents, and attributing threats with a higher degree of accuracy. Social Network Analysis empowers our cybersecurity framework to construct detailed and interpretable graphs of attacker profiles, facilitating a more comprehensive view of the threat landscape and bolstering our capabilities in attribution and threat detection.

In addition to its graphical representation, Social Network Analysis is instrumental in quantifying and analyzing network properties. Measures such as centrality, degree, and betweenness facilitate the identification of key players and nodes in the threat network. This information allows security analysts to prioritize response actions and focus on the most critical elements of a security incident. Moreover, SNA can be applied in collaborative defense efforts, enabling organizations to share threat intelligence and

Tracking Cyber-Adversaries and Constructing their Digital Profile

identify common attack patterns across different entities. The synergy of Social Network Analysis with our cybersecurity framework elevates its capabilities, providing a holistic view of threat actor behaviors and connections. By incorporating SNA, our framework contributes to enhanced threat attribution, network visualization, and the proactive defense against the ever-evolving landscape of cyber threats.

2. BUILDING A CYBER PROFILE

Tracking Cyber-Adversaries and Constructing their Digital Profile

The objective while constructing a cyber-profile involves identifying a set of traits that could collectively establish a pattern indicative of an attacker. These traits can derive from an adversary's networking presence, whether within an internal network or across the Internet, and, either en masse or even individually, pose as signatures of their discrete attacking personality. At this point, it is significant to remember our initial hypotheses: adversaries are humans and, as every other human/user with an online presence, they exhibit habits, routines, unique methodologies, goals, and varying levels of attitude.

Dr. Mary Aiken, eminent professor of cyberpsychology and chair of the department of Cyberpsychology at Capitol Technology University, states in her book "The Cyber Effect" [9] that "typical criminal hackers share a set of personality traits". From this line, it is interesting to note not only that she observes how the personality of a person can be reflected in their digital world, but also the definition of a criminal hacker. From her experience as a member of the INTERPOL Global Cybercrime Expert Group she recognizes that offensive adversaries differ, in some ways, from ethical hackers. This can be clarified even more if we consider the targets of known adversaries against the targets of typical ethical hackers, mostly employed within companies specialized, as a whole or individual departments, in providing cybersecurity services. For the first group priorities are recognition, appeal, challenge, economical gain and impact while, for the second group, while these objectives remain pertinent, they are compelled to give priority to contractual obligations. What we can deduct from this single line to assist our study is that in a cybersecurity incident, the target is just as significant as the source even in terms of attribution. As Dr. Aiken adds later on, "Criminals reveal who they are and where they live not just from how they commit their crimes, but also from the locations they choose", a significant statement in the context of cyber geolocation.

Another characteristic that varies between users, as long as cyber adversaries, is their level of experience and familiarity with digital systems. *Practice makes perfect* is an idiom that can be applied to every endeavor associated with the notion of achievement. With this hypothesis as a foundation, we can posit that hackers of higher experience levels will exhibit fewer failed attempts compared to their less experienced counterparts. Moreover, concealed behind every flawlessly executed and undetected intrusion should be numerous unsuccessful attacks that equip adversaries with the necessary insights into potential obstacles they might face when targeting their ultimate objectives. These recorded attempts serve as essential data for the detection, attribution, and prevention of subsequent attacks. Facts have shown that, even minor typographical mistakes [10] can turn into evidence of anomalous and unwanted behavior.

Tracking Cyber-Adversaries and Constructing their Digital Profile

To incorporate these attributes into machine learning algorithms, it is essential to examine how they manifest in technical artifacts. An actual case [11] illustrates the investigation and attribution of the OceanLotus group in connection with a series of phishing attacks. Hakan Tanriverdi, reporter for the Germanic public broadcasting corporation Bayerischer Rundfunk, led an investigation [12] against the group, registered as APT 32 [13] under MITRE ATT&CK framework, which concluded with an important collection of artifacts, like IP addresses, specific ports, passive DNS data and SSL certificates owned and utilized from the Vietnamese group. Their research started small, with simple clues as an IP address, and grew big with continuous investigation. As he explained, “Look for new domains on those Ips. Try to find the pattern”. Their targets exhibited the same level of importance. Having excluded major German companies in the automotive industry, which could be seen as valuable targets globally, they came to the conclusion that OceanLotus “mainly hacks Vietnamese people”, taking into consideration those that could be categorized as domestic targets. In this way, the group was attributed as of Vietnamese origin as well. Finally, we can also observe the importance of mistakes from this story. Research on an SSL certificate used for the group’s phishing websites lead to the discovery of dozens of others. As Tanriverdi reported, “either because they were lazy or they used automation, [OceanLotus] had one certificate for 280 websites”.

The assessment of technical artifacts within a threat actor’s network presence is of utmost importance within the cybersecurity domain. The preceding case study serves as a poignant reminder of the pivotal role these artifacts play in the attribution and mitigation of cyber threats. Technical artifacts, encompassing elements such as IP addresses, specific port usage, passive DNS data, and SSL certificates, function as digital signatures that facilitate the tracing of malicious actors’ activities. These artifacts provide the initial cyber trail and markers that lead to the identification of threat actors and the depiction of their tactics and techniques. Furthermore, a rigorous examination of these digital cues empowers cybersecurity professionals to discern recurring patterns, tactics, and targeted entities, which are instrumental in constructing a comprehensive threat profile. As demonstrated in the OceanLotus investigation, even seemingly trivial details, such as SSL certificates, can unveil an extensive network of nefarious operations. Consequently, the systematic evaluation of technical artifacts not only expedites attribution but also equips organizations with the insights required to fortify their defenses against impending cyber threats, rendering it an indispensable component within contemporary cybersecurity strategies.

In order to efficiently manage and categorize the technical artifacts associated with a threat actor’s network presence, we propose to consolidate the technical artifacts associated with network presences under a unified label known as **network presence**. This approach will be greatly facilitated by leveraging open-source knowledgebases and

repositories, which provide a wealth of information and intelligence on known threat actors, their tools, and infrastructure. This strategic approach will enable us to harness collective knowledge and expertise effectively in our pursuit of cyber threat attribution.

2.1 HUMAN HIDING IN LOGS

The concept of focusing on human characteristics within the aftermath of a cyber incident could be considered, often, overlooked, although the reason for this is sensible. Under the pressure of time, analysts need to concentrate towards more tangible items in order to support their task, like evidence of data exfiltration, malicious installed software, masqueraded processes, etc.

Howbeit, certain experts in the field of cyber security emphasize on how important is taking into account the human factor concealed within logs. Jon DiMaggio, Chief Security Strategist at Analyst1 and former Sr. Threat Intelligence Analyst at Symantec, reminds us in his book, “The Art of Cyberwarfare: An Investigator’s Guide to Espionage, Ransomware, and Organized Cybercrime” [14] to, “[...] keep in mind there’s always someone behind the attack. The important thing to remember about cyberattacks is that real people are behind them, people who have habits and preferences, such as the specific tools or passwords they like to use, the aliases or personas used to create fake accounts, domain registration, and themes in infrastructure names, among many others” [p. 118]. On this premise he also adds that cyber adversaries “[...] will have tools and tactics they favor and frequently use. They also likely have unique behaviors or methods that they use and reuse from one attack to another” [p. 132]. Some of these identifiers, that could be translated into technical artifacts when collected, are what we will name as a hacker’s **toolbox**, meaning their equipment which makes them feel confident, necessary, and trusted resources that would assist their way to a successful hack. Others, which carry a deeper personal significance and don’t significantly impact the intrusion process, can be modified so that they leave their own mark, will be named as **miscellaneous** signatures.

Organizing abstract data from varying separate sources under a threat actor frame holds a substantial degree of significance within our study. Through this approach, Machine Learning and Social Network Analysis algorithms developed for the purpose of carrying out attribution tasks will evaluate connections and patterns in a more efficient and accurate way. DiMaggio describes attributable data as “[...] any data that can provide supporting evidence toward making a valid attribution” [p. 120]. Sources for this data can be logs from local networks which, evidently, suffered a cyberattack, or open sources

Tracking Cyber-Adversaries and Constructing their Digital Profile

from the Internet. “Open-source information can be detailed as finding the identity of a malware author on a hacking forum” [p. 120], denotes the author on this matter.

However, not all artifacts carry equal weight in the attribution process. DiMaggio provided insight into Symantec’s approach of distinguishing Chinese espionage groups from specific software they used. As he describes, the investigation team “created a list of the malware and hacking tools associated with each group. Not all tools used by espionage groups are unique or custom, but Symantec narrowed its list to include only those that it could uniquely attribute to an espionage attacker” [p. 7].

Under the miscellaneous category, an additional intriguing artifact pertains to timestamps. By capturing timeframe during an attack, we can gather information about the adversary’s operational hours, their time zone, and even the required duration to carry out an attack and infiltrate a system. DiMaggio notes on that, “You’ll want to track attacker activities and timeframes by analyzing timestamps on log data associated with the activity. Time-zone analysis – that is, documenting the exact time at which each malicious event took place on your network – can help you track times when the attacker was active” [p.120]. Through this data we understand better the threat actor’s experience and mastery levels, along with potential targets they could strive to overtake.

Concluding with the miscellaneous artifacts which could directly connect a personality to malicious online activities, DiMaggio in his book includes a short story from the Dark Web about an individual with the alias “Wexford”. The user with the referenced username engaged a forum and “claimed he worked for and supported the Suncrypt ransomware gang but he never got paid”. The debate ignited by this particular post culminated in “providing analysts with an interesting insight into the inner working of a Russian organized crime gang” [p. 170]. This example demonstrates how even the most accomplished threat actors can make mistakes which expose their, formerly, well-kept secret techniques and methodologies on online sites.

Favored Tactics, Techniques and Procedures (from now on referenced as TTPs) are among the aspects that should be taken into consideration while building a cyber adversary’s profile. The way an attacker or an APT group prefers to maneuver through systems and execute attack vectors can differ drastically. While building our profile frames, they will take place in a separate category, that of **attack methods**.

2.2 INDICATION OF COMPROMISE BECOMES INDICATION OF COMPROMISER

In the ever-evolving landscape of cybersecurity, the shift from “Indication-of-Compromise” (IoC) to “*Indication-of-Compromiser*” (*IoCer*) marks a pivotal juncture in threat detection, analysis and attribution. As digital threats grow more sophisticated and persistent, the conventional approach of identifying signs that a breach has occurred falls short of providing adequate defense.

Traditionally, the focus of threat detection and mitigation lay in the identification of explicit signs of intrusion, often relying on indicators such as IP addresses, file hashes, and domain reputation. However, the escalating sophistication of malicious actors and the adoption of advanced persistent threat (APT) tactics have rendered this conventional approach insufficient. The *IoCer* paradigm introduces a different perspective, leveraging advanced technical artifacts and behavioral analytics to unveil not just the occurrence of anomalies, but also, the distinct attributes and strategies of the threat actors orchestrating them. By integrating recorded IoCs and conducting a comprehensive qualitative and quantitative analysis of their relevance, our aim is to identify distinctive behavioral patterns among individuals. These patterns should possess the requisite robustness to differentiate their profiles from those of other individuals, whether malicious or benign.

Previous studies have been conducted on this subject. F.Skopi and T.Pahi on their paper, “Under false flag: using technical artifacts for cyber attack attribution” [15] highlighted the significance of assembling technical artifacts within the context of a threat actor profile frame and assessing its distinctiveness based in conjunction with MITRE’s ATT&CK knowledgebase [16] and the Cyber Kill Chain framework [17]. “A prerequisite of cyber attribution is to discover the applied techniques, tools and procedures (TTPs). Based on that, the further goal is to identify the source of certain attacks that leads to the threat actor” [p.2]. Building upon the foundation laid by Skopi and Pahi’s research, the utilization of technical artifacts within the context of a threat actor profile gains even more significance in today’s dynamic cybersecurity landscape.

Their emphasis on assessing the distinctiveness of these artifacts in conjunction with established frameworks and knowledge provides a structured approach to unraveling the complexities of cyberattacks. By delving into TTPs employed in cyber incidents, a crucial steppingstone towards successful cyber attribution is achieved. This process not only aids in attributing attacks but also extends to the identification of the originating threat actors themselves, thereby contributing to a comprehensive understanding of the

Tracking Cyber-Adversaries and Constructing their Digital Profile

evolving threat landscape. As they state, “It is therefore of utmost importance to get it right. An important prerequisite is to know about common attack tools and techniques and understand which traces (typically artifacts) they potentially leave in a victim’s infrastructure (or even elsewhere, like at the premises of cloud providers and other external parties)” [p.1].

Concerning the aforementioned frameworks, we can observe how threat actors function as covert entities, strategically embedding tangible evidence of their operations within each discrete stage of the Cyber Kill Chain. This meticulously structured framework systematically outlines the stages of a cyberattack, providing a comprehensive understanding of intrusion pathways. Ranging from initial reconnaissance and incursion to lateral movement, data exfiltration, exploitation, and eventual impact, each phase retains indelible imprints of the attacker’s techniques, motives, and technical expertise. Analyzing these detectable footprints along the Cyber Kill Chain equips cybersecurity experts with invaluable insights, empowering them to enhance defensive strategies and cultivate a robust digital terrain.

Notably, cybersecurity analysis and attribution frequently result in experts allocating greater emphasis to stages like lateral movement, occasionally overshadowing the importance of reconnaissance. This transition in focus is linked to the realization that although reconnaissance establishes the foundation for an attack, it is during the initial access phase that threat actors’ tactics and sophistication become evident, revealing intricate indications that can uncover their motives and possible consequences. Skopi and Pahi mention in their research “We argue that even in these [preparation] phases [of an attack], when attackers prepare for a complex attack, traces may be left, such as the attempts to buy zero day exploits in the dark net, excessive scanning activities, social engineering attempts and the like” [p.4].

The extent to which hackers might leave traces during the reconnaissance phase of an attack stands as a pivotal concern. As malicious actors commence their offensive operations, they engage in reconnaissance to meticulously gather intelligence, pinpoint vulnerabilities, and chart potential pathways for intrusion. This phase serves as a pivotal juncture where the delicate balance between their covert maneuvers and the digital breadcrumbs they inadvertently leave behind becomes a focal point of analysis. The dynamic interplay between hackers’ stealthy actions and the digital ecosystem’s inherent traceability underpins the complex discussion surrounding the probability of trace presence during this preliminary stage of cyber exploits.

Amidst these complexities, the urgent need for collaborative data sharing from diverse cyber incidents emerges as a key enabler in advancing cyberattack attribution. By pooling together information and insights drawn from a wide array of attacks,

cybersecurity professionals can collectively refine their understanding of attack methodologies, tactics, and techniques. This collaborative approach empowers organizations to recognize patterns that span beyond isolated incidents, shedding light on commonalities in hackers' approaches. This, in turn, amplifies the ability to attribute cyberattacks more accurately by linking shared characteristics and distinctive hallmarks to specific threat actors or groups. Therefore, the practice of exchanging incident data goes beyond the interests of individual organizations, serving as a vital foundation in strengthening the collective cybersecurity landscape against ever more sophisticated and persistent threats.

2.3 THE REVERSE PYRAMIDS OF GOALS

We could say that threat actors represent a heterogeneous collection of individuals motivated by a broad spectrum of motivations. While some of them are motivated by financial gain, seeking to exploit vulnerabilities for profit, others are driven by a sense of curiosity and a desire to explore the digital landscape. Ethical hackers aim to expose and rectify security flaws, advocating for a more secure online environment, while others pursue hacktivism, using their skills to further social or political causes. Ultimately, the motivations behind hacking are as varied as the hackers themselves, making it a complex and multifaceted subset within the field of cybersecurity.

To incorporate this hypothesis into our study, we will explore the applicability of the Expectancy Value Theory of Motivation [18], a psychological framework that seeks to explain why individuals choose to engage in specific actions or pursue particular goals. At its core, this theory posits that people's choices and actions are driven by a combination of two key factors: their expectations for success and their subjective assessment of the value of a particular task or domain. For instance, individuals are more likely to engage in an activity when they believe they can excel in it and when they personally value the activity. The theory further dissects task value into four essential components, including attainment value, intrinsic value, utility value, and cost, each playing a unique role in shaping motivation. Moreover, expectancy-value theory recognizes that these expectations and task values are influenced by a multitude of factors, encompassing individual characteristics such as abilities, beliefs, and self-concepts, as well as external influences like cultural norms and socializing agents. Research has consistently affirmed the distinct yet interconnected nature of expectations for success and task value, underscoring their profound impact on children's academic and achievement-related outcomes. In essence, expectancy-value theory offers valuable

Tracking Cyber-Adversaries and Constructing their Digital Profile

insights into the complex interplay between personal beliefs and values, shedding light on the intricate dynamics of human motivation and decision-making.

Drawing a parallel between expectancy-value theory and the realm of cyber attacker attribution provides a fascinating perspective on the motivations behind a threat actor's actions. Just as individuals' choices and behaviors are influenced by their expectations for success and subjective values in traditional settings, cyber attackers' decisions to engage in malicious activities can be understood through a similar lens. In the context of cyber attacker attribution, their "expectations for success» can be seen as their assessment of the likelihood of evading detection and successfully executing an attack, driven by factors like their technical proficiency and knowledge of security measures. On the other hand, the "task value" for these attackers may relate to their motivations, such as financial gain, ideological beliefs, or competitive advantage, with each of these factors representing a different aspect of the subjective value they assign to their actions. Much like in traditional domains, understanding the interplay between these factors, expectations for success and task values, can shed light on the motivations and decision-making processes of cyber attackers, aiding cybersecurity professionals in identifying and mitigating threats effectively.

As we delve deeper into the intricacies of cyber-attack attribution and the myriad factors shaping the motivations of threat actors, it becomes increasingly evident that a comprehensive understanding of these elements is paramount for advancing our capabilities in tracing and identifying cyber adversaries. Drawing upon the insights derived from expectancy-value theory, we can now conceptualize the intricate interaction between threat actors and the investigative techniques underpinning our cyber attribution efforts.

In the following visual representation, two imposing pyramids are placed side by side, each representing a unique aspect of the digital domain and encapsulating a comparison between threat actors and the robust security measures implemented to safeguard digital systems. The upright pyramid represents the expertise and capabilities of threat actors, reflecting their evolving skills and tactics in the cyber domain. Conversely, the inverted pyramid represents the layers of security measures deployed to safeguard digital systems. This inverted structure underscores the multi-faceted approach organizations adopt to fortify their cyber defenses, incorporating firewalls, encryption, access controls, and intrusion detection systems, among others.

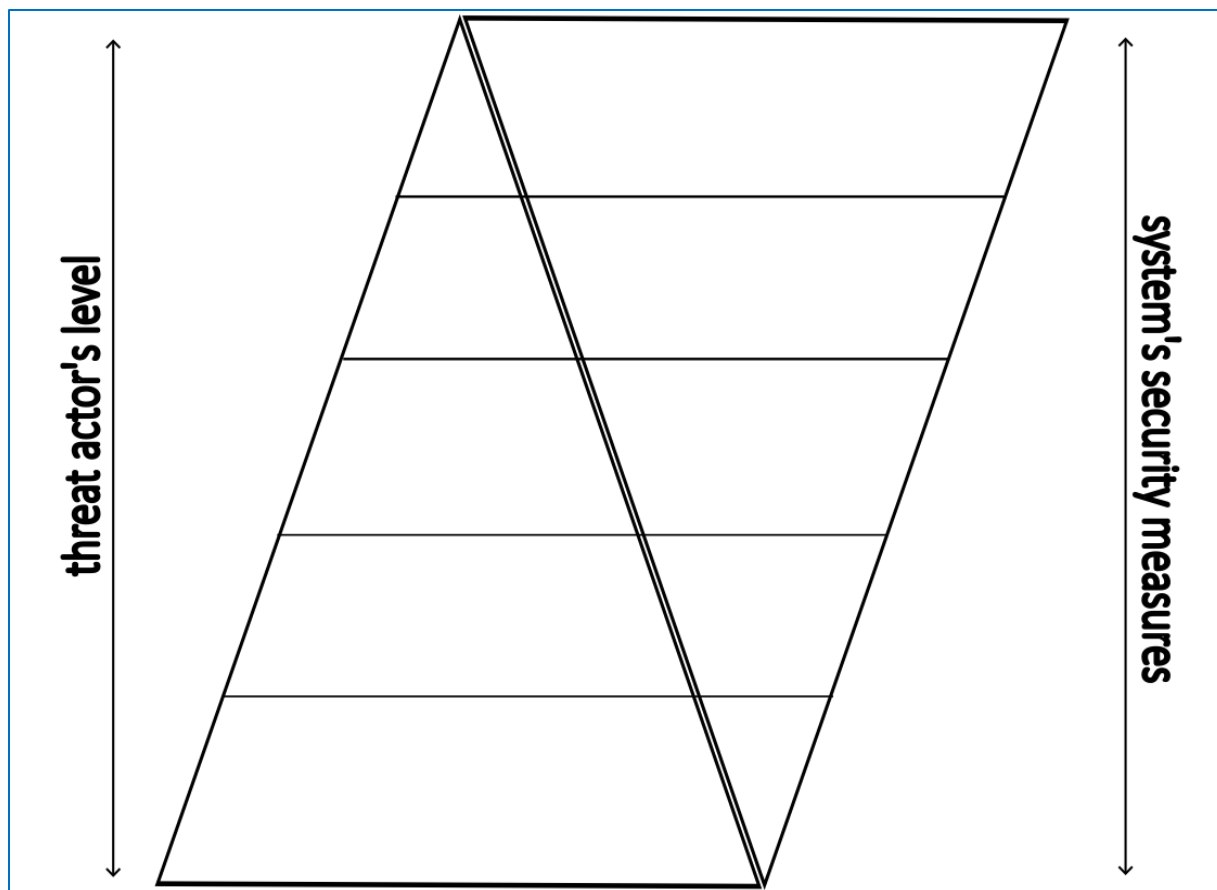


Figure 3: The visual comparison of a threat actor's level and their targeted systems.

The convergence of these two pyramids at their points visually illustrates the dynamic relationship between hackers and security measures. It signifies how hackers with advanced levels of expertise may target and circumvent even the most resilient security defenses.

Concluding, cyber attacker attribution is a complex landscape, mirroring the diverse motivations and behaviors of the threat actors who operate within it. Just as the Expectancy Value Theory of Motivation sheds light on why individuals engage in specific actions, it can offer valuable insights into the decision-making processes of cyber threat actors. By examining their expectations for success and the subjective value they place on their actions is vital for cybersecurity professionals seeking to identify and mitigate threats effectively. As we continue to delve into the intricacies of cyber attribution, it becomes evident that this comprehensive understanding is essential for advancing our capabilities in tracing and identifying cyber adversaries.

A key insight emerges when we consider the expertise-driven motivation of cyber attackers in relation to their target selection. The convergence of these factors provides a compelling illustration of how attackers strategically match their capabilities with their selected targets. For cyber adversaries, their “expectations for success” are intrinsically tied to their technical proficiency, knowledge of security measures, and awareness of potential vulnerabilities. In this context, the “task value” embodies their motivations, which may include financial gain, ideological beliefs, or competitive advantage, with each motive representing a different aspect of the subjective value they attribute to their actions. Much like skilled individuals pursuing challenges in traditional domains, cyber adversaries strategically assess the risk-reward balance and engage in activities where their expertise matches the target’s security posture. Understanding this interplay between expectations for success and task value unveils the motivations and decision-making processes of cyber attackers, empowering cybersecurity professionals to identify and counter threats with greater precision.

2.4 AN OVERVIEW OF A THREAT ACTOR’S PROFILE

The cyber actor’s profile serves as a curated repository, capturing an extensive array of technical artifacts and behavioral indicators that offer deep insights into the actor’s strategies, toolset, and overarching presence within the intricate network landscape. Through the systematic documentation and categorization of diverse elements, ranging from the utilization of specific tools to subtle patterns of behavior, the profile provides a multifaceted perspective on the actor’s modus operandi and operational footprint within digital environments. This wealth of information enables us to glean invaluable intelligence regarding the actor’s methodologies and objectives, laying the groundwork for effective threat detection, attribution, and mitigation strategies.

A potential threat actor’s profile could be consisted by the following parts:

- **Toolbox:** The profile documents the cyber actor’s expansive arsenal of tools, while cataloging an extensive array of software utilities specifically selected to suit their objectives. Within this repository lie both widely recognized tools essential to offensive operations and bespoke creations crafted to circumvent defenses and achieve particular goals. Prominently featured are common enumeration tools such as Nmap, renowned for its versatility in network scanning and reconnaissance, alongside a repertoire of popular fuzzers including Gobuster, Feroxbuster, and FFuF, renowned for their effectiveness in probing web applications for vulnerabilities. Additionally, the profile encompasses a diverse

Tracking Cyber-Adversaries and Constructing their Digital Profile

selection of open-source tools sourced from public repositories like GitHub, showcasing the actor's resourcefulness in leveraging community-driven innovation to augment their capabilities. Notably, the inclusion of uncommon self-crafted tools underscores the actor's unparalleled sophistication and adaptability, showcasing their ability to develop specialized solutions tailored to exploit unique vulnerabilities and evade traditional security measures. This meticulous curation of tools within the profile provides our framework with invaluable insights into the actor's capabilities and tactics.

- **TTPs (Tactics, Techniques, and Procedures):** Technical artifacts cataloged within the profile unveil the cyber actor's strategic maneuvers throughout the entirety of the cyber kill chain. Spanning from the initial stages of reconnaissance through weaponization and delivery, these artifacts document the actor's tactics, providing a comprehensive blueprint of their operational methodology. By delineating the actor's progression through each phase of the kill chain, the profile offers to our framework invaluable insights into the evolving threat landscape.
- **Miscellaneous:** The profile further incorporates miscellaneous technical artifacts, ranging from unusual usernames and filenames to distinctive strings and triggered errors. Despite their seemingly innocuous nature, these subtle anomalies serve as pivotal identifiers, offering invaluable insights into the behaviors and methodologies of threat actors. By recording and arranging these nuanced indicators, the profile enhances the granularity of threat intelligence, enabling our framework to attribute malicious activities with precision. Through the comprehensive analysis of these miscellaneous artifacts, we can identify patterns and adversary tactics.
- **Network presence:** Finally, the profile offers a comprehensive snapshot of the cyber actor's network presence, presenting a wealth of vital information derived from their IP address. Delving beyond mere identification, the profile documents details such as subnet allocation, Autonomous System Numbers (ASNs), and behavioral markers sourced from open-source knowledge bases. These contextual insights provide a deeper understanding of the actor's infrastructure and affiliations, enabling our framework to draw connections and discern potential threat actor affiliations. By correlating activities with contextual information, we can strengthen the attribution process, shedding light on the broader ecosystem of malicious activities and facilitating proactive defense strategies. This holistic approach to network analysis equips our framework with the intelligence needed to anticipate threats and identify adversary affiliations.

Tracking Cyber-Adversaries and Constructing their Digital Profile

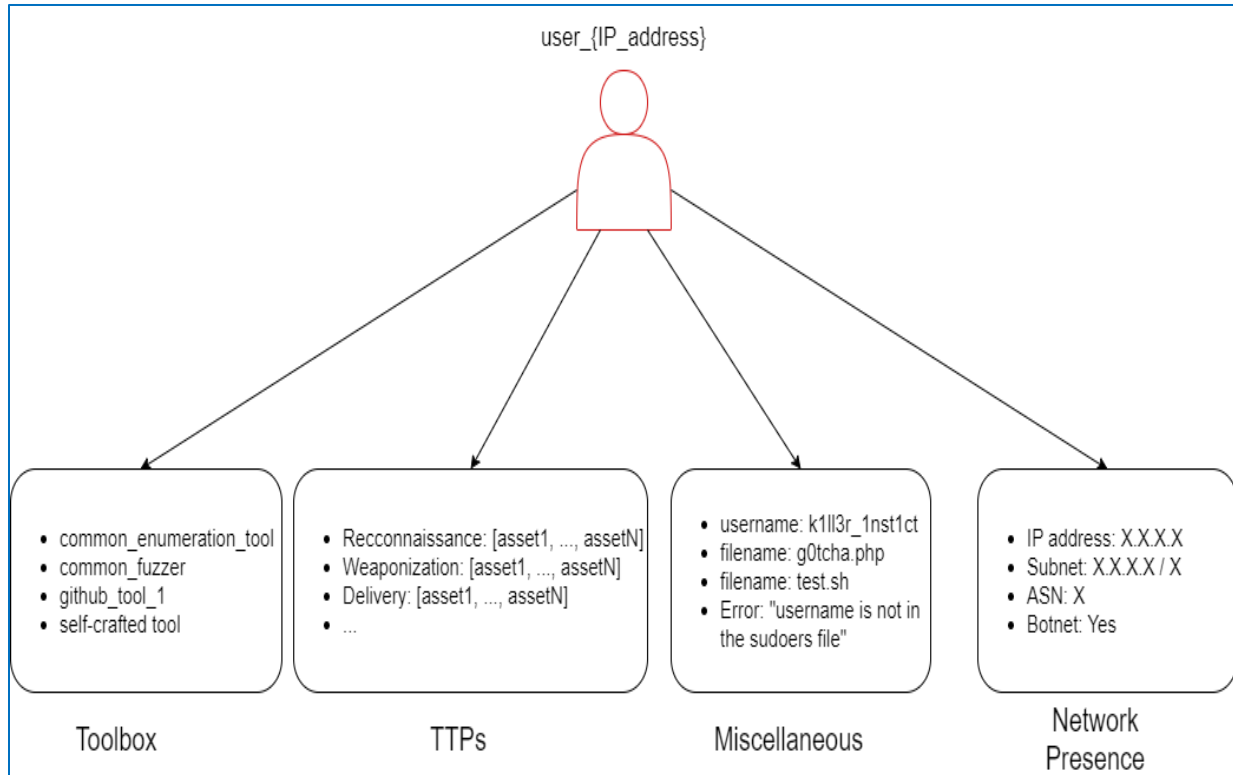


Figure 4: Visual representation of a threat actor's cyber profile.

3. THE FRAMEWORK ARCHITECTURE

The architecture of our framework integrates unsupervised machine learning and social network analysis to enhance the understanding and identification of malicious actors. At its core, the framework operates in three distinct stages, each designed to collect, organize, and analyze data from various sources efficiently.

In the initial stage, our framework employs a comprehensive approach to data collection, tapping into a wide array of sources renowned for their rich reservoirs of security-related information. These sources encompass pivotal components of network defense, including Web Application Firewalls (WAFs), which analyze and filter HTTP traffic for potential threats, and Intrusion Detection Systems (IDS), which monitor network traffic for suspicious activities or anomalies. Additionally, the framework harnesses the collective intelligence of open-source knowledge bases, leveraging repositories of threat intelligence and historical attack data contributed by cybersecurity experts and communities worldwide. This multi-sourced data is then systematically funneled into a centralized hub orchestrated by a properly designed data collector. Here, the collected data undergoes thorough parsing, where it is meticulously dissected and restructured into a standardized format. This process is pivotal in ensuring uniformity and compatibility across disparate data streams, facilitating seamless integration and analysis in subsequent stages of the framework. By centralizing and standardizing the diverse influx of data, our framework establishes a solid foundation for robust and insightful threat analysis.

Transitioning to the second stage of our framework, the parsed data undergoes a diligent structuring process to form comprehensive threat actor profiles, each uniquely identified as “user_{user's_IP_address}”. These profiles serve as repositories of invaluable insights, housing a diverse array of information organized into distinct dictionaries. Within these dictionaries, the networking data category captures critical details pertaining to the user’s interactions within the network, encompassing elements such as IP addresses, subnets, Autonomous System Numbers (ASNs), and behavioral indicators gleaned from open-source knowledge bases. Concurrently, the attack methods dictionary meticulously catalogs offensive strategies employed by the user, encompassing a spectrum of data including IDS signatures indicative of Tactics, Techniques, and Procedures (TTPs), logs delineating utilized tools, and miscellaneous information such as unique Uniform Resource Identifiers (URIs) and triggered error messages. Subsequently, these curated profiles find their place within a centralized database, where they coexist alongside other recorded profiles, forming an extensive repository suitable for comparative analysis and further investigation.

Tracking Cyber-Adversaries and Constructing their Digital Profile

In the third and final stage, our framework utilizes the power of unsupervised machine learning methodologies, notably relying on cosine similarity, to undertake a comparative analysis of a user's profile attributes against a compendium of previously recorded profiles. This analytical process allows for the identification of nuanced similarities and deviations, serving as a basis for discerning patterns and trends within the vast expanse of threat data. Moreover, the architecture integrates social network analysis techniques to visually depict relationships between profiles, along with their weight. By constructing a graph where nodes represent individual profiles and edges signify their respective similarities, the framework provides a comprehensive visualization of the underlying connections within the threat landscape.

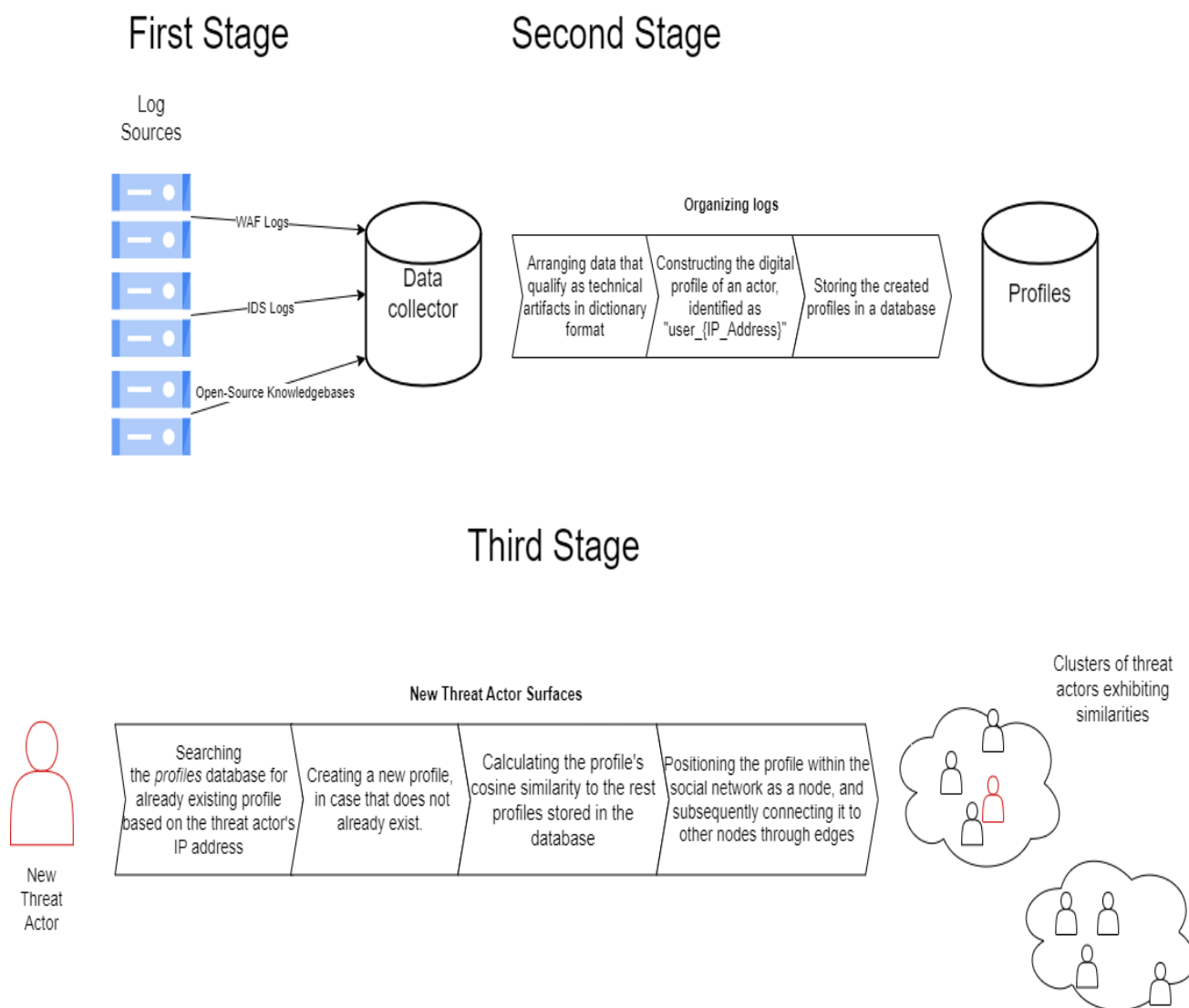


Figure 5: Framework's architecture.

Tracking Cyber-Adversaries and Constructing their Digital Profile

In conclusion, the framework architecture presented embodies a sophisticated and comprehensive approach to cyber threat actor attribution. By leveraging a conscientious process of data collection, parsing, and analysis, the framework enables the identification and differentiation of distinct threat actors operating within complex network environments. Through the integration of unsupervised machine learning techniques and social network analysis, it offers a robust solution for analyzing diverse technical artifacts and behavioral patterns, ultimately empowering cybersecurity professionals with actionable intelligence for threat attribution and mitigation strategies. Furthermore, the framework's adaptability and scalability ensure its relevance in addressing the evolving nature of cyber threats, while ongoing research and development efforts promise continued enhancements to its capabilities. In a landscape characterized by persistent cyber threats, the framework stands as a cornerstone in the ongoing battle to safeguard digital assets and preserve the integrity of digital ecosystems.

4. INITIAL STAGE OF THE FRAMEWORK - GATHERING LOGS FROM DIVERSE SOURCES

The initial stage of our framework is a pivotal juncture, setting the groundwork for subsequent phases. This chapter is dedicated to the process of collecting logs from a diverse array of sources, with the primary aim of transforming these data points into technical artifacts stored within MongoDB collections. These logs will serve as the raw materials that for the succeeding stages of a comprehensive framework aimed at attributing attacks to threat actors through unsupervised machine learning. The dataset utilized in this research, sourced from the website of the United States Military Academy West Point [19] and originating from a red team exercise conducted by the National Security Agency (NSA), is the foundation of our analysis. This dataset comprises Apache web access and error logs in addition to Snort Intrusion Detection System (IDS) logs, offering a substantial reservoir of data for analysis and investigation.

Logs, often viewed as mere digital footprints, play an indispensable role in cybersecurity investigations. They provide a chronological record of events, allowing analysts to trace the steps of attackers and gain insights into their tactics, techniques, and procedures (TTPs). In the context of our framework, these logs serve as the technical artifacts that enable us to construct a holistic picture of cyber threat actors. By gathering logs from diverse sources, such as web servers and intrusion detection systems, we set the stage for uncovering the hidden patterns and anomalies that can serve as indicators of a threat actor's profile, enhancing our framework for cyber adversary attribution. Our focus on the NSA-sourced dataset further enhances the authenticity and relevance of our research, given its origins in a high-stakes red team exercise.

Technical artifacts serve as crucial elements in understanding malicious activities within network and Internet environments. Within the framework's context, the following technical artifacts have the potential to play a pivotal role in clarifying the tactics, techniques, and infrastructure of threat actors. From IP addresses and domain names to network traffic patterns and cryptographic keys, each artifact offers invaluable insights into the threat landscape. By leveraging these artifacts, our framework enables proactive threat detection, attribution, and the formulation of targeted mitigation strategies. A list of possible assets that could be utilized as technical artifacts:

- IP addresses (both source and destination)
- MAC addresses
- Domain names
- URLs and URIs

Tracking Cyber-Adversaries and Constructing their Digital Profile

- Port numbers
- Autonomous System Numbers (ASNs)
- SSL certificates
- HTTP headers
- User-Agent strings
- Cookies
- File hashes (MD5, SHA-1, SHA-256)
- Registry keys (Windows)
- Event logs (Windows)
- System files and directories
- Process names and IDs
- Command-line arguments
- Registry modifications
- Network traffic patterns (e.g., frequency, volume)
- Packet payloads
- Protocol anomalies (e.g., unusual packet sizes, malformed packets)
- DNS queries and responses
- Geolocation data
- Timestamps and time intervals
- Error messages and alerts
- API calls and usage patterns
- Cryptographic keys and certificates
- Shell scripts and batch files
- Configuration files (e.g., .conf, .xml)
- Database queries and access logs

As we commence the process of log collection and organization, we recognize that this initial stage is the foundation upon which our framework is constructed. The data contained within these logs will serve as the essential raw material, analogous to puzzle pieces awaiting assembly, for the subsequent phases of our project. Through this technical endeavor, we advance toward our ultimate objective—unraveling the intricacies of cyberattacks and achieving precise attribution to the corresponding threat actors.

4.1 GATHERING LOGS FROM THE WEB APPLICATION

The following example of Apache [20] web server access log entries demonstrate the wealth of information contained within these logs concerning incoming requests to a web server. This log entry is structured into several key components:

```
Nov 11 09:41:43 www logger: 10.2.25.83 - - [11/Nov/2011:09:41:43 -0500] "HEAD / HTTP/1.0" 302 -  
Nov 11 09:42:05 www logger: 10.2.25.83 - - [11/Nov/2011:09:42:05 -0500] "GET /Iliketosendtrafficyou HTTP/1.0" 302 232  
Nov 11 09:42:06 www logger: 10.2.25.83 - - [11/Nov/2011:09:42:06 -0500] "GET /admin.php HTTP/1.0" 302 218
```

Figure 6: An example of Apache web server access log structure.

The log entry commences with a timestamp and server hostname, enabling the tracking of when the request took place and on which server. In this case, the event occurred on November 11th, 2011, at 09:42:07 in the Eastern Standard Timezone on a server identified as ‘www.’

Subsequently, the IP address of the remote client, denoted as ‘10.2.25.83,’ is recorded. Nevertheless, the presence of two hyphens (‘- -’) following it indicates the absence of user identity and authentication information for the client.

HTTP requests made by clients are described in the log, including the request method (e.g., “GET” and “HEAD”), the requested URL, and the HTTP protocol version. This information illustrates how clients interact with the server.

HTTP status codes, such as ‘302’ in this example, communicate how the server handled the request. The status code serves as a communication tool between the server and the client, guiding the latter on how to proceed.

The log entry often concludes with the size of the server’s response in bytes. This figure aids in assessing data transfer and load times, offering valuable performance insights.

From the Apache access log lines presented in Figure 3, we can extract key insights regarding the attribution of a cyber attacker, related to their personality traits and preferred tactics. In this scenario, our adversary exhibited distinctive characteristics by making a request featuring an unusual string, “Iliketosendtrafficyou” which was logged as a URI. Furthermore, they endeavored to access a commonly used endpoint, “admin.php”. Additionally, as part of their modus operandi, they employed a request method involving the HTTP header HEAD.

Tracking Cyber-Adversaries and Constructing their Digital Profile

This analysis underscores the importance of examining not just the technical aspects of cyberattacks but also their behavioral and methodological dimensions. To parse these logs and extract all necessary information, we have crafted a dedicated Python parser. This parser functions by reading log lines from an access log file, employing a regular expression pattern for parsing, and then inserting the extracted data into a MongoDB database. It is also able to handle cases where multiple log entries belong to the same user by updating existing user documents or creating new ones. Additionally, it provides error handling for log lines that cannot be parsed. Key parts of the code will be analyzed further.

Connection with the Mongo Database:

```
# Establish a connection to a MongoDB instance running locally on port 27017
db_client = pymongo.MongoClient("mongodb://localhost:27017/")

# Select the 'project' database in MongoDB
db = db_client["project"]

# Choose the 'westpoint_users' collection within the 'project' database
access_log = db["westpoint_users"]
```

Our MongoDB will act as a Central Data Collector would in a cloud infrastructure. Given our expectations of how the logs should be structured within the database, the parser is tasked with the responsibility of establishing a connection with the database and the designated collection for storing web application logs.

Segmenting log line details using regular expressions:

```
# Define a regular expression pattern for parsing log lines
log_format = r'(\S+)\s+(\S+)\s+(\S+)\s+(\S+)([^\:]+):\s+(\S+)\s+\s+\s+\s+\s+([\[\]]+)\s+"([\^"]+)"\s+(\d+)\s+(\S+)'
```

This regular expression pattern matches and extracts specific information from each log line. It captures various components of a log entry such as IP address, timestamp, HTTP method, URI, response code, and bytes sent.

Associating the extracted data with corresponding variables:

```
# Check if the line matches the pattern and the IP address is not ':::1'
if matches and (matches.group(5) != ':::1'):
    # Extract relevant information from the matched log line
    ip = matches.group(5)
    timestamp = matches.group(6)
    http_method = matches.group(7).split()[0]
    uri = matches.group(7).split()[1] if len(matches.group(7).split()) > 1 else ""
    response_code = matches.group(8)
    bytes_sent = matches.group(9)
```

If a log line conforms to the pattern and the IP address is not “:::1”, the parser will extract the necessary information, categorizing them as IP, Timestamp, HTTP Method, URI, Response Code, and Byte Count. Log lines with the IP “:::1” will be excluded as it is only a special notation in Internet Protocol version 6 (IPv6) that represents the loopback address, similar to “127.0.0.1” in IPv4, and does not contribute to the cyber attacker attribution process.

Aggregating the data into a dictionary:

```
# Create a dictionary containing the extracted data
data = {
    'IP': ip,
    'timestamp': timestamp,
    'http_method': http_method,
    'uri': uri,
    'response_code': response_code,
    'bytes': bytes_sent
}
```

Collectively, the essential information will be organized and stored as a dictionary, subsequently being inserted into the designated collection. This structured data will reside as a subdocument named “logs” within the corresponding user’s document, identifiable through the “user_{source_IP_address}” field.

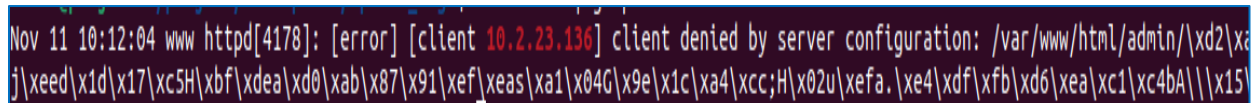
Storing the data for subsequent analysis:

```
# Check if a user document with the same IP address already exists in MongoDB
existing_user = access_log.find_one({'user': f'user_{ip}'})

if existing_user:
    # Update the existing user document by appending the log entry to the 'logs' list
    access_log.update_one(
        {'_id': existing_user['_id']},
        {'$push': {'logs': data}}
    )
else:
    # Create a new user document with the IP address and the log entry
    new_user = {
        'user': f'user_{ip}',
        'logs': [data]
    }
    access_log.insert_one(new_user)
```

The parser will search for the presence of a document with the field “user_{current_source_IP_address}” already exists. If none is found, it will generate a new document with the corresponding field and insert the “data” dictionary in the subdocument “logs”. In the event that such a document already exists, it will update that document with the data retrieved from the extraction and association process.

A similar process will be performed for the error logs from an Apache web server. The following image demonstrates a line from the error logs file generated during the red team engagement.



```
Nov 11 10:12:04 www httpd[4178]: [error] [client 10.2.23.136] client denied by server configuration: /var/www/html/admin/\xd2\xaj\xeed\x1d\x17\xc5H\xbf\xdea\xd0\xab\x87\x91\xef\xead\x04G\x9e\x1c\xa4\xcc;H\x02u\xefa.\xe4\xdf\xfb\xd6\xea\xc1\xc4BA\\\x15'
```

Figure 7: An example of Apache web server error log structure.

Apache error log entries like the one provided serve as a means to document and communicate issues encountered by the server during its operation. In a typical error log entry, we will find timestamps, indicating when the error occurred, information about the server process responsible for generating the error, and details regarding the nature of the error.

The error message itself provides valuable information about the nature of the problem. For instance, «client denied by server configuration» suggests that access to a particular resource was denied due to server configuration settings. Lastly, part of the log entry shows the URL or path to the resource that the client was trying to access.

Tracking Cyber-Adversaries and Constructing their Digital Profile

What can be initially start observed from these figures is how dissimilarities between two distinct users can manifest in technical artifacts. In the case of the user with the source IP address 10.2.25.83 shorter URIs were documented, with an absence of errors, and even the presence of a string with a more personalized significance. The second user, identified by the source IP address 10.2.23.136, had longer payloads recorded, alongside a few instances of errors. Later, these two users will automatically be assigned to distinct clusters created following the implementation of SNA and cosine similarity algorithms within our framework.

To collect the required technical artifacts and integrate them into our cyber attacker profiling procedure, we've devised a specialized Python parser. In a same way with the Apache access logs, for the error logs the parser will read the generated file line by line while matching them with the corresponding regular expression pattern.

Segmenting log line details using regular expressions:

```
# Define the regular expression pattern
pattern = r'^(\w+\s+\d+\s+\d+:\d+:\d+)\s+(\w+)\s+(\w+)\s+(\d+)\s+:\s+[error]\s+\[client\s+(\[d.\]+\)\]\s+(.+)'
```

The pattern is designed to capture various components of an error log entry, including timestamp, server information, process information, client IP address, and error message. The extracted pieces of information will be collected and stored in their respective variables.

Associating the extracted data with corresponding variables:

```
if matches:
    timestamp = matches.group(1)
    server = matches.group(2)
    process = matches.group(3)
    pid = matches.group(4)
    ip = matches.group(5)
    splitted_error = matches.group(6).split(": ")
    error_message = splitted_error[0]
    uri = splitted_error[1]
```

In a similar with the previous implementation for the access logs, the now structured data will be organized and stored within a dictionary, ensuring that it remains well-organized and accessible for subsequent processing and analysis.

Aggregating the data into a dictionary:

```
# Create a dictionary containing the extracted data
data = {
    'IP': ip,
    'timestamp': timestamp,
    'http_method': http_method,
    'uri': uri,
    'response_code': response_code,
    'bytes': bytes_sent
}
```

The final step of this process involves transferring the populated dictionary into the corresponding MongoDB collection, where it will be securely stored for further utilization and analysis in the cyber attacker attribution course of action.

Storing the data for subsequent analysis:

```
# Check if a user document with the same IP address already exists
existing_user = access_log.find_one({'user': f'user_{ip}'})

if existing_user:
    # Update the existing user document by appending the log entry to the log list
    access_log.update_one(
        {'_id': existing_user['_id']},
        {'$push': {'errors': data}}
    )
```

The parser will undertake an examination to ascertain whether a user document associated with the same IP address already resides within the MongoDB collection. The objective is to append the data to an existing document that corresponds to the previously parsed access log file. This approach ensures that data continuity is maintained, and new insights are integrated into the existing user profiles.

Log parsing tools play a valuable role in monitoring and detecting potential threats within network and system logs. Our preceded examples of parsers are able to provide insights into user behavior, detect anomalies, and identify patterns that may indicate malicious activity against web applications, when taking part of the complete framework of cyber attacker profiling. However, it is important to emphasize that log parsing alone is insufficient for conclusive attacker attribution.

While log parsing can provide valuable data for incident detection and initial investigation, it should be integrated into a broader cybersecurity strategy that encompasses a range of security measures and tools. Effective cyber attacker attribution requires a multifaceted approach that combines various techniques and sources of information to establish a clearer picture of the threat landscape and the identities behind

Tracking Cyber-Adversaries and Constructing their Digital Profile

cyberattacks. Thus, proceed by applying a comparable log parsing approach to logs generated by an Intrusion Detection System (IDS) installed on the target.

The example documents provided in Figures 8 and 9 are the outcome of processing extensive collections of Apache web access and error logs. These collections encompass a broader range of logs, and for the sake of this study, we have selected and included only two illustrative examples, corresponding to the IP addresses 10.2.25.83 and 10.2.23.136. These documents, as stored in a MongoDB collection, serve as representative samples showcasing the structured data resulting from our parsing process, and they are retained for further analysis and reference in our cyber attacker profiling study. Closing this section, Appendix 1 showcases the schema of the collection.

Tracking Cyber-Adversaries and Constructing their Digital Profile

```
[
  {
    _id: ObjectId("649ed69ac366c3bca72d95ed"),
    user: 'user_10.2.25.83',
    logs: [
      {
        IP: '10.2.25.83',
        timestamp: '11/Nov/2011:09:41:43 -0500',
        http_method: 'HEAD',
        uri: '/',
        response_code: '302',
        bytes: '-'
      },
      {
        IP: '10.2.25.83',
        timestamp: '11/Nov/2011:09:42:05 -0500',
        http_method: 'GET',
        uri: '/Iliketosendtrafficyou',
        response_code: '302',
        bytes: '232'
      },
      {
        IP: '10.2.25.83',
        timestamp: '11/Nov/2011:09:42:06 -0500',
        http_method: 'GET',
        uri: '/admin.php',
        response_code: '302',
        bytes: '218'
      },
      {
        IP: '10.2.25.83',
        timestamp: '11/Nov/2011:09:42:06 -0500',
        http_method: 'GET',
        uri: '/%32%47%32%47%78%13%99%76%78%13%99%76',
        response_code: '302',
        bytes: '229'
      },
      {
        IP: '10.2.25.83',
        timestamp: '11/Nov/2011:09:42:06 -0500',
        http_method: 'GET',
        uri: '/staff.aspx?%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76',
        response_code: '302',
        bytes: '380'
      },
      {
        IP: '10.2.25.83',
        timestamp: '11/Nov/2011:09:42:07 -0500',
        http_method: 'GET',
        uri: '/admin/',
        response_code: '302',
        bytes: '215'
      }
    ]
  }
]
```

Figure 8: Corresponding Apache web access and error logs for the user with IP address 10.2.25.83.

Tracking Cyber-Adversaries and Constructing their Digital Profile

```
[
  {
    _id: ObjectId("649ed69bc366c3bca72d9633"),
    user: 'user_10.2.23.136',
    logs: [
      {
        IP: '10.2.23.136',
        timestamp: '11/Nov/2011:10:12:02 -0500',
        http_method: 'GET',
        uri: '/admin/%11%8B%AF%1D%F9%22%92%AAjf%17%8Fd%B9zk%80%B8%85yL%FF%13%CE%F1%CE%02M%D1P%89%EC%AC%3E?%25x%B21%94%E9%D4%CF%EF%B2%F6',
        response_code: '302',
        bytes: '478'
      },
      {
        IP: '10.2.23.136',
        timestamp: '11/Nov/2011:10:12:03 -0500',
        http_method: 'GET',
        uri: '/admin/%CB%B9%85E9%CEz%BD%06%A2%F9Z%FC%B4%B7%B0%06n%9CAI%16%BA%04%C6%EA%19%E1T%94%20%D7HV%3C%85c%A0(%E4%8C%B6Cb%B3I%A1%11hE%0C%05w%ACjm',
        response_code: '302',
        bytes: '428'
      },
      {
        IP: '10.2.23.136',
        timestamp: '11/Nov/2011:10:12:04 -0500',
        http_method: 'GET',
        uri: '/admin/%D2%A6%8C%9A%D7%9A%B8%92%08%3C%A0%3E%20%25M%D4V%90%94%A2J&%D8%C6%B62%F0%97j%EEed%1D%17%C5H%BF%DEa%D0%AB%0F%B6*%0C%5FE%7ZZ%19%0A%FAz%D0%BD+n4%F8%06I',
        response_code: '302',
        bytes: '217'
      }
    ],
    errors: [
      {
        IP: '10.2.23.136',
        timestamp: 'Nov 11 10:12:04',
        process: 'httpd',
        error_message: 'client denied by server configuration',
        uri: '/var/www/html/admin/\\xd2\\xa6\\x8c\\x9a\\xd7\\x9a\\xb8\\x92\\|b<\\xa0> %M\\xd4V\\x90\\x94\\xa2J&\\xd8\\xc6\\xa1\\x04G\\x9e\\x1c\\xa4\\xcc;H\\x02u\\xefa.\\xe4\\xdf\\xfb\\xd6\\xea\\xc1\\xc4bA\\|\\|\\|x15\\x02Q,\\|xa1.\\x0f\\xb6*\\x0c\\xc5'
      }
    ]
  }
]
```

Figure 9: Corresponding Apache web access and error logs for the user with IP address 10.2.23.136.

4.2 GATHERING LOGS FROM THE INTRUSION DETECTION SYSTEM

Leveraging Intrusion Detection System (IDS) logs holds significant importance within the context of the cyber attacker attribution process for several compelling reasons. These logs provide a detailed record of network activities, including the source IP addresses, timestamps, and the nature of detected events, which serve as invaluable indicators, aiding in retracing the steps of potential attackers. By analyzing these logs, security professionals can establish a timeline of the intrusion and identify patterns or anomalies indicative of specific attack techniques or attacker behavior.

Furthermore, IDS logs offer contextual information through rule IDs, classifications, and external references to known vulnerabilities or exploits. This context assists in understanding the attacker's motivations, tactics, and potential targets, aiding in the development of a more comprehensive attribution profile.

The log example presented in the following figure is sourced from the Snort IDS [21] that was installed on the targeted system. This log offers a practical exemplification of the type of information that Intrusion Detection Systems can provide, providing insights on the detection and monitoring capabilities they bring to the cyber attacker attribution process.

```
[**] [1:1301:12] WEB-PHP admin.php access [**]  
[Classification: Attempted Information Leak] [Priority: 2]  
11/11-09:35:41.891872 10.2.25.83:48086 -> 154.241.88.201:80  
TCP TTL:61 TOS:0x0 ID:32954 IpLen:20 DgmLen:167 DF  
***AP*** Seq: 0xEDAB0A0A Ack: 0xDE7AA9F4 Win: 0x5C TcpLen: 32  
TCP Options (3) => NOP NOP TS: 11569848 223925202  
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-1032][Xre
```

Figure 10: An example of Snort IDS log structure.

The initial segment of the log entry corresponds to the signature or rule that initiated the alert. This signature offers essential insights into the nature of the event that was detected, furnishing details about the specific type of activity or behavior that led to the alert. This information is instrumental in understanding the context and potential threat associated with the event and enables us to swiftly categorize and comprehend the nature of the security incident.

Tracking Cyber-Adversaries and Constructing their Digital Profile

The Classification field within the log entry is a pivotal component that serves the purpose of not only indicating the event's type or category but also provides valuable information regarding its severity or priority level. This additional layer of information is indispensable for accurately gauging the significance of the event and its potential impact on the security posture.

The rest lines concern the networking traffic between the actor and the system. We can observe a timestamp of the time the event occurred, the source IP address and the source port of the actor, as well as the system's IP address and port (which port is also indicative of the targeted service). Also, information about the TCP (Transmission Control Protocol) packet's attributes, including "Time To Live (TTL)", "Type of Service (TOS)", "Identification (ID)", "IP Length (IpLen)", "Datagram Length (DgmLen)", and the "Don't Fragment (DF) flag" and more data regarding the TCP packet are provided. Lastly, we have references denoted as Xref, which point to external resources pertinent to the event. These references serve as hyperlinks to websites or databases containing additional public knowledge regarding vulnerabilities or exploits associated with the detected event. They serve as valuable resources for in-depth investigation and assessment, while offering insights into the specific security risks posed by the event.

To extract the requisite information from these logs, we have developed a dedicated Python parser. In the forthcoming paragraphs, we will provide a comprehensive explanation of the parser's design, functionality, and how it facilitates the extraction and organization of crucial data from these logs. This parser serves as a fundamental component in our cyber attacker attribution process, enabling the systematic analysis of security events and the identification of pertinent patterns, indicators, and attack methods. As before, the final organized data will be stored in a separate collection of our MongoDB.

Segmenting log line details using regular expressions:

```
# Regular expressions to extract the desired fields
pattern_signature = r"\[.*\*\] \[(\d+:\d+:\d+)\] (.*) \[.*\*\]"
pattern_priority = r"\[Priority: (\d+)\]"
pattern_ip = r"(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):(\d+)"
pattern_tcp_info = r"Seq: (0x\w+) Ack: (0x\w+) Win: (0x\w+) TcpLen: (\d+)"
```

As the structure of Snort IDS logs is notably more intricate compared to that of the Apache web server, the associated regular expression patterns are correspondingly more complex. To ensure optimal results, we adopt a segmented approach. Specifically, we divide the parsed log lines from the log file into four distinct sections, each stored in its corresponding variable: "pattern_signature", "pattern_priority", "pattern_ip", and "pattern_tcp_info". This strategic segmentation allows for efficient handling of the diverse

Tracking Cyber-Adversaries and Constructing their Digital Profile

data contained within the Snort IDS logs, ultimately leading to more precise and comprehensive results in our cyber attacker attribution efforts.

Associating the extracted data with corresponding variables:

```
- - - - -
# Regular expressions to extract the desired fields
pattern_signature = r"[\*\*\] \[(\d+:\d+:\d+)\] (.*) \[\*\*\]"
pattern_priority = r"\[Priority: (\d+)\]"
pattern_ip = r"(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):(\d+)"
pattern_tcp_info = r"Seq: (0x\w+) Ack: (0x\w+) Win: (0x\w+) TcpLen: (\d+)"

# Extract signature and priority
signature_match = re.search(pattern_signature, log_entry)
signature_id = signature_match.group(1) if signature_match else None
signature_desc = signature_match.group(2) if signature_match else None

priority_match = re.search(pattern_priority, log_entry)
priority = priority_match.group(1) if priority_match else None

# Extract IP and port information
ip_ports = re.findall(pattern_ip, log_entry)
source_ip, source_port = ip_ports[0] if ip_ports else (None, None)
destination_ip, destination_port = ip_ports[1] if len(ip_ports) > 1 else (None, None)

# Extract TCP information
tcp_match = re.search(pattern_tcp_info, log_entry)
seq_number = tcp_match.group(1) if tcp_match else None
ack_number = tcp_match.group(2) if tcp_match else None
win_size = tcp_match.group(3) if tcp_match else None
tcp_len = tcp_match.group(4) if tcp_match else None
```

Likewise, the process of categorization demands a more comprehensive effort. IDS logs yield a wealth of information that involves thorough organization and arrangement. These logs contain a broader spectrum of data points and details that we need to systematically structure for subsequent analysis and interpretation.

Aggregating the data into a dictionary:

```
# Return the extracted fields as a dictionary
data = {
    "signature_id": signature_id,
    "signature_desc": signature_desc,
    "priority": priority,
    "source_ip": source_ip,
    "source_port": source_port,
    "target_ip": destination_ip,
    "target_port": destination_port,
    "seq_number": seq_number,
    "ack_number": ack_number,
    "win_size": win_size,
    "tcp_len": tcp_len
}
```

The data segments are stored and structured within a dictionary. This approach allows us to neatly organize and encapsulate the requisite information for further analysis. Each key within the dictionary corresponds to a specific data category, ensuring that the data remains well-organized and readily accessible for subsequent processing and investigation.

Storing the data for subsequent analysis:

```
existing_user = snort_log.find_one({'user': f'user_{source_ip}'})
if existing_user:
    # Update the existing user document by appending the log entry to the log list
    snort_log.update_one(
        {'_id': existing_user['_id']},
        {'$push': {'logs': data}}
    )
else:
    # Create a new user document with the IP address and the log entry
    new_user = {
        'user': f'user_{source_ip}',
        'logs': [data]
    }
    snort_log.insert_one(new_user)
```

Finally, the organized data will be inserted into the MongoDB collection, following the same procedure as with the Apache access and error log data. This step ensures that the data is securely stored and readily available for in-depth analysis and the cyber attacker attribution process.

Tracking Cyber-Adversaries and Constructing their Digital Profile

The exemplar document presented in Figure 11 is the result of parsing an extensive collection of Snort IDS logs. These logs encompass a wide array of security events and incidents. For the purpose of this study, we have chosen to include a specific example, corresponding to the IP address 10.2.23.136. This document stands as an illustrative specimen exemplifying the organized data derived from our parsing process. Closing, Appendix 2 demonstrates the schema of the collection.

Tracking Cyber-Adversaries and Constructing their Digital Profile

```
[
  {
    _id: ObjectId("649ee3bc5f30fef3703dcc8f"),
    user: 'user_10.2.23.136',
    logs: [
      {
        signature_id: '1:12592:3',
        signature_desc: 'SMTP ClamAV recipient command injection attempt',
        priority: '1',
        source_ip: '10.2.23.136',
        source_port: '60613',
        target_ip: '7.204.241.161',
        target_port: '25',
        seq_number: '0x274BE122',
        ack_number: '0xD94AD8A1',
        win_size: '0x418',
        tcp_len: '20'
      },
      {
        signature_id: '1:12592:3',
        signature_desc: 'SMTP ClamAV recipient command injection attempt',
        priority: '1',
        source_ip: '10.2.23.136',
        source_port: '49399',
        target_ip: '7.204.241.161',
        target_port: '25',
        seq_number: '0xCB36D5BC',
        ack_number: '0xB8EAB21E',
        win_size: '0x418',
        tcp_len: '20'
      },
      {
        signature_id: '1:12592:3',
        signature_desc: 'SMTP ClamAV recipient command injection attempt',
        priority: '1',
        source_ip: '10.2.23.136',
        source_port: '39826',
        target_ip: '7.204.241.161',
        target_port: '25',
        seq_number: '0x937E4AB7',
        ack_number: '0xB5DA159D',
        win_size: '0x418',
        tcp_len: '20'
      },
      {
        signature_id: '119:7:1',
        signature_desc: '(http_inspect) IIS UNICODE CODEPOINT ENCODING',
        priority: '3',
        source_ip: '10.2.23.136',
        source_port: '38103',
        target_ip: '154.241.88.201',
        target_port: '80',
        seq_number: '0x96E1096C',
        ack_number: '0x6E45187E',
        win_size: '0xB7',
        tcp_len: '32'
      }
    ]
  }
]
```

Figure 11: Corresponding Snort IDS logs for the user with IP address 10.2.23.136.

4.3 COLLECTING PUBLIC INFORMATION FROM OPEN-SOURCE KNOWLEDGE BASES

Gathering information from open-source knowledge bases like AlienVault [22], AbuseIPDB [23], VirusTotal [24], and MITRE ATT&CK is an essential part for the cyber threat intelligence and attribution efforts. These public information sources provide a wealth of data and insights that can aid cybersecurity professionals in identifying and understanding malicious activities and the threat actors behind them.

AlienVault, now part of AT&T Cybersecurity, is a prominent cybersecurity platform that offers comprehensive threat intelligence capabilities. Apart from providing threat intelligence feeds and reports concerning established threats and vulnerabilities, it also offers an advanced API integration for IP tracking. With AlienVault's API integration for IP tracking, cybersecurity professionals can access a real-time stream of threat data related to IP addresses. This includes information about IP reputation, historical behavior, geographic location, and associated threats. The integration also enables the evaluation patterns of suspicious activity and assess the severity of potential threats, allowing for more informed decision-making in incident response.

Within our framework, we will employ the AlienVault OTX API to assess the IP addresses associated with potential threat actors. This integration serves as a straightforward illustration of how open-source knowledge bases can be effectively utilized. It's important to note that the actual red team engagement occurred within an internal network environment, resulting in the logging of only internal network IP addresses. Integrating with the AlienVault OTX (Open Threat Exchange) API we code automates the process of querying and retrieving information about specific IP addresses. For the dataset in scope the gathered information includes the IP address itself, its ASN (Autonomous System Number), reputation, and whether it is marked as private.

Open-source intelligence will be systematically employed to enrich each profile generated from the previously parsed logs. To facilitate this integration, a dedicated HTTP POST request must be dispatched to the AlienVault OTX API. This segment of our codebase bears the responsibility of skillfully constructing the request and subsequently retrieving a JSON response pertaining to the specific IP address under consideration.

Upon receiving the JSON response, we have the capability to extract any valuable pertinent information. These include:

- **Autonomous System Number (ASN):** This data reveals the specific Autonomous System Number associated with the queried IP address. It

Tracking Cyber-Adversaries and Constructing their Digital Profile

serves as a key identifier of the network to which the IP address belongs. It aids in the attribution process by revealing potential affiliations or connections and marking its origin.

- **Reputation:** This metric offers insights into the reputation of the IP address in question. It provides valuable context regarding whether the IP address has been associated with malicious activities or if it maintains a reputable status within the cyber landscape.
- **Private IP Status:** In our specific dataset, this attribute assumes significance as it indicates whether the IP address in question is categorized as private. This distinction holds relevance within the context of our internal network-centric study. For a dataset which would include public IP addresses as well, it would allow us to distinguish between internal network addresses and publicly routable ones, contributing to a more precise attribution process.

These extracted details contribute to a more comprehensive and informed profiling of potential threat actors, enabling us to assess their affiliations, historical behavior, and the context of their presence within our network environment. This holistic approach enhances our cyber attacker attribution capabilities and strengthens our overall cybersecurity posture.

```
# Initialize an empty dictionary called 'ip_data' to store information about the IP address.
ip_data = {"IP": [], "ASN": [], "Reputation": [], "Private": []}
try:
    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        # Checking if the response status code is 200, indicating a successful request.
        data = response.json()
        # Parsing JSON data from the response.

        ip_data["IP"].append(ip_address)
        ip_data["ASN"].append(data['asn'])
        ip_data["Reputation"].append(data['reputation'])
    else:
        if 'IP is private.' in response.json().values():
            ip_data["IP"].append(ip_address)
            # Mark the IP address as private by adding "True" under the "Private" key.
            ip_data["Private"].append("True")
        else:
            print(f"Error occurred while querying the API for IP address {ip_address}.")
            print(response.json())
except:
    # Catch any exceptions that may occur during the request and print an error message.
    print(f"Error occurred while making the request for IP address {ip_address}:", e)
```

Tracking Cyber-Adversaries and Constructing their Digital Profile

The central functionality concerning the analysis of IP addresses is implemented within the code featured in the figure above. This section of the code performs two key functions:

Firstly, it initiates the retrieval of relevant information from the data received via communication with the AlienVault OTX API. This data serves as a critical source of insights into the queried IP address.

Subsequently, the extracted data is systematically organized and stored within a dictionary structure. This dictionary is carefully designed to encapsulate key-value pairs. Each key corresponds to a specific data attribute, while its associated value contains the fitting information.

- **IP:** This key serves as a clear identifier of the IP address currently under examination, providing unambiguous context for analysis.
- **ASN (Autonomous System Number):** This attribute designates the specific ASN to which the IP address in question is assigned.
- **Reputation:** The reputation key assesses and records the reputation status of the IP address.

In the concluding phase of the code's execution, it conducts a verification process to ascertain whether the IP address in question belongs to a private network. If this condition is met, a distinctive identifier, namely the string «IP is private», is included within the HTTP response. This string acts as a definitive indicator, signaling that the IP address is part of an internal or private network infrastructure.

This operational feature plays key role within the broader context of our data profiling process, a comprehensive commentary of which will be presented in the subsequent chapter. It is an integral component of our data profiling framework, facilitating the systematic collection and analysis of critical data attributes associated with IP addresses.

In the forthcoming chapter, we will embark on a detailed exploration of our data profiling process. This in-depth examination will encompass the methodologies, techniques, and tools employed to construct comprehensive profiles of potential threat actors. The insights gleaned from these profiles are instrumental in enhancing our cyber attacker attribution capabilities, providing a more profound understanding of their behaviors and motivations.

5. MIDDLE STAGE OF THE FRAMEWORK - ASSEMBLING DISTINCT TECHNICAL ARTIFACTS UNDER INDIVIDUAL PROFILES

Having successfully gathered an array of technical artifacts from diverse sources in the previous chapter, we move forward with our threat actor attribution framework. The middle stage takes center stage, as we transition from raw data to the assembly of distinct technical artifacts. Our goal is to construct individual profiles for threat actors, a task that necessitates the synthesis of technical artifacts, all meticulously organized during this pivotal phase.

The data we have collected, classified as Networking Data, Attack Methods, and Miscellaneous information, demands a methodical organization. With python3 programming, we classify and format this data into three sub-collections. These sub-collections will play a vital role in constructing comprehensive threat actor profiles within our MongoDB database. The categorization process is as follows:

1. **Networking Data:** This subset provides insights into the digital infrastructure employed by threat actors, including IP addresses, domains, and network-related information;
2. **Attack Methods:** Here, we document the tactics and techniques utilized by threat actors in their cyberattacks, offering a detailed view of their modus operandi;
3. **Miscellaneous:** This category contains supplementary information that enriches our analysis, including timestamps, contextual details, and any additional data that enhances our understanding of the threat landscape.

By assembling these data subsets, we set the stage for the final phase of this chapter—constructing threat actor profiles. This crucial task involves merging the three subcollections into a unified MongoDB collection, aptly named as "user_{IP_address}" This intimate structured data repository will serve as the cornerstone for the advanced analytics and attribution processes that follow in our framework.

5.1 BUILDING THE WEB LOGS PART

Cyber-attacker's attribution is a critical aspect of cybersecurity that involves identifying the source or origins of a cyber-attack. Logs from an Apache web server are requisite in this process. In the provided code segment, you may refer at Appendix 3, the code is focused on collecting web server access logs and extracting valuable information for potential attribution efforts.

To begin, the code retrieves Apache web access and error logs from a MongoDB database, where these logs are stored. Each log entry is associated with a user, and the code iterates through these logs to gather essential data. It starts by extracting the user's unique identifier, often referred to as the "user_{IP_address}". This identifier plays a key role in tracing and associating actions with individual users since it serves as the consistent marker for their respective profiles.

Next, the script collects Uniform Resource Identifiers (URIs) from the user's web access logs. URIs provide information about the resources accessed on the web server. By compiling a set of URIs associated with a user, analysts can gain insights into the user's online activities and the specific web pages or resources they interacted with.

Furthermore, the script conducts a more in-depth analysis of the composition of these URIs. It dissects each URI into its constituent parts, such as paths, parameters, and queries. This detailed deconstruction of URIs provides valuable insights into the user's behavior and intentions. Path information reveals the specific web pages visited, parameters provide additional context or data passed in the URL, and queries shed light on multiple operations.

In summary, this section of the code demonstrates the importance of web server access logs in the context of cyber-attack attribution. It collects user-specific access data, including URIs and their components, providing analysts with a wealth of information to investigate and attribute cyber threats effectively. These logs serve as a critical resource for understanding user behavior and identifying potential security incidents.

5.2 BUILDING THE NETWORKING DATA PART

The provided code segment is a part of the script responsible for building networking-related data based on a user's ID and IDS (Intrusion Detection System) logs. This data can be crucial for cyber-attack attribution efforts, as it provides insights into the user's network activity and potential security threats.

The code begins by extracting the IP address from the user's ID. This IP address is typically embedded within the user ID and can be valuable in identifying the network source of various activities. By splitting the user ID and isolating the IP address, the script prepares to gather information related to this IP.

```
def build_networking_data(user_id, ids_logs):
    ip = user_id.split('_')[1]
    ip_info = check_ip(ip)

    ports = set()
    targets = set()
    try:
        for item in ids_logs:
            ports.add(item['Source_port'])
            target = item['Target_ip'] + ':' + item['Target_port']
            targets.add(target)
    except (TypeError):
        pass
    data = {
        'IP_info': ip_info,
        'Ports': list(ports),
        'Sum_of_ports': len(ports),
        'Targets': list(targets)
    }

    return data
```

Next, the script collects data about ports and targets. It iterates through the provided IDS logs, which contain information about network activities, including source ports and target IP addresses with associated target ports. The script compiles a set of unique source ports and creates target identifiers by combining the target IP and port. These two sets, “ports” and “targets”, serve as valuable indicators of network behavior.

The “ports” set provides insights into the ports used for network communication, which can be indicative of the type of services or protocols involved. This information can be useful for identifying potential vulnerabilities or suspicious network patterns.

Tracking Cyber-Adversaries and Constructing their Digital Profile

The “targets” set, on the other hand, stores combinations of target IP addresses and ports. These combinations represent the destinations of network connections initiated by the user. Analyzing this data can reveal communication patterns and potentially highlight interactions with malicious or suspicious hosts.

In summary, this code segment plays a crucial role in building networking-related data for cyber-attack attribution. It extracts IP information, compiles lists of source ports, and creates target identifiers. This data helps analysts understand the user’s network activity and aids in identifying potential threats or anomalous behavior within the network. It is a fundamental step in the process of attributing cyber-attacks to specific sources or entities.

5.3 BUILDING THE ATTACK METHODS PART

The provided code contains two functions that contribute in cyber-attack attribution by collecting and processing IDS (Intrusion Detection System) logs.

The “collect_ids_logs(user_id, db)” function focuses on retrieving IDS logs specific to a given user from a MongoDB database. It takes a user identifier “(user_id)” and a database connection (“db”) as inputs. The function first retrieves IDS logs from the «snort_logs» collection in the database. It then attempts to find a document in the collection that matches the provided user ID. If a matching document is found, the function proceeds to extract detailed IDS log data from the ‘logs’ field of the document. For each log entry, it collects information such as signature description, priority, source port, target IP address, and target port. This information is organized into dictionaries and added to the “data” list. In case no matching document is found or if there are any exceptions (such as KeyError or TypeError), the function sets “data” to indicate that the document is empty or that there was an error.

Tracking Cyber-Adversaries and Constructing their Digital Profile

```
def collect_ids_logs(user_id, db):
    ids_logs = db["snort_logs"]

    user_entry = ids_logs.find_one({'user': user_id})

    data = []

    try:
        if user_entry and 'logs' in user_entry:
            for log in user_entry['logs']:
                signature_desc = log['signature_desc']
                priority = log['priority']
                src_port = log['source_port']
                target_ip = log['target_ip']
                target_port = log['target_port']

                to_append = {
                    'Signature_description': signature_desc,
                    'Priority': priority,
                    'Source_port': src_port,
                    'Target_ip': target_ip,
                    'Target_port': target_port
                }
                data.append(to_append)
            else:
                data = 'Empty document'
    except (KeyError, TypeError):
        data = 'Empty document'

    return data
```

The “build_attack_methods(ids_logs)” function is responsible for building a dictionary that contains information about IDS signatures and priority levels. It starts by initializing two empty sets: “ids_signatures” and “priority_levels”. These sets are used to ensure that only unique IDS signature descriptions and priority levels are collected from the provided IDS logs. The function then iterates through the “ids_logs” list, attempting to add the “Signature_description” and “Priority” values of each log entry to their respective sets. This ensures that duplicate signatures or priority levels are not included. Finally, the function returns a dictionary (“data”) that includes lists of IDS signatures and priority levels. This data is valuable for categorizing and analyzing the attack methods observed in the logs.

Tracking Cyber-Adversaries and Constructing their Digital Profile

```
def build_attack_methods(ids_logs):  
  
    ids_signatures = set()  
    priority_levels = set()  
    #dns_logs = set()  
  
    try:  
        for item in ids_logs:  
            ids_signatures.add(item['Signature_description'])  
            priority_levels.add(item['Priority'])  
    except(TypeError):  
        pass  
    data = {'IDS_signatures':list(ids_signatures),'Priority_levels':list(priority_levels)}  
    return data
```

5.4 BUILDING THE MISCELLANEOUS PART

The provided code segments are integral to the script's functionality in handling errors and miscellaneous data within web server access logs, contributing significantly to the comprehension of web server activity and potential issues.

In the first code snippet, the script initializes an empty dictionary called "errors" with two empty lists, "URIs" and "Messages". It then proceeds to check if the document contains an "errors" field and if there are any error entries within it. If errors are detected, the script iterates through these error entries, extracting and appending the "uri" (Uniform Resource Identifier) and "error_message" values to the corresponding lists within the "errors" dictionary.

This code plays a crucial role in capturing details about errors encountered during web server access. The "errors" dictionary serves as a repository for storing URIs associated with errors and their corresponding error messages.

```
if 'errors' in document and document['errors']:  
    for log in document['errors']:  
        errors["URIs"].append(log['uri'])  
        errors["Messages"].append(log['error_message'])
```


Tracking Cyber-Adversaries and Constructing their Digital Profile

Moving on to the second code snippet, the script constructs a dictionary named “miscellaneous”, which contains various pieces of information extracted from web server access logs. It computes the following key elements:

- **Sum_of_errors:** This field calculates the total number of error messages by determining the length of the “Messages” list within the errors dictionary;
- **Errors:** This field includes the entire errors dictionary, encompassing URIs and error messages related to web server access errors;
- **Sum_of_queries:** This field calculates the total number of queries by determining the length of the “queries” list;
- **Queries:** This field contains a list of queries extracted from the web server access logs;
- **Params:** This field comprises a list of parameters extracted from the web server access logs;
- **Paths:** This field holds a list of paths extracted from the web server access logs.

```
#build miscellaneous
miscellaneous = {'Sum_of_errors':len(errors["Messages"]),'Errors':errors,
'Sum_of_queries':len(queries),'Queries': list(queries),
'Params': list(params), 'Paths': list(paths)}
:
```

The miscellaneous dictionary serves as a comprehensive summary of various facets of web server access, such as errors, queries, parameters, and paths. It acts as a valuable resource for gaining insights into the overall activity of the web server and identifying potential issues or patterns that may emerge.

In essence, these code segments are instrumental in collecting and structuring data related to errors and miscellaneous information present in web server access logs. This consolidation of information enhances the framework’s capacity to analyze web server activity comprehensively and pinpoint potential security incidents or anomalies.

5.5 BUILDING THE FINAL PROFILE

This part of the code is responsible for creating and updating user profiles based on the collected data from various sources, such as web server access logs, IDS logs, and miscellaneous information. It oversees the storage and systematic arrangement of

Tracking Cyber-Adversaries and Constructing their Digital Profile

this data within a MongoDB database, preparing it for subsequent analysis and future reference.

```
#create new profile
existing_user = profiles.find_one({'user': user_id})

if existing_user:
    # Update the existing user document by appending the log entry to the log list
    profiles.update_one(
        {'_id': existing_user['_id']},
        {'$push': {'networking_data': networking_data,
                  'attack_methods': attack_methods,
                  'miscellaneous': miscellaneous}}
    )
else:
    new_user = {
        'user': user_id,
        'networking_data': networking_data,
        'attack_methods': attack_methods,
        'miscellaneous': miscellaneous
    }
    profiles.insert_one(new_user)
```

The script begins by attempting to find an existing user profile in the “profiles” collection of the MongoDB database using the user’s unique identifier, “user_id”. If an existing user profile is found, it indicates that the user has had previous interactions or activities recorded. In this case, the script updates the existing user document.

The update process involves appending the newly collected data to the user’s profile. Specifically, it appends the “networking_data”, “attack_methods”, and “miscellaneous” dictionaries to their respective fields within the user’s profile document. This update mechanism allows for the continuous aggregation of data over time as the user interacts with the system or experiences various network events.

However, if the script does not find an existing user profile, it assumes that this is a new user or there has been no previous data recorded for the user. In this scenario, it creates a new user profile document in the “profiles” collection.

The new user profile includes the following fields:

- **user**: This field stores the user’s unique identifier, “user_id”;
- **networking_data**: This field contains networking-related data gathered from IDS logs and other sources;
- **attack_methods**: This field includes information about IDS signatures, priority levels, and web presence;

Tracking Cyber-Adversaries and Constructing their Digital Profile

- **miscellaneous:** This field consolidates miscellaneous data, such as error information, query details, parameters, and paths.

In summary, this section of the code is responsible for managing user profiles within the MongoDB database. It checks for the existence of a user profile and either updates it with new data or creates a new profile if the user is new to the system. This approach ensures that a comprehensive record of user activity and associated data is maintained, facilitating further analysis and attribution of network-related events or cyber-attacks.

The user profile associated with the IP address 10.2.25.83, which is detailed in Figure 12, serves as a representative illustration of the expended format for a threat actor's profile (for the respecting schema, you may refer to Appendix 4). It does not only exemplify the anticipated scheme, but also embodies a comprehensive template for how information should be systematically compiled and organized. This profile serves as a blueprint for the systematic aggregation of information, aligning it in a manner conducive to the subsequent application of machine learning techniques, for the context of this study Cosine Similarity, and the utilization of graph theory, for the context of this study Social Network Analysis.

Tracking Cyber-Adversaries and Constructing their Digital Profile

```
{
  _id: ObjectId("64de87628689a2d8a8b41e40"),
  user: 'user_10.2.25.83',
  networking_data: {
    IP_info: {
      IP: [ '10.2.25.83' ],
      ASN: [],
      Reputation: [],
      Private: [ 'True' ]
    },
    Ports: [ '48086' ],
    Sum_of_ports: 1,
    Targets: [ '154.241.88.201:80' ]
  },
  attack_methods: {
    IDS_logs: {
      IDS_signatures: [ 'WEB-PHP admin.php access' ],
      Priority_levels: [ '2' ]
    },
    Web_logs: {
      URIs: [
        '/admin.php',
        '/admin/',
        '/Iliketosendtrafficyou',
        '/%32%47%32%47%78%13%99%76%78%13%99%76%',
        '/',
        '/staff.aspx?%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%'
      ],
      Paths: [
        '/admin.php',
        '/admin/',
        '/staff.aspx',
        '/Iliketosendtrafficyou',
        '/',
        '/%32%47%32%47%78%13%99%76%78%13%99%76%'
      ],
      HTTP_methods: [ 'HEAD', 'GET' ],
      Response_codes: [ '302' ]
    }
  },
  miscellaneous: {
    Sum_of_errors: 0,
    Errors: { URIs: [], Messages: [] },
    Sum_of_querries: 2,
    Queries: [
      '',
      '%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%'
    ],
    Params: [ '' ],
    Paths: [
      '/admin.php',
      '/admin/',
      '/staff.aspx',
      '/Iliketosendtrafficyou',
      '/',
      '/%32%47%32%47%78%13%99%76%78%13%99%76%'
    ]
  }
}
```

Figure 12: The complete profile of the user with IP address 10.2.25.83.

Tracking Cyber-Adversaries and Constructing their Digital Profile

The goal of this formation is to enable the precise construction of clusters and networks that represent user activity. By adhering to this structured approach, we position ourselves to derive valuable insights from the data, which is instrumental in facilitating the accurate generation of these clusters and activity networks through the utilization of advanced analytical methodologies. The database's informational schema is demonstrated above.

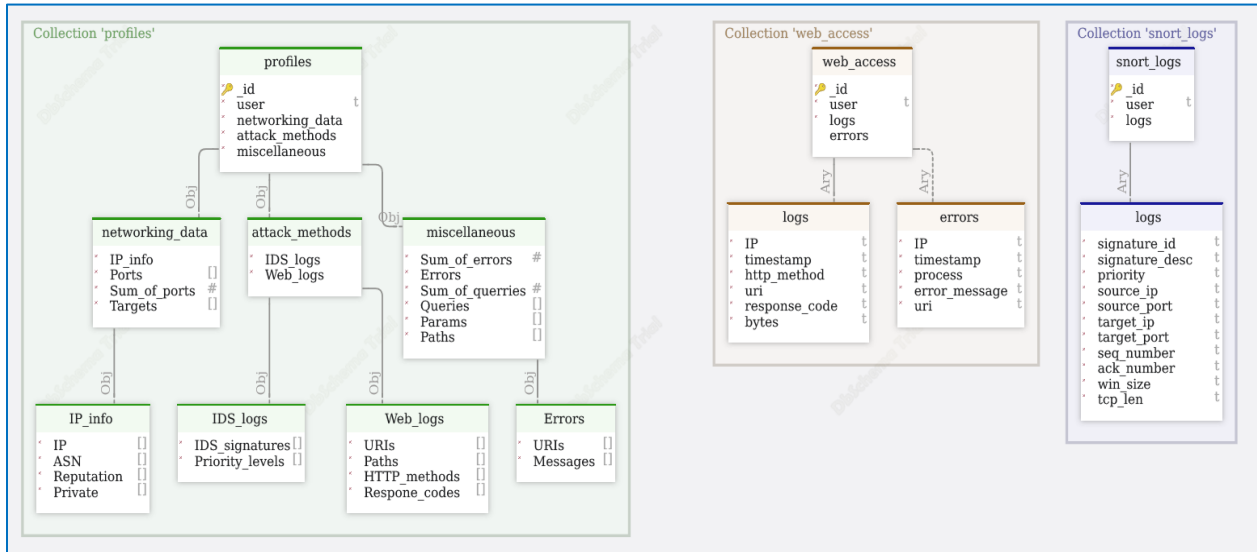


Figure 13: The schemas of the "web_access", "snort_logs" and the, concluded, "profiles" databases.

6. THIRD STAGE OF THE FRAMEWORK – IMPLEMENTATION AND TESTING

In this chapter, we enter the conclusive phase of our framework, where we assess the relational aspects of the scrupulously constructed threat actor profiles. Our primary focus lies in the quantitative measurement of profile similarities using cosine similarity - a pivotal metric for calculating resemblance. Subsequently, we utilize the capabilities of Social Network Analysis (SNA) to translate these computed similarities into a comprehensive visual graph.

Cosine similarity serves as our quantitative tool to gauge the degree of resemblance between profiles. Each threat actor profile is represented as a vector within a high-dimensional space. The cosine similarity metric quantifies the proximity between these vectors, facilitating the identification of relationships and potential collaborations among threat actors. This mathematical approach allows us to uncover previously hidden connections within our dataset.

The visualization element brings us to Social Network Analysis (SNA), a technique that transforms the computed similarities into a comprehensible graph. This graph representation illuminates the intricate network of relationships among threat actors, providing a visually intuitive means of understanding their connections. Furthermore, we augment the framework's functionality by integrating it with a Flask [25] server. This integration enhances accessibility by allowing users to interact with the SNA graph and facilitates specific searches within the MongoDB collection housing the threat actor profiles. This technical achievement adds a layer of usability, making our framework a comprehensive solution for cyber threat analysis and intelligence. In essence, this chapter represents the culminating stage of our framework.

6.1 DATA VECTORIZATION WITH TF-IDF

In the context of our research, we employ the Term Frequency-Inverse Document Frequency (TF-IDF) [26] technique to convert textual information extracted from cyber-attack profiles into a numerical format. TF-IDF vectorization is a popular technique in natural language processing (NLP) and information retrieval that helps convert text documents into numerical vectors suitable for machine learning algorithms. In TF-IDF vectorization, each document is transformed into a vector, where each dimension corresponds to a unique term in the entire corpus. The term frequency (TF) measures how often a term appears in a document, while the inverse document frequency (IDF)

Tracking Cyber-Adversaries and Constructing their Digital Profile

measures how important a term is across a collection of documents. The idea is that common words like "the" and "and" are given lower importance, while rare and meaningful words are given higher importance. This helps in capturing the distinctive features of each document for tasks like text classification, clustering, and information retrieval. In summary, TF-IDF assigns higher values to terms that are frequent in a particular document but rare across the entire corpus, effectively highlighting terms that are discriminative and relevant to that document.

In Python's scikit-learn library, the "TfidfVectorizer" module provides an easy and efficient way to perform TF-IDF vectorization on a collection of text documents. This process can be initiated by creating an instance of the TfidfVectorizer class, where you have the flexibility to define numerous parameters such as tokenization strategies, the removal of stop-words, and considerations for n-grams, among other options. Following this setup, you can apply the vectorizer to your training data, using the fit method to construct the TF-IDF model. Once this model is established, you can effortlessly transform your text documents into TF-IDF vectors utilizing the transform method.

The TfidfVectorizer module proves to be exceptionally valuable in the preparation of text data for deployment in machine learning algorithms. Not only does it convert text into a numerical format, but it also adeptly manages various preprocessing tasks. Its flexibility allows to customize the vectorization process to suit specific task and data, making it a versatile and indispensable component in text analytics and machine learning projects.

Within the domain of attributing cyber-attacks to their source, the code section dedicated to "Text Data Vectorization" and its utilization of "TF-IDF vectorization" assumes a pivotal role. This segment is instrumental in the process of converting textual data extracted from cyber-attack profiles into a numerical format suitable for subsequent analysis.

```
# Vectorize the combined text list (TF-IDF vectorization)
vectorizer = TfidfVectorizer(max_features=1000)
profile_vectors = vectorizer.fit_transform(combined_text_list)
```

For our framework, we perform text data vectorization by employing TF-IDF vectorization techniques, utilizing a maximum of 1000 features. The resulting TF-IDF vectors are then stored in the variable denoted as "profile_vectors". This vectorization process paves the way for subsequent analysis, including the computation of cosine similarity. In the upcoming section, we will examine how cosine similarity quantifies the

likeness between profiles by assessing their TF-IDF vectors. This measure relies on an assessment of their respective TF-IDF vectors, allowing us to gain deeper insights into the extent of likeness or dissimilarity among these profiles.

6.2 CALCULATING AFFINITY BETWEEN CYBER PROFILES WITH COSINE SIMILARITY

Cosine similarity is a metric used to measure the similarity between two vectors in a multi-dimensional space. It is particularly popular in natural language processing and information retrieval for comparing the similarity between text documents. The cosine similarity between two vectors is calculated as the cosine of the angle between them, which ranges from -1 (completely opposite) to 1 (identical), with 0 indicating orthogonality (no similarity). In the context of text data, vectors typically represent the TF-IDF or word embeddings of documents. Cosine similarity is beneficial because it not only considers the magnitude of the vectors but also their orientation, making it robust for comparing text documents.

Scikit-learn provides also a convenient module called “cosine_similarity” to compute cosine similarities between sets of data points. You can use it by passing two sets of vectors, for example our TF-IDF vector derived from the textual descriptions of the cyber-attack profiles which was showcased in the previous section, to the function, and it will return a matrix of cosine similarity scores, where each entry (i, j) represents the cosine similarity between the i th and j th vectors. This module simplifies the process of measuring document similarity, making it efficient and straightforward for various text mining and recommendation tasks.

In practice, cosine similarity is valuable for applications such as document retrieval, clustering similar documents, and recommendation systems. This measure is indispensable for assessing the similarity or dissimilarity between textual data, thereby facilitating well-informed decisions. It stands as a foundational concept in the field of text analytics and assumes a central role in numerous NLP and machine learning processes.

By applying this feature in our framework, we extract a symmetric and square similarity matrix, with each entry (i, j) representing the cosine similarity between the TF-IDF vectors of profiles i and j . The diagonal entries (i.e., when i equals j) represent the similarity of a profile to itself, which is always 1.


```
# Calculate cosine similarity between profile vectors  
similarity_matrix = cosine_similarity(profile_vectors)
```

Cosine similarity values range from -1 to 1. A cosine similarity of 1 indicates that two vectors are identical in direction, meaning the profiles are highly similar. A value of 0 suggests no similarity, while a value of -1 implies that the vectors are diametrically opposed, signifying dissimilarity. The cosine similarity matrix is often used in conjunction with a similarity threshold. In this code, a threshold of 0.4 is used. Nevertheless, the threshold can be fine-tuned according to specific requirements. Profiles that exhibit similarity scores surpassing this threshold are regarded as adequately similar to justify the establishment of a network graph edge between them. This threshold allows for the identification of meaningful connections while filtering out less relevant similarities.

The similarity matrix and threshold are essential components for constructing the network graph. Edges are created between profiles that surpass the similarity threshold, forming a visual representation of connections between cyber-attackers based on their textual profiles. This graph can then be analyzed further to identify clusters, central nodes, or patterns that aid in attribution efforts. Subsequent sections will provide a detailed presentation of this network creation procedure, along with a graphical display of the final clusters created for the dataset in scope.

6.3 UTILIZING SOCIAL NETWORK ANALYSIS FOR NETWORKING GRAPH GENERATION

Social Network Analysis (SNA) is a powerful interdisciplinary field that examines the relationships and interactions among individuals, organizations, or entities within a social system. At its core, SNA seeks to uncover the hidden patterns, structures, and dynamics that underlie these relationships. By representing these relationships as nodes and edges in a network graph, SNA enables the visualization and analysis of complex social structures. It allows us to identify central actors, key influencers, hubs, and subgroups, reveal hidden structures and vulnerabilities, along with the flow of information and resources, shedding light on topics as diverse as the spread of diseases, information diffusion, organizational dynamics, and the formation of online communities.

Tracking Cyber-Adversaries and Constructing their Digital Profile

One of the key strengths of Social Network Analysis lies in its ability to uncover both individual and collective behaviors. Through quantitative measures and visualizations, SNA can reveal important insights, such as the identification of opinion leaders, the detection of isolated or marginalized individuals, or the assessment of a network's resilience to disruptions. Moreover, SNA extends beyond descriptive analysis, offering valuable predictive capabilities by modeling how networks evolve over time and how interventions or changes might impact their structure.

In the field of cybersecurity, Social Network Analysis can pose as an advantageous tool for understanding the tactics, techniques, and procedures employed by cyber adversaries. By mapping the connections between malicious actors, their infrastructure, and their targets, SNA aids in cyber attacker attribution, which is essential for identifying the adversary behind cyberattacks. Analysts can use SNA to track the flow of malicious traffic, uncover hidden relationships between threat actors, and discern patterns of behavior indicative of coordinated attacks. This not only provides organizations with the capability to identify the source of an attack but also provides valuable insights into the broader threat landscape, allowing for proactive defense measures and threat intelligence sharing.

For our research, we will utilize NetworkX [27], a widely adopted Python library designed for the generation, manipulation, and examination of intricate networks and graphs. It plays a pivotal role in SNA for cyber attacker attribution by providing a powerful framework to represent and analyze various aspects of the cyber threat landscape. NetworkX will allow us to model network data as graphs, making it easier to visualize the relationships and connections between various entities, such as IP addresses, attack methods and miscellaneous strings. With its extensive library of algorithms, NetworkX can be used to calculate key network metrics, identify central nodes or actors, and detect anomalies or patterns indicative of malicious activity within a network.

Creating edges

In our framework, we incorporate Social Network Analysis (SNA) to establish connections or edges between profiles or nodes. This involves creating connections or edges between profiles or nodes, contingent upon their respective similarity scores. This approach allows us to represent the relationships and interactions between different profiles in a structured manner. It is a fundamental step that helps us uncover patterns and connections within the cyber threat landscape, providing valuable insights for cyber attacker attribution.

Tracking Cyber-Adversaries and Constructing their Digital Profile

```
# Function to create edges in parallel
def create_edges(params):
    i, j, similarity_matrix, threshold, profiles = params
    edges = []
    similarity_score = similarity_matrix[i, j]
    if similarity_score > threshold:
        user1 = profiles[i]['user']
        user2 = profiles[j]['user']
        edges.append((user1, user2, similarity_score))
    return edges
```

The code snippet is designed to create edges between pairs of profiles (nodes) in the network graph. The similarity score between profiles *i* and *j* is calculated using the “similarity_matrix[i, j]” expression. This score represents how similar or related these two profiles are based on their textual descriptions, referring to the textual content or information within each cyber-attack profile that describes the characteristics or details of the cyber-attacks and are used for similarity calculations and network analysis in the code. In the context of this code, a cyber-attacker profile likely contains various textual data or descriptions that provide information about the attributes, activities, or characteristics of the associated cyber-attack. These textual descriptions can include details such as networking data, attack methods and miscellaneous.

After calculating the corresponding matrix, the code checks whether the similarity score is greater than the specified threshold. If the similarity score exceeds this threshold, it indicates that the profiles are similar enough to warrant creating an edge between them in the network graph. If the similarity score surpasses the threshold, the function retrieves the user IDs (“user1” and “user2”) associated with the two profiles at indices *i* and *j* in the “profiles” list. It then creates an edge representation as a tuple containing the user IDs and the similarity score, “(user1, user2, similarity_score)”. The created edge, if applicable, will be added to the “edges” list. This list accumulates all the edges that meet the similarity threshold criteria.

Subsequently, the newly formed edge is incorporated into the network graph denoted as “G”, taking into account the similarity scores computed between cyber-attack profiles. For each tuple in the edges list, the code uses the “G.add_edge()” method provided by the NetworkX library to add an edge to the network graph “G”.

```
# Add edges to the graph
for user1, user2, similarity_score in edges:
    G.add_edge(user1, user2, weight=similarity_score)
```

Tracking Cyber-Adversaries and Constructing their Digital Profile

By adding these edges to the graph, the code visually represents the connections or relationships between cyber-attacker profiles. Each edge in the graph corresponds to a pair of profiles that exhibit a significant level of similarity based on their textual descriptions. The weight of the edge (similarity score) provides additional information about the strength of the connection. Once the edges are added, the resulting network graph can be subjected to various network analysis techniques. These analyses are valuable for understanding the relationships and potential attributions in the context of cyber-attacks.

Preparing for visualization

To prepare our data for the graph visualization, the spring layout algorithm provided by NetworkX will generate the layout of nodes within the network graph “G”. The “nx.spring_layout()” function is called to compute the layout of nodes within the graph. The spring layout algorithm is a force-directed layout algorithm commonly used in network visualization. It simulates a physical system in which nodes are treated as particles with repulsive forces between them and attractive forces between connected nodes (edges). Over iterations, the algorithm reaches an equilibrium where nodes settle into positions that balance these forces.

```
# Generate the spring layout using NetworkX
pos = nx.spring_layout(G)
```

The result of the “nx.spring_layout()” function is a dictionary called pos, where each node in the graph is mapped to a “(x, y)” coordinate representing its position in a two-dimensional space. The “pos” dictionary contains the layout information for all nodes in the graph. The positions generated by the spring layout algorithm are determined by the algorithm's calculations and the interactions between nodes. Nodes that are strongly connected or have high similarity scores tend to be placed closer together, while less connected nodes are positioned farther apart. The spring layout aims to reveal patterns, clusters, and structures within the network graph.

The following code snippet is responsible for storing the positions of nodes (profiles) in the network graph “G” as attributes associated with each node. The code iterates through the positions of nodes stored in the “pos” dictionary, which is obtained by applying the spring layout algorithm provided by NetworkX to the network graph “G”. This layout algorithm calculates optimal positions for each node in the graph, arranging them in a way that visually represents the relationships between nodes.

```
# Store the positions in the nodes' attributes
for node, (x, y) in pos.items():
    G.nodes[node]["pos"] = (x, y)
```

For each node, the pos dictionary provides the position as a tuple of “(x, y)” coordinates. These coordinates represent the node's location in a two-dimensional space, where “x” and “y” specify the horizontal and vertical positions, respectively. The code then assigns these “(x, y)” coordinate tuples as attributes to the corresponding nodes in the network graph “G”. The “G.nodes[node]” syntax accesses the attributes associated with the node, and “pos” is used as the key to store the position information. This allows the position of each node to be associated with that node as an attribute within the graph's data structure.

Storing node positions as attributes is significant for the subsequent visualization of the network graph using Plotly. When the graph is plotted, these positions are used to determine the physical layout of nodes in the visual representation. By associating positions with nodes as attributes, the code ensures that the positions are readily available for visualization.

Constructing the node and edge traces

This step is constitutive to the development of an interactive and informative network graph visualization. This visualization will allow us to explore the network of cyber-attacker profiles, identify central nodes, detect clusters of similar profiles, and gain insights into the patterns and connections among the profiles. The interactive nature of the visualization enhances the exploration and understanding of the network structure.

The code snippet defines two types of Plotly traces, “node_trace” and “edge_trace”, which are used to create a visualization of the network graph generated earlier in the code. For the node traces, the “x” and “y” lists will store the x and y coordinates of the nodes' positions in the graph. The “pos” dictionary, generated using the spring layout algorithm, provided earlier these positions. The “text” list will store text labels associated with each node. These labels typically contain information about the node, such as the user ID, similarity score, and connected nodes, providing context when hovering over nodes in the visualization.

For the edge traces, similarly the “x” and “y” lists will store the x and y coordinates of the edges' endpoints in the graph. The endpoints are determined by the positions of

Tracking Cyber-Adversaries and Constructing their Digital Profile

the nodes connected by each edge. Distinct configuration settings tailored to the visual presentation of the final graph are defined for each of them.

```
# Create node and edge traces for Plotly visualization
node_trace = go.Scatter(
    x=[],
    y=[],
    text=[],
    mode="markers",
    hoverinfo="text",
    marker=dict(
        showscale=True,
        colorscale="YlGnBu",
        size=15,
        colorbar=dict(
            thickness=15,
            title="Node Connections",
            xanchor="left",
            titleside="right"
        )
    )
)

edge_trace = go.Scatter(
    x=[],
    y=[],
    line=dict(width=0.5, color="#888"),
    hoverinfo="none",
    mode="lines"
)
```

These traces are essential for creating an interactive and informative visualization of the network graph. The node traces allow users to see the positions of nodes, explore their details through hover text, and potentially differentiate nodes based on their attributes using color scaling. On the other hand, edge traces visually represent the connections between nodes using line segments, helping users understand the relationships between profiles.

Plotting the data

The following code snippet of the code focuses on preparing the “node_trace” and “edge_trace” objects for Plotly [28] visualization, ensuring that nodes and edges in the

Tracking Cyber-Adversaries and Constructing their Digital Profile

network graph are appropriately represented. These traces will be used to create an interactive visualization of the network graph, allowing users to explore and analyze the relationships and attributes of cyber-attacker profiles. The text labels and similarity scores provide valuable context when interacting with the visualization.

```
for idx, node in enumerate(G.nodes()):
    x, y = G.nodes[node]["pos"] # Use the node positions generated by NetworkX
    node_trace["x"] += (x,)
    node_trace["y"] += (y,)

    # Add similarity score and node's user ID as a text label
    similarity_score = 0.0 # Default similarity score
    for edge in edges:
        if (edge[0] == node or edge[1] == node) and edge[2] > similarity_score:
            similarity_score = edge[2]
            other_node = edge[0] if edge[1] == node else edge[1]
            similarity_label = f"User: {node}<br>Similarity: {similarity_score:.2f}<br>Connected to: {other_node}"

    # Append the similarity label to the text list
    node_trace["text"] += (similarity_label,)

    # Print progress after processing every 100 nodes
    if (idx + 1) % 100 == 0 or idx + 1 == len(G.nodes()):
        print(f"Node Progress: {idx + 1}/{len(G.nodes())}")

for idx, edge in enumerate(G.edges()):
    x0, y0 = G.nodes[edge[0]]["pos"]
    x1, y1 = G.nodes[edge[1]]["pos"]
    similarity_score = G.edges[edge]["weight"] # Retrieve similarity score from edge attribute
    edge_trace["x"] += (x0, x1, None)
    edge_trace["y"] += (y0, y1, None)
    similarity_texts.append(f"Similarity: {similarity_score:.2f}") # Append similarity score text
```

Concerning the node traces, the code iterates through the nodes in the network graph “G” using a for loop. The nodes represent cyber-attacker profiles, and each iteration processes one node. For each node, it retrieves the “(x, y)” position information from the “pos” attribute, which was generated earlier using the spring layout algorithm. These coordinates determine the node’s position in the graph layout. These coordinates are added to the “x” and “y” lists of the “node_trace” object.

Following, it calculates a similarity score for the current node, initially set to 0.0. It then iterates through the “edges” list to find edges connected to the current node and updates the similarity score if a higher similarity is found. Additionally, it identifies the connected node (“other_node”) with which the current node has the highest similarity.

Tracking Cyber-Adversaries and Constructing their Digital Profile

Similar to the node trace, for the edge traces the code iterates through the edges in the network graph “G”. For each edge, it retrieves the “(x, y)” positions of the two nodes connected by the edge. These positions determine the endpoints of the line segment representing the edge in the visualization. Next, the code retrieves the similarity score associated with the edge from the edge attribute. This score reflects the strength of the connection between the two profiles connected by the edge. Eventually, the “(x0, y0)” and “(x1, y1)” coordinates for the edge endpoints, as well as the similarity score text, are added to the “x” and “y” lists of the “edge_trace” object. This data is used to draw the edges in the graph visualization.

In Appendixes 9 and 10, you will find an illustrative example of a generated plot based on the dataset in scope. It provides a visual representation of how user profiles have been categorized into two distinct clusters.

Within the left cluster, we observe the user labeled as “user_10.2.25.83” (Figure 14), who shares a relatively high similarity score of 0.87 with “user_10.2.25.5”. Meanwhile, on the right cluster, we will find “user_10.2.23.136” (Figure 15), whose profile exhibits a similarity score of 0.52 with “user_10.2.23.65”. This visual depiction allows us to infer that “user_10.2.25.83” and “user_10.2.23.136” have profiles with noticeable dissimilarities. The spatial separation between them on the plot serves as a visual representation of this dissimilarity, providing insights into how profiles relate to one another.

Tracking Cyber-Adversaries and Constructing their Digital Profile

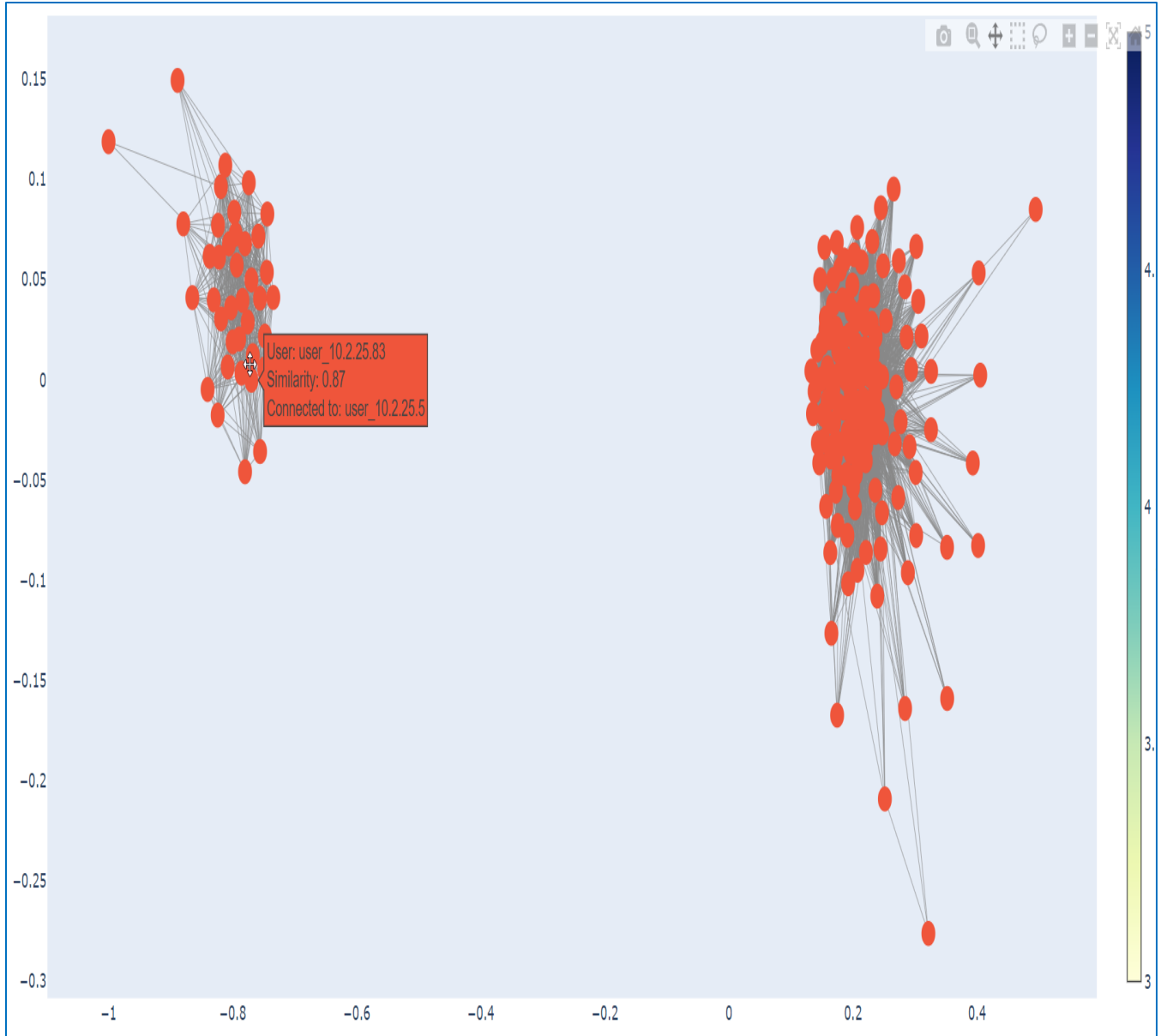


Figure 14: User's "user_10.2.25.83" position in the generated network graph.

Tracking Cyber-Adversaries and Constructing their Digital Profile

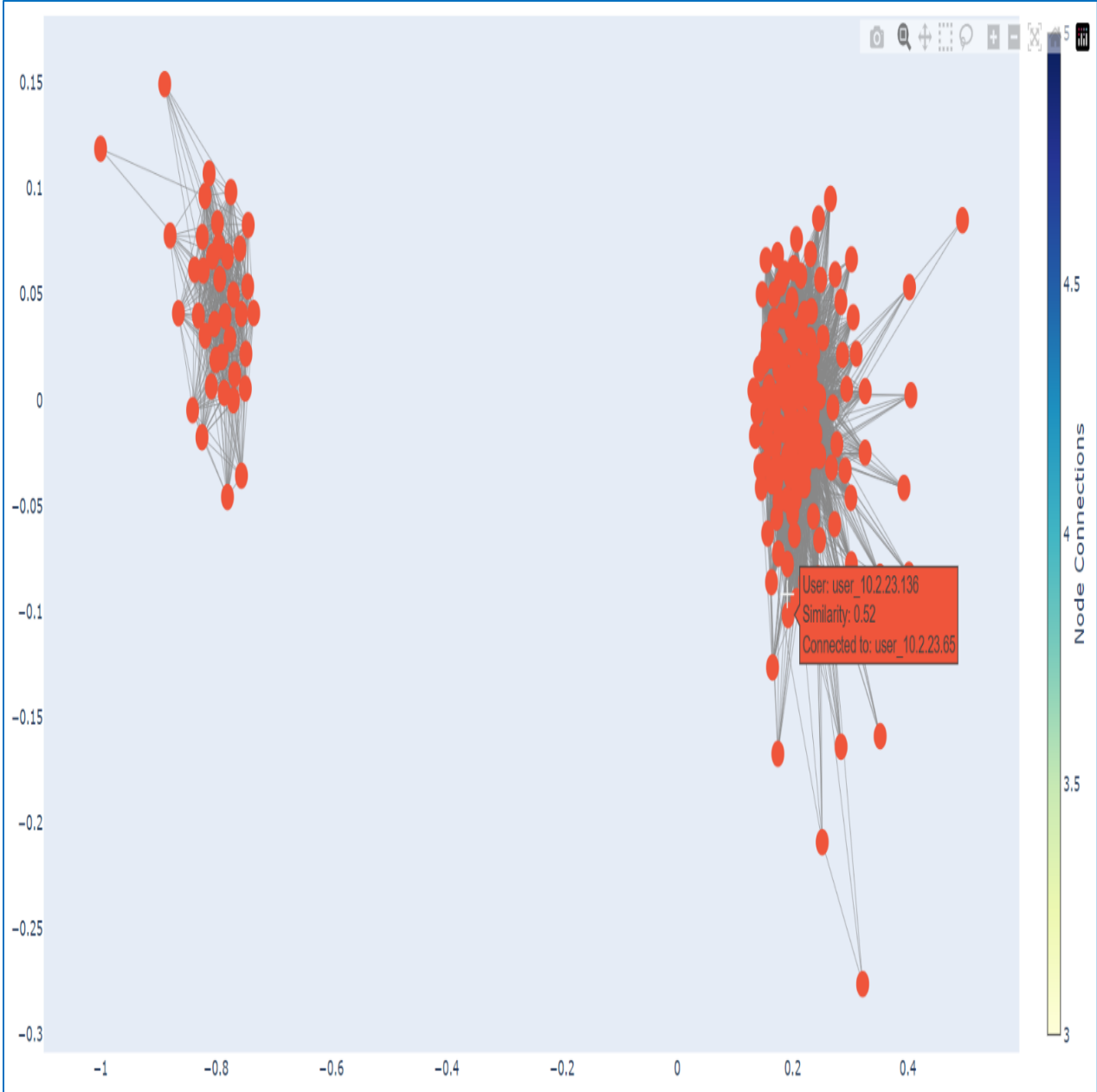


Figure 15: User’s “user_10.2.23.136” position in the generated network graph.

6.4 ENHANCING THE PLOTTING FEATURE FOR WEB APPLICATION INTEGRATION

We will conclude our framework by integrating the networking graph generation into a web application with the help of Python's Flask. This web application extends the capabilities of the previously developed code by transforming it into an interactive and user-friendly platform for exploring our cyber-attacker profiles. The application leverages the capabilities of Flask, a popular web framework, to create a seamless user experience. At its core, the application connects to a MongoDB database, retrieves, and preprocesses cyber-attacker profile data, and then constructs a network graph representing the relationships and similarities between these profiles. This graph is not only visually informative but also interactive, allowing users to navigate through the network, identify key nodes, and uncover patterns of interest.

One of the application's features is its ability to cache the generated network graph, optimizing performance and reducing redundant computations. This means that the graph is computed and visualized only once within a specified time frame, making it readily available to users without the need for repetitive and time-consuming calculations. Moreover, the application empowers users to perform targeted searches within the database, offering a search functionality where users can query for specific cyber-attacker profiles by their unique user IDs. This integration of data retrieval, graph visualization, and search capabilities transforms the code into a dynamic web application ready to assist analysts and security experts in the complex task of cyber-attacker attribution and analysis.

Within this Flask web application, a pivotal role is played by the data preprocessing stage, which transforms raw cyber-attack profile data into a format suitable for analysis and visualization. Drawing from a MongoDB database, the application fetches the relevant data, including user IDs and various attributes related to networking data, attack methods, and miscellaneous information, as it was documented in the previous paragraphs. These individual attributes are ingeniously combined to create comprehensive textual representations for each profile. Subsequently, the application employs the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization technique to convert this textual data into a numerical format. This preprocessing step not only reduces the dimensionality of the data but also enables the calculation of cosine similarity scores between profile vectors, a fundamental metric for establishing connections and relationships between cyber-attack profiles. The result is a robust

Tracking Cyber-Adversaries and Constructing their Digital Profile

foundation that fuels the creation of the network graph, enriching the application's capabilities for visualizing and analyzing complex cyber-attack attribution data.

Subsequently, the incorporation of Flask routes sets the stage for user interaction and data presentation. The application defines two primary routes, each serving a distinct purpose. The root route, accessible via the homepage, showcases the generated network graph, a dynamic representation of the relationships and similarities among cyber-attack profiles. This route is programmed with a caching mechanism that optimizes performance by storing the graph output for a specified period, reducing computational load for repeated requests. In Appendix 5 you may find a demo of the final plot as presented in the root route of the web application.

```
@app.route('/')
@cache.cached(timeout=3600) # Cache for one hour
def index():
    graph = cache.get('cached_graph')

    G = nx.Graph() # Create a new graph
```

Additionally, the application's search route introduces a valuable search functionality. Users can submit queries to search for specific profiles by their unique user IDs. Upon receiving a query, the application retrieves matching documents from the MongoDB database and presents them as search results. Together, these routes provide users with seamless navigation, graph exploration, and targeted data retrieval, enhancing the overall user experience. In Appendix 6 you may find a demo of the search capability as presented in the root route of the web application.

```
@app.route('/search', methods=['POST'])
def search():
    # Handle search queries and retrieve matching documents from MongoDB
    query = request.form.get('query')
    document = collection.find_one({"user": query})
    return render_template('search_results.html', document=document)
```

In summary, this Flask web application enhances the previous code by providing a user-friendly interface for exploring the network graph of cyber-attacker profiles and performing searches on the MongoDB database. Users can view the graph on the homepage, interact with it, and search for specific profiles by user ID. The use of Flask-

Tracking Cyber-Adversaries and Constructing their Digital Profile

Caching optimizes the performance by caching the graph, reducing the computational load for repeated requests.

7. LEVERAGING THE FRAMEWORK: FUTURE APPLICATION AND IMPLEMENTATION

The cybersecurity landscape remains in a perpetual state of flux, driven by the dynamic evolution of cyber adversaries' tactics and techniques. This chapter examines two pivotal of our current framework: the incorporation of the Security-as-a-Service (SecaaS) model and the integration of supervised machine learning.

The adoption of the Security-as-a-Service model represents a significant departure from traditional security practices, offering organizations a pathway to heightened scalability and adaptability. It enables the seamless aggregation of security resources and the collective analysis of security logs from a multitude of sources. This scalability not only opens the door to a surge in data volume but also empowers the framework to scrutinize and decode security threats on a broader and more comprehensive canvas. Beyond this, the collaborative ecosystem that ensues not only sharpens threat detection capabilities but also encourages the sharing of threat attribution across organizations, thereby providing an encompassing view of the ever-evolving landscape of cyber threats and attack patterns. In this theoretical scenario, the advantages of amplified machine learning capabilities, efficient resource allocation, and collective defense initiatives converge to propel your cybersecurity framework into the future.

Simultaneously, the integration of supervised machine learning into the cybersecurity framework promises to usher in a new era of precision and adaptability. This shift represents a crucial step in enhancing the model's capacity to identify and classify specific threats. By leveraging labeled data generated from the insights of the unsupervised model, the supervised machine learning model becomes proficient in recognizing known threat patterns and making accurate predictions. This refinement allows for a more automated and precise approach to threat classification, encompassing specific attack patterns associated with distinct threat actors or groups. The result is a cybersecurity framework that exhibits increased accuracy and speed, effectively safeguarding digital assets in the ever-changing landscape of cybersecurity.

7.1 SECURITY-AS-A-SERVICE MODEL

Scalability and Increased Data Volume

In the transition to a Security-as-a-Service model hosted in the cloud, scalability becomes a pivotal asset. The framework gains the ability to effortlessly accommodate a growing influx of logs from various organizations. This scalability is crucial because it directly translates into an exponential increase in data volume. With a centralized collector aggregating logs from multiple sources, including diverse organizations with distinct threat landscapes, the framework now operates on a much grander scale. This influx of data provides an expansive repository for threat analysis, offering a panoramic view of potential security incidents and attacks.

The increased data volume yields transformative benefits for the framework's operations. It empowers the framework to delve deeper into threat detection and attribution, uncovering patterns that might have remained hidden in smaller datasets. As the volume of logs increases, the framework's analytical capabilities become more robust, allowing it to identify emerging threats, detect sophisticated attack techniques, and correlate events across multiple organizations. Its adaptability to handle this data growth positions it as an effective solution by continuously monitoring for evolving cyber threats and helping organizations respond proactively to safeguard their digital assets.

Cross-Organization Threat Attribution

One of the most compelling advantages of adopting a Security-as-a-Service model within a cloud network of diverse organizations is the ability to perform cross-organization threat attribution. With logs and data pouring in from multiple entities, the framework gains the capacity to draw connections and insights that transcend individual organizational boundaries. It would be able to identify shared attack profiles, similar tactics, and common threat actor behaviors across various organizations. This cross-organization perspective provides invaluable context for security teams, offering a more comprehensive understanding of the evolving threat landscape.

As the framework merges data from different sources, it acts as a digital detective, unveiling patterns of attack that might have otherwise gone unnoticed. By recognizing coordinated or advanced attack campaigns that span multiple organizations, it not only aids in rapid threat mitigation but also facilitates the sharing of threat intelligence among these entities. The framework's cross-organization threat attribution capabilities enable organizations to collectively fortify their defenses against sophisticated adversaries,

Tracking Cyber-Adversaries and Constructing their Digital Profile

strengthening their cybersecurity posture through collaboration and the power of collective knowledge.

Machine Learning and Advanced Analytics

The move to a Security-as-a-Service model hosted in the cloud empowers our cyber attacker attribution framework with the full potential of machine learning and advanced analytics. With access to abundant computational resources, it can now harness the capabilities of sophisticated machine learning algorithms. These algorithms can be trained on the vast and diverse dataset aggregated from multiple organizations, continually improving their ability to detect and attribute cyber threats. Machine learning models can automatically adapt to evolving attack techniques, providing organizations with proactive threat detection and rapid response capabilities.

Additionally, the cloud environment facilitates real-time analysis, allowing our framework to process incoming logs and data with remarkable speed. This enables the identification of emerging threats as they happen, reducing response times and minimizing the impact of security incidents. Advanced analytics tools, such as anomaly detection, behavior analysis, and predictive modeling, become integral to the framework's operations. By leveraging these tools, our SecaaS can provide organizations with insights into potential vulnerabilities, attack patterns, and threat actor behaviors, allowing them to take preemptive action to safeguard their digital assets. In essence, machine learning and advanced analytics elevate your framework from a reactive security tool to a proactive and predictive defender against cyber threats.

Collaboration and Threat Intelligence Sharing

The adoption of a Security-as-a-Service model within a cloud network of diverse organizations fosters a collaborative ecosystem for cybersecurity defense. By centralizing log collection and threat analysis, our framework becomes a hub for organizations to share critical threat intelligence. Organizations would anonymously contribute their security data to the centralized collector, which then would aggregate and analyze it to identify common attack patterns and trends. This collaborative defense approach empowers organizations with a unified front against cyber threats, as they collectively leverage the intelligence gleaned from the broader network to fortify their security measures.

Furthermore, your SecaaS can facilitate secure channels for real-time communication and collaboration among participating organizations. When one entity

Tracking Cyber-Adversaries and Constructing their Digital Profile

detects a novel threat or a sophisticated attack, they can swiftly share this information with others through the platform. This enables rapid incident response, allowing organizations to proactively defend against similar threats. In essence, our framework transforms into a catalyst for information sharing, promoting a culture of collective cybersecurity awareness and preparedness. By pooling their resources and intelligence, organizations can effectively combat the ever-evolving threat landscape, making it significantly more challenging for adversaries to infiltrate their networks and systems.

The adoption of a Security-as-a-Service (SaaS) model within a cloud network, or even in the decentralized approach of a Blockchain [29], consisting of various organizations presents a compelling set of advantages with the potential to transform cybersecurity practices. This innovative approach places scalability at the foreground, facilitating the seamless consolidation of security logs from multiple sources across organizations. This newfound scalability carries the commitment of increased data volume, a highly valuable asset that empowers the framework to scrutinize and interpret security threats on a broader scale. Furthermore, this collaborative ecosystem not only improves threat detection but also encourages cross-organizational attribution of threats, delivering a comprehensive perspective on the evolution of cyber threats and attack patterns. Within this theoretical context, the benefits of enhanced machine learning capabilities, resource efficiency, and collective defense efforts converge to advance your cybersecurity framework into the future.

In this theoretical evolution, the SecaaS model not only enhances collaboration among organizations but also brings advanced analytics and machine learning to the forefront. As the central collector aggregates data from diverse sources, it fosters a culture of collective cybersecurity awareness, with organizations uniting to strengthen their defenses against the ever-evolving threat landscape. Simultaneously, the cloud environment facilitates the integration of advanced analytics and machine learning. These capabilities empower our framework to perform real-time analysis, detect emerging threats, and provide predictive insights into potential vulnerabilities, further elevating its effectiveness in safeguarding digital assets. In this paradigm, collaboration and advanced analytics converge to offer a forward-looking approach to cybersecurity, transcending the limitations of isolated, organization-centric security measures.

7.2 TRANSITION TO SUPERVISED MACHINE LEARNING

The transition from unsupervised to supervised machine learning marks a significant advancement in the capabilities of our cybersecurity framework. Our current framework leverages unsupervised machine learning, excelling in the detection of unknown patterns, anomalies, and potential threats through the analysis of extensive security data without predefined labels. While unsupervised learning is invaluable for its capacity to uncover novel threats, it typically lacks the precision and specificity characteristic of supervised machine learning.

In the shift towards a supervised approach, our framework leverages the insights acquired from its unsupervised counterpart to construct a labeled dataset. This labeled dataset plays a pivotal role in training a supervised machine learning model, empowering it to render more accurate predictions and classifications. The transition process involves several key steps.

Dataset Creation

The dataset creation process assumes a critical role in the transition from unsupervised to supervised machine learning within your cybersecurity framework. This intricate procedure commences with the ongoing operation of our existing unsupervised machine learning model, which diligently examines incoming security logs to identify patterns, anomalies, and potential threats. When the unsupervised model identifies an event as suspicious or noteworthy, it is marked for further examination. In the subsequent phase, these flagged events should be reviewed and assigned with the appropriate labels. These labels serve as the ground truth, indicating whether the flagged event indeed constitutes a security threat or a false positive.

Sustaining the diversity and equilibrium of the dataset is of uppermost importance. It should encompass a representative number of instances for each threat category or behavior of interest to ensure the model's performance remains impartial. Furthermore, the dataset should span a variety of attack vectors, enabling the supervised model to generalize from different scenarios and exhibit resilience against a wide spectrum of threats. The process of dataset creation is an ongoing task rather than a one-time effort, as new threat patterns emerge, and existing threat vectors evolve over time. Ensuring the dataset's continued relevance and currency guarantees that the supervised model continues to learn from the latest threats and security trends, all while adhering to data

Tracking Cyber-Adversaries and Constructing their Digital Profile

privacy and compliance regulations. The dataset should also undergo quality assurance procedures to maintain its accuracy and dependability, involving periodic reviews, validation of labels, vigilance against potential biases, and audits to ensure its integrity.

Feature Extraction

The feature extraction process assumes a central role in the transition from unsupervised to supervised machine learning. It harnesses the insights produced by the unsupervised machine learning model and shapes them into the fundamental building blocks of the labeled dataset for supervised learning. These features encompass a wide range of attributes associated with the flagged events, including timestamps, IP addresses, access patterns, and various technical elements. These features lay the groundwork for the supervised machine learning model, serving as its input variables for threat identification and attribution.

In this shift, the unsupervised model maintains its crucial function as the initial line of defense, identifying potentially suspicious activities and anomalies within the security logs. Concurrently, it serves as a feature generator, capturing the relevant technical attributes of the flagged events. These attributes are then integrated into the dataset, enabling the supervised model to learn from data rich in features. Importantly, the transition ensures that the insights derived from the unsupervised model are not simply discarded but are instead converted into valuable inputs for supervised machine learning. This process enhances the model's capacity to recognize established threat patterns and make accurate predictions, significantly enhancing its effectiveness in detecting and attributing security threats.

Model Training

The model training phase stands as the final significant step in the transition from unsupervised to supervised machine learning within your cybersecurity framework. With the labeled dataset and feature-rich inputs in place, our framework could proceed to train the supervised machine learning model. This model is engineered to identify and classify specific threat types, leveraging the labeled data to generalize patterns linked to known attack vectors. It benefits from the domain expertise applied during the feature extraction stage, ensuring it can establish a comprehensive understanding of threat behaviors and their technical attributes.

This transition empowers our framework to deliver more precise threat identification and attribution. It enables the model to automate aspects of threat

Tracking Cyber-Adversaries and Constructing their Digital Profile

classification, such as recognizing attack patterns associated with specific threat actors or groups. The model becomes adept at making accurate predictions based on the features extracted from the security logs. By learning from the labeled data, the supervised model can efficiently categorize events into well-defined classes.

In this transformative progression from unsupervised to supervised machine learning, our cybersecurity framework is able to undergo substantial development. The transition commences with the thorough establishment of a labeled dataset, an essential asset that enables our framework to render accurate predictions and attributions. The dataset, covering a broad array of threats and attack vectors, serves as a valuable knowledge repository for the subsequent phase.

The feature extraction process captures the essence of the flagged events, preserving the technical characteristics that encapsulate known and emerging threats. The transition ensures that the insights derived from the unsupervised model's analysis are not discarded but are channeled into meaningful features that inform the supervised model. As the unsupervised model acts as the initial guardian, the transition will enhance its effectiveness by leveraging its capabilities in threat identification and behavior analysis.

Finally, the model training phase marks the apex of the transition. The supervised machine learning model is trained to recognize specific threat patterns and behaviors, benefiting from the wealth of knowledge contained in the labeled dataset. With automation capabilities and the ability to categorize events into distinct classes, the model becomes a formidable tool in the fight against cyber threats. This transition equips our framework to provide more precise threat identification and attribution, making it more resilient and responsive to the evolving challenges in the realm of cybersecurity. It emerges as a forward-looking guardian, enhancing the detection and categorization of threats.

8. CONCLUSIONS

In the realm of cybersecurity, the ability to attribute malicious activities to specific threat actors is essential for understanding and mitigating cyber threats effectively. Traditional approaches to threat detection and attribution often rely on the identification of Indicators of Compromise (IoCs), which serve as telltale signs of malicious activity within an environment. However, as cyber adversaries become increasingly sophisticated in their tactics and techniques, the need for more nuanced methodologies capable of discerning between multiple threat actors has become apparent. In response to this challenge, our framework introduces the concept of *Indicators-of-Compromiser (IoCer)*, which extends beyond the traditional IoC paradigm by not only identifying signs of compromise but also providing insights into the unique characteristics and behaviors of individual threat actors.

The differentiation between Indicators of Compromise and Indicators of Compromiser represents a paradigm shift in threat detection and attribution, offering a more comprehensive understanding of the adversaries operating within environments. By leveraging a diverse array of technical artifacts and behavioral patterns, our framework enables the identification of subtle patterns and distinctive signatures unique to each threat actor. In this chapter, we present the results of applying our framework in a red team exercise scenario, where it successfully differentiated between two distinct threat actors based on their *IoCer* profiles. Through a detailed analysis of these findings, we highlight the practical applicability and effectiveness of our framework in real-world cybersecurity scenarios, underscoring its potential to enhance threat detection and attribution capabilities.

Through a meticulous integration of unsupervised machine learning techniques and social network analysis, our framework has proven its efficacy in comprehensively analyzing a wide array of technical artifacts and behavioral patterns. By harnessing the power of unsupervised machine learning, our framework autonomously identifies underlying patterns and structures within the data, allowing for the detection of subtle correlations and anomalies that may evade traditional analysis methods. Additionally, social network analysis provides a holistic view of the relationships between various entities within the threat landscape, allowing for the identification of interconnected nodes and clusters indicative of coordinated malicious activity. This synergistic approach not only enhances the depth and breadth of analysis but also enables accurate threat attribution and differentiation.

Tracking Cyber-Adversaries and Constructing their Digital Profile

The successful differentiation of two distinct threat actors during a red team exercise serves as a compelling validation of the practical applicability and effectiveness of our framework in real-world scenarios. By harnessing a diverse range of technical artifacts, our framework demonstrated its capability to discern subtle nuances and distinctive signatures unique to each threat actor. From analyzing IP addresses and network traffic patterns to scrutinizing cryptographic keys and domain names, our framework provided a granular examination that facilitated precise attribution and offered valuable insights into the tactics, techniques, and infrastructure utilized by each adversary. This level of granularity not only enhances our understanding of threat actor behavior but also strengthens our ability to anticipate and respond to emerging threats in dynamic cybersecurity environment.

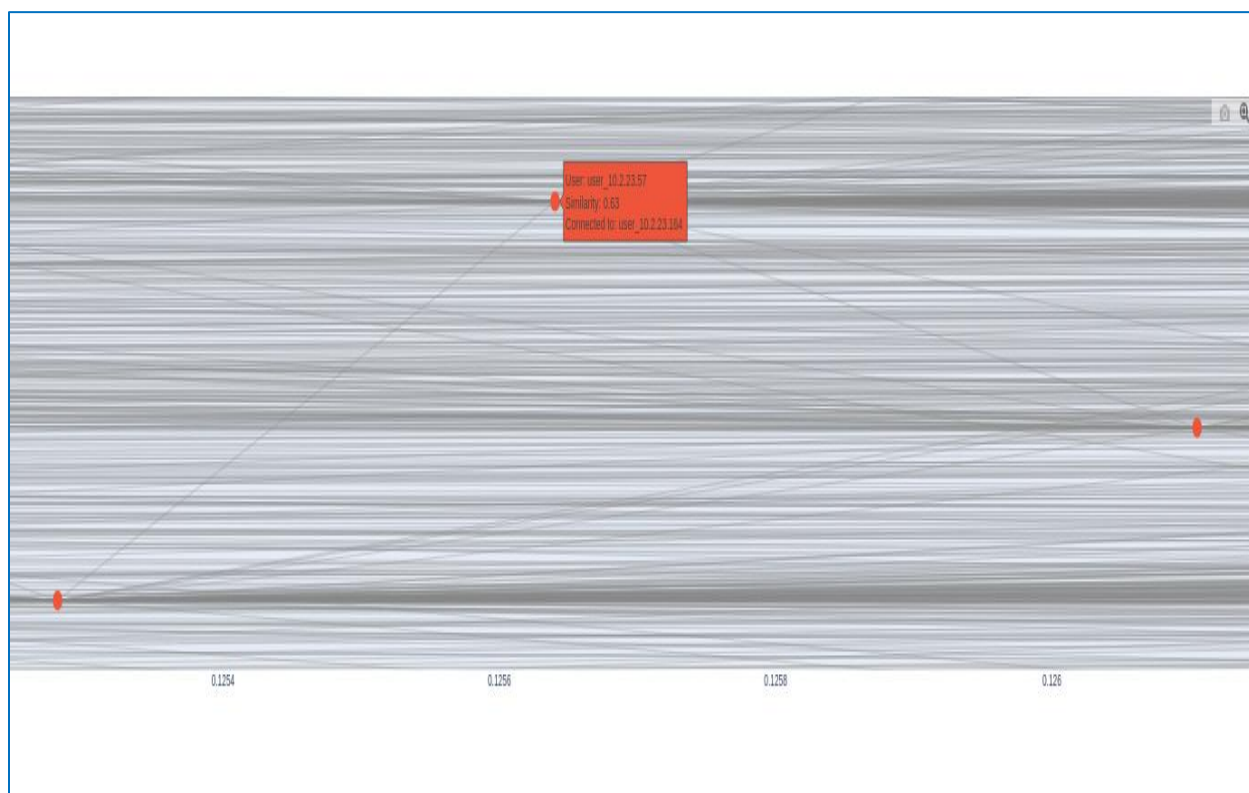


Figure 16: A node representing a user's profile and edges connecting profiles based on their similarities.

Furthermore, the capability of our framework to distinguish between multiple threat actors underscores its versatility and adaptability in navigating the ever-evolving landscape of cyber threats. With adversaries continually refining their tactics and techniques, the significance of accurate threat attribution and differentiation cannot be overstated. Our framework serves as a robust solution to this ongoing challenge, equipping cybersecurity professionals with actionable intelligence to effectively mitigate risks and strengthen defenses against emerging threats. By providing nuanced insights

Tracking Cyber-Adversaries and Constructing their Digital Profile

into the behaviors and infrastructure of threat actors, our framework empowers organizations to stay ahead of evolving threats and proactively safeguard their digital assets.

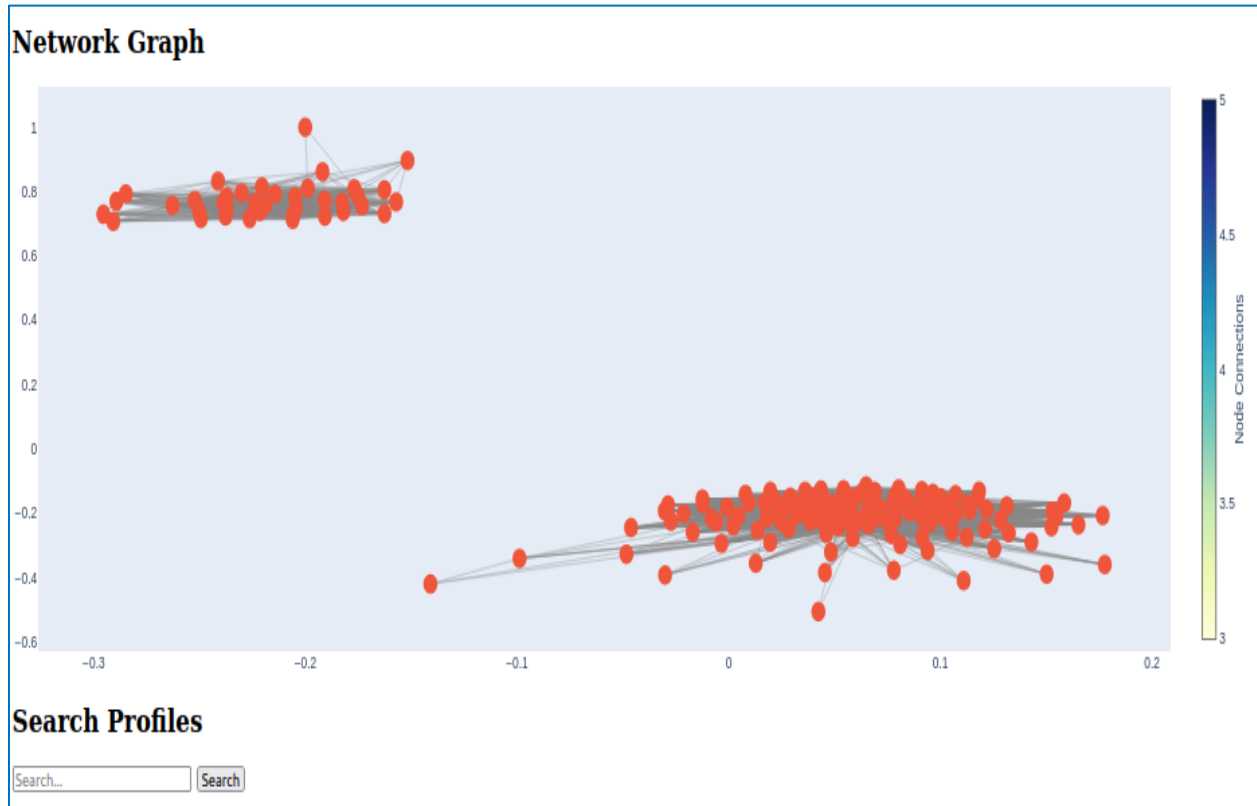


Figure 17: Overview of the framework as a web application.

Looking ahead, it is imperative to dedicate further research and development efforts to enhance the capabilities and scalability of our framework. By continuously refining our methodologies and integrating new data sources and analytical techniques, we can ensure that our framework remains at the forefront of threat attribution and differentiation in the dynamic cybersecurity landscape. This ongoing commitment to innovation will enable us to adapt to emerging threats and effectively address the evolving tactics of cyber adversaries, thereby bolstering our defenses and safeguarding digital ecosystems worldwide.

REFERENCES

- [1] M. Carvalho, "SecaaS-security as a service," *ISSA Journal*, pp. 20-24, October 2011.
- [2] Fyodor, "Nmap Security Scanner," 2018. [Online]. Available: <https://nmap.org>.
- [3] "Gobuster: Directory/File & DNS Busting Tool Written in Go.," 2020. [Online]. Available: <https://github.com/OJ/gobuster>.
- [4] D. Bellucci, "SQLmap - Automatic SQL Injection and Database Takeover Tool.," 25 July 2006. [Online]. Available: <https://github.com/sqlmapproject/sqlmap/>.
- [5] K. Poireault, "Infosecurity Magazine," 27 February 2023. [Online]. Available: <https://www.infosecurity-magazine.com/news-features/cti-attributing-cyberattacks/>.
- [6] T. Micro, "Trend Micro," 19 May 2022. [Online]. Available: https://web.archive.org/web/20230323060837/https://www.trendmicro.com/en_us/ciso/22/e/cyber-attribution-benefits.html.
- [7] J. Han, M. Kamber and J. Pei, "2 - Getting to Know Your Data," in *Data Mining (Third Edition)*, *The Morgan Kaufmann Series in Data Management Systems*, Morgan Kaufmann; 3rd edition, 2012, pp. 39-82.
- [8] U. Gupta, G. Trivedi and D. Singh, "Chapter Eleven - Human AI: Social network analysis," in *Emotional AI and Human-AI Interactions in Social Networking*, Academic Press, 2024, pp. 213-235.
- [9] M. Aiken, *The Cyber Effect: A Pioneering Cyberpsychologist Explains How Human Behavior Changes Online*, Random House Publishing Group, 2016.
- [10] Reuters, "The Guardian," 10 March 2016. [Online]. Available: <https://www.theguardian.com/business/2016/mar/10/spelling-mistake-prevented-bank-heist>.
- [11] M. Bernhard, "SCOPENOW," 1 December 2020. [Online]. Available: <https://www.skopenow.com/news/tracking-down-hackers-with-passive-dns-data-and-ssl-certificates>.
- [12] H. Tanriverdi, M. Zierer, A. K. Wetter, K. Biermann and T. D. Nguyen, "Interaktiv," 8 October 2020. [Online]. Available: <https://interaktiv.br.de/ocean-lotus/en/>.

Tracking Cyber-Adversaries and Constructing their Digital Profile

- [13] E. Romain Dumont, "MITRE ATT&CK," 14 December 2017. [Online]. Available: <https://attack.mitre.org/groups/G0050/>.
- [14] J. DiMaggio, *The Art of Cyberwarfare: An Investigator's Guide to Espionage, Ransomware, and Organized Cybercrime*, No Starch Press, 2022.
- [15] F. Skopik and T. Pahi, "Under false flag: using technical artifacts for cyber attack attribution.," Springer Open, 2020.
- [16] "(n.d.), MITRE Corporation. ATT&CK® Navigator.," [Online]. Available: <https://attack.mitre.org/>.
- [17] "(n.d.), Lockheed Martin Corporation. The Cyber Kill Chain®.," [Online]. Available: <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>.
- [18] E. e. al, "Achievement and achievement motives: Psychological and sociological approaches," in *Expectancies, values and academic behaviors.*, San Francisco, Free Man, 1983, pp. 75-146.
- [19] "Data Capture from National Security Agency (NSA), West Point Military Academy," 2011. [Online]. Available: <https://web.archive.org/web/20190609195310/http://www.westpoint.edu/centers-and-research/cyber-research-center/data-sets>.
- [20] "The Apache Software Foundation. (n.d.). Apache HTTP Server.," [Online]. Available: <https://httpd.apache.org/docs/>.
- [21] "Snort Open Source Community. (n.d.). - The World's Most Widely Deployed Open Source Intrusion Detection and Prevention Technology.," [Online]. Available: <https://www.snort.org/documents>.
- [22] "AlienVault. (n.d.). Unified Security Management & Threat Intelligence.," [Online]. Available: <https://otx.alienvault.com/faq>.
- [23] "AbuseIPDB. (n.d.). AbuseIPDB - Free IP Address Abuse Report and Reputation Lookup.," [Online]. Available: <https://www.abuseipdb.com/faq.html>.
- [24] "VirusTotal. (n.d.). VirusTotal - Free Online Virus, Malware and URL Scanner.," [Online]. Available: <https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works>.
- [25] "Pallets Projects. (n.d.). Flask: A lightweight WSGI web application framework.," [Online]. Available: <https://flask.palletsprojects.com/>.

Tracking Cyber-Adversaries and Constructing their Digital Profile

- [26] "scikit-learn developers. (n.d.). scikit-learn: Machine Learning in Python.," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer.





- [27] D. A. S. P. J. S. Aric A. Hagberg, "NetworkX: High Productivity Software for Complex Networks.," 2020. [Online]. Available: <https://networkx.org/documentation/stable/>.

- [28] "Plotly Technologies Inc. (n.d.). Plotly Python Graphing Library.," [Online]. Available: <https://plotly.com/python/>.

- [29] T.-E. Kavalierou and I. Kantzavelou, "The Smart Contract Guard Model," Poster Presentation in the 3rd Summit on Gender Equality in Computing (GEC 2021), Athens, July 2 2021.


APPENDIX 1

Web logs collection's documentational schema.

Collection web_access		
Idx	Field Name	Data Type
* 	_id	objectId
*	user	string
*	logs	array[object]
*	logs.IP	string
*	logs.timestamp	string
*	logs.http_method	string
*	logs.uri	string
*	logs.response_code	string
*	logs.bytes	string
	errors	array[object]
*	errors.IP	string
*	errors.timestamp	string
*	errors.process	string
*	errors.error_message	string
*	errors.uri	string
Indexes		
	_id_	Primary Key ON _id
Referring Virtual Relation		
	JSON	logs ✓ logs()
	JSON	errors ✓ errors()

APPENDIX 2

Snort IDS collection's documental schema.

Collection snort_logs		
Idx	Field Name	Data Type
* 	_id	objectId
*	user	string
*	logs	array[object]
*	logs.signature_id	string
*	logs.signature_desc	string
*	logs.priority	string
*	logs.source_ip	string
*	logs.source_port	string
*	logs.target_ip	string
*	logs.target_port	string
*	logs.seq_number	string
*	logs.ack_number	string
*	logs.win_size	string
*	logs.tcp_len	string
Indexes		
	_id_	Primary Key ON _id
Referring Virtual Relation		
	JSon	logs < logs()

APPENDIX 3

Collecting and organizing a web application's logs.

```

for document in web_logs.find():
    #get user_id
    user_id = document['user']

    #collect URIs
    uris = set()
    for log in document['logs']:
        uris.add(log['uri'])

    #collect items from uris
    paths = set()
    params = set()
    queries = set()
    uri_items = {"Paths": [], "Params": [], "Queries": []}
    for uri in list(uris):
        item = urlparse(uri)
        paths.add(item.path)
        params.add(item.params)
        queries.add(item.query)
    uri_items["Paths"].append(list(paths))
    uri_items["Params"].append(list(params))
    uri_items["Queries"].append(list(queries))

    #collect HTTP methods
    http_methods = set()
    for log in document['logs']:
        http_methods.add(log['http_method'])

    #collect response codes
    response_codes = set()
    for log in document['logs']:
        response_codes.add(log['response_code'])

    timestamps = []
    #collect timestamps
    for log in document['logs']:
        timestamps.append(log['timestamp'])

    timestamp_format = "%d/%b/%Y:%H:%M:%S %z"

    dt1 = datetime.strptime(timestamps[0], timestamp_format)
    dt2 = datetime.strptime(timestamps[-1], timestamp_format)

    duration = dt2 - dt1

    errors = {"URIs": [], "Messages": []}

    if 'errors' in document and document['errors']:
        for log in document['errors']:
            errors["URIs"].append(log['uri'])
            errors["Messages"].append(log['error_message'])
    #print(errors)

    web_presence_logs = {
        "URIs": list(uris),
        "Paths": list(paths),
        "HTTP_methods": list(http_methods),
        "Response_codes": list(response_codes),
    }

```

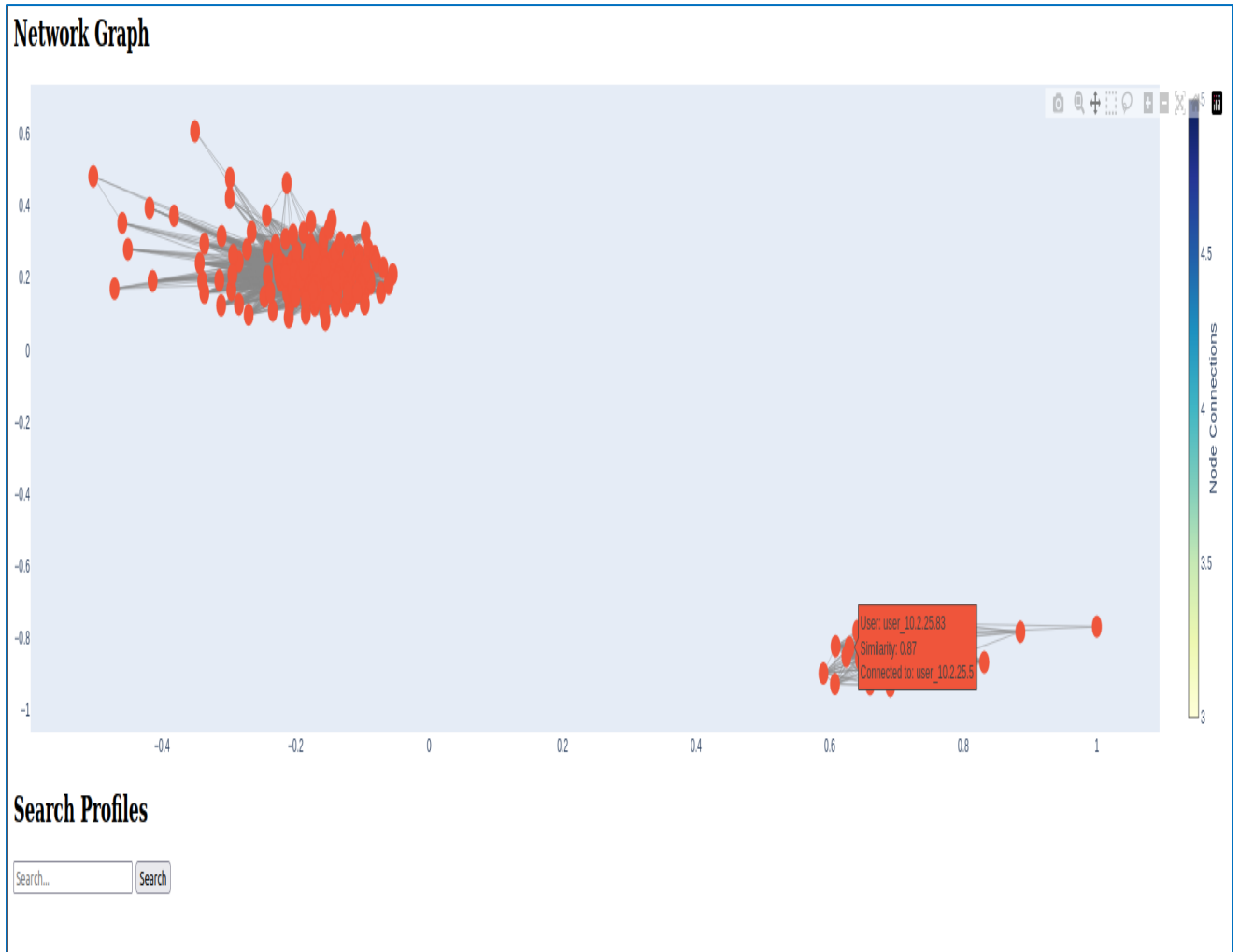
APPENDIX 4

Profiles collection documentational schema.

Collection profiles		
Idx	Field Name	Data Type
* 🔑	_id	objectId
*	user	string
*	networking_data	object
*	networking_data.IP_info	object
*	networking_data.IP_info.IP	array[string]
*	networking_data.IP_info.ASN	array
*	networking_data.IP_info.Reputation	array
*	networking_data.IP_info.Private	array[string]
*	networking_data.Ports	array[string]
*	networking_data.Sum_of_ports	int
*	networking_data.Targets	array[string]
*	attack_methods	object
*	attack_methods.IDS_logs	object
*	attack_methods.IDS_logs.IDS_signatures	array[string]
*	attack_methods.IDS_logs.Priority_levels	array[string]
*	attack_methods.Web_logs	object
*	attack_methods.Web_logs.URIs	array[string]
*	attack_methods.Web_logs.Paths	array[string]
*	attack_methods.Web_logs.HTTP_methods	array[string]
*	attack_methods.Web_logs.Response_codes	array[string]
*	miscellaneous	object
*	miscellaneous.Sum_of_errors	int
*	miscellaneous.Errors	object
*	miscellaneous.Errors.URIs	array
*	miscellaneous.Errors.Messages	array
*	miscellaneous.Sum_of_queries	int
*	miscellaneous.Queries	array[string]
*	miscellaneous.Params	array[string]
*	miscellaneous.Paths	array[string]
Indexes		
🔑	_id_	Primary Key ON _id
Referring Virtual Relation		
✓	JSon	networking_data ✓ networking_data()
✓	JSon	attack_methods ✓ attack_methods()
✓	JSon	miscellaneous ✓ miscellaneous()

APPENDIX 5

The final plot as presented in the web application.



APPENDIX 6

The search profiles capability within the web application.

Search Results

Matching Documents

Field	Value
_id	64de87628689a2d8a8b41e40
User	user_10.2.25.83
Networking Data	<ul style="list-style-type: none"> • IP Info: <ul style="list-style-type: none"> ◦ IP: 10.2.25.83 ◦ ASN: [] ◦ Reputation: [] ◦ Private: ['True'] • Ports: <ul style="list-style-type: none"> ◦ 48086 • Sum of Ports: 1 • Targets: <ul style="list-style-type: none"> ◦ 154.241.88.201:80
Attack Methods	<ul style="list-style-type: none"> • IDS Logs: <ul style="list-style-type: none"> ◦ IDS signatures: WEB-PHP admin.php access ◦ Priority levels: ['2'] • Web Logs: <ul style="list-style-type: none"> ◦ URIs: <ul style="list-style-type: none"> ▪ /admin.php ▪ /admin/ ▪ /%32%47%32%47%78%13%99%76%78%13%99%76 ▪ / ▪ /staff.aspx?%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76%78%13%99%76 ◦ Paths: <ul style="list-style-type: none"> ▪ /admin.php ▪ /admin/ ▪ /staff.aspx ▪ /%32%47%32%47%78%13%99%76%78%13%99%76 ▪ / ▪ /%32%47%32%47%78%13%99%76%78%13%99%76 ◦ HTTP Methods: ['HEAD', 'GET'] ◦ Response Codes: • Miscellaneous: