Discovering the Extent to which Malicious Actors influence Computer Network Traffic with Complex Network Analysis

Gustavo Estribi
College of Engineering and Computer Science
University of Central Florida
Orlando, Florida, USA
gu510710@ucf.edu

Ramsey Shaban
College of Engineering and Computer Science
University of Central Florida
Orlando, Florida, USA
rshaban24@gmail.com

Phaneendra Allu
College of Engineering and Computer Science
University of Central Florida
Orlando, Florida, USA
ph677833@ucf.edu

Avnish Oswal
College of Engineering and Computer Science
University of Central Florida
Orlando, Florida, USA
av871060@ucf.edu

ABSTRACT

Using Python and Complex Network Analysis (CNA) tools to convert computer network traffic captures into graph for visualization and analysis, using a lab-created dataset† of labeled malicious traffic. Using select data from network captures, a comprehensive view of all computers in the network and their communication with each other can be made. Further refinement on labeled malicious traffic may enable training of machine learning models to analyze and detect anomalous traffic in a real-time capture.

† IoT-23 Dataset [7]

1. Introduction

As cyber threats continue to evolve in complexity and scale, more advanced methodologies are being applied to help in the detection and mitigation. Some of the most advanced methodologies involve the use of Machine Learning (ML) where models are tasked with specific issues to detect certain malicious behavior and classify them [1]. Currently, Deep learning (DL), a key component of Machine Learning, has performed even better by improving the capacity for learning of these models[2].

In recent years, graph networks have emerged as a powerful tool in the domain of cybersecurity, offering a versatile framework for modeling and analyzing intricate relationships inherent in cyber threats. In this work we explore the possibility of applying graph network analysis to provide a representation of interconnected data points, making them well-suited for capturing diverse interactions among various entities in cyberspace. By encoding entities such as IP addresses, domains, and user accounts as nodes. Relationships such as communications, access privileges, and malware propagation as edges, graph models enable a holistic view of the threat landscape to facilitate the identification of patterns and anomalies that signify potential threats and uncover hidden connections or predict emerging threats with greater accuracy. This could enable proactive defense measures, such as the development of targeted countermeasures and the prioritization of security investments by gaining valuable insights into the strategies employed by threat actors.

To begin analyzing and predicting cybersecurity issues in a computer network, discovering themalicious traffic is necessary.

2. Related Work

Cyber threat intelligence (CTI) is a discipline within cybersecurity that focuses on collecting, analyzing, and disseminating information related to potential or actual cyber threats. It encompasses a broad spectrum of activities aimed at understanding the tactics, techniques, and procedures (TTPs) used by threat actors to compromise systems, steal data, or disrupt operations. Cyber Threat Intelligence gathers data from various sources, including open-source intelligence (OSINT), closed forums, dark web marketplaces, malware analysis reports, incident response logs, and proprietary threat intelligence feeds. This data is then analyzed to identify patterns, trends, and correlations that indicate potential cyber threats, such as malware campaigns, phishing attacks, or advanced persistent threats (APTs). The insights derived from CTI are used to inform decision-making and enhance cybersecurity defenses, enabling organizations to better anticipate, detect, and respond to cyber-attacks. In one of the related works on which we based our report [5] the author presents a comprehensive exploration of the application of Social Network Analysis (SNA) in the domain of Cyber Threat Intelligence (CTI). The author proposes a systematic methodology for analyzing the Malware Information Sharing Platform (MISP) dataset, augmented by the application of the Diamond Model framework in conjunction with social network analysis (SNA) techniques. This methodology encompasses several interconnected steps aimed at unraveling the intricate web of cyber threats embedded within the dataset. Initially, meticulous preprocessing of the MISP dataset is conducted to extract relevant attributes and normalize the data, laying the foundation for subsequent analysis. The author underscores how degree centrality, a fundamental metric in social network analysis, contributes to identifying key organizations and malicious IP addresses affecting them.

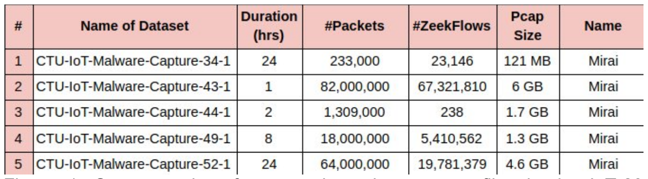
Degree centrality quantifies the number of connections (edges) a node has within the network, serving as a proxy for its importance and influence. In the context of cyber threat intelligence analysis, nodes with a high degree centrality are indicative of entities that are highly connected within the network and potentially influential in propagating information or exerting influence. By analyzing degree centrality metrics within the constructed network, analysts can identify key organizations that are central to the dissemination of cyber threats, as well as malicious IP addresses that exhibit high connectivity and pose significant risks to these organizations; enabling a more targeted and proactive approach to cybersecurity, allowing organizations to prioritize their defenses and allocate resources effectively to mitigate the impact of cyber threats.

Other fields that are being applied for cybersecurity threat intelligence are machine learning (ML) techniques: deep learning (DL) and graph convolutional neural networks (GCNs). The algorithms used by these techniques process vast volumes of data from diverse sources such as network traffic logs, malware samples, and threat intelligence feeds to detect anomalies and indicators of compromise (IOCs). GCNs extend DL capabilities to graph-structured data, enabling the analysis of relationships and interactions between entities in the cyber threat landscape.

In their paper [6] on network traffic analysis through a graph-based approach, the authors propose a novel methodology that leverages Traffic Dispersion Graphs (TDGs) along with temporal dissection and data-level preprocessing to enhance the classification of node behaviors. TDGs serve as a representation of network traffic, with nodes representing entities such as IP addresses or devices, and edges denoting interactions between them. Incorporating temporal dissection helps break down the network traffic data into temporal segments, enabling the analysis of node behaviors over time and capturing evolving patterns and anomalies. Additionally, data-level preprocessing techniques are applied to enhance the quality and relevance of the data for classification purposes.

The authors explored how in recent years Graph Convolutional Networks (GCNs), a class of neural networks designed to operate on graph-structured data, have gained popularity due to their ability to effectively model complex relationships and dependencies in the context of cybersecurity for tasks such as network intrusion detection, malware detection, threat intelligence analysis, and vulnerability assessment. By encoding the TDGs as input for GCNs, the authors use Graph Convolutional Networks (GCNs) to effectively capture the structural and temporal patterns within the network traffic data, allowing for more accurate classification of malicious entities. This approach enables the GCNs to learn from the complex and dynamic nature of network traffic data, leading to improved performance in detecting and classifying malicious behavior within the network

3. Case Study



#	Name of Dataset	Duration (hrs)	#Packets	#ZeekFlows	Pcap Size	Name
1	CTU-IoT-Malware-Capture-34-1	24	233,000	23,146	121 MB	Mirai
2	CTU-IoT-Malware-Capture-43-1	1	82,000,000	67,321,810	6 GB	Mirai
3	CTU-IoT-Malware-Capture-44-1	2	1,309,000	238	1.7 GB	Mirai
4	CTU-IoT-Malware-Capture-49-1	8	18,000,000	5,410,562	1.3 GB	Mirai
5	CTU-IoT-Malware-Capture-52-1	24	64,000,000	19,781,379	4.6 GB	Mirai

Figure 1: Summary data for network packet capture files in the IoT-23 dataset.

Dataset

The IoT-23 dataset, which was made available in January 2020, contains network traffic data from Internet of Things devices. Twenty different malware scenarios are simulated on IoT devices, and three samples of normal, non-malicious device traffic are included. This data was collected in the Stratosphere Laboratory of CTU University in the Czech Republic by the AIC group in 2018 and 2019[7]. IoT-23 provides a diverse collection of tagged data sets that display both benign and malicious IoT traffic, specifically designed to aid researchers in the development of machine learning algorithms. The project is funded by the Prague-based Avast Software.

Every malicious instance found in the IoT-23 dataset involved the installation of a particular piece of malware on a Raspberry Pi, which carried out a variety of tasks and made use of various protocols. The distinct IoT botnet patterns and related protocols that can be seen in the traffic data are what define these cases. On the other hand, the benign data was obtained from the network activity of three real Internet of things (IoT) devices: a Somfy smart door lock, an Amazon Echo, and a Philips HUE smart LED lamp. This ensured that real network behaviors were recorded. The compromised and safe scenarios were carried out in a network space that was controlled and furnished with the same unrestricted internet access that actual Internet of Things devices usually have.

The process starts with a manual analysis of the original packet capture (.pcap) file, where suspicious flows are found. After that, these flows are labeled using an analysis dashboard. The labels are assigned using a formula that follows predefined rules. By extracting the data for every flow from the conn.log file and comparing it to these preset rules, the Flaber Python script further automates the labeling process. Flaber automatically assigns the designated label when the data in a flow meets the labeling criteria.[7]

Each capture's folder contains several files that are necessary for analysis:

README.md: Offers extensive information about the malware and its capture, including the likely name of the malware, its cryptographic hashes (md5, sha1, sha256), the length of the capture, a link to its VirusTotal entry, and a list of all the files in the folder. [7]

pcap: The network traffic capture's original pcap file is contained in this file.

conn.log.labeled. After Zeek was used to analyze the original .pcap file, an improved version of the Zeek conn.log file was produced. It enhances the information found in a typical conn.log file by including two extra label columns in addition to the standard network flow data.[7]
Attack: A label applied to flows that take advantage of security holes such as command injections in HTTP headers or brute-force telnet attacks.

Benign: Awarded when no malevolent or suspicious activity is found on the connection.

C&C: Binary downloads, incoming and outgoing commands, or periodic connections all point to the presence of a connection to a command and control (C&C) server.

DDoS: A term used to describe the traffic flows that are a part of an attack that distributes denial of service to a single IP address.

FileDownload: A label for connections that download files bigger than 3KB or 5KB; frequently associated with dubious IP addresses or ports.

HeartBeat: Typically displaying periodicity and connecting to known malicious ports or IPs, HeartBeat is used for minimal data connections that sustain contact with a C&C server.

Mirai: Designated for traffic patterns that display typical of Mirai Botnet.

Okiru: Tag for links displaying the less frequent Okiru botnet's patterns.

Part Of a Horizontal Port Scan: Identifies connections engaged in port scanning horizontally; these connections can be distinguished by multiple target IP addresses, consistent ports, and comparable byte transfers.

Torii: Used for connections displaying the traits of the botnet known as Torii, which is less widespread than Mirai.[7]

4. Methodology

Complex network analysis (CNA) of computer networks is a relatively new field, with no established method of converting captured network traffic packets into a network graph of nodes and edges. Devising a new method requires defining nodes as individual computers (hosts) and defining edges as the volume of communication between hosts, as a sum of bytes recorded.

Using a dataset with labeled malicious traffic will enable sure definition of nodes that have been influenced by a malicious actor, and traffic that results from this influence.

4.1 Data Extraction

The Pandas Library is utilized to examine the structure of a labeled file. After bypassing the metadata, the data is then loaded into a Pandas DataFrame, tailored to the previously identified structure. Column headers are established for the dataset, and data is loaded under these headings after skipping the initial metadata lines. Only specific columns (src_ip, dst_ip, duration, src_bytes, dst_bytes, label) are preserved for subsequent analysis.

After extracting the DataFrame, the next step is to ascertain the weight of nodes. This involves cleaning the src_bytes and dst_bytes columns by converting any non-numeric values to zeros and converting their datatype to integers.

A dictionary is set up to collect the summed bytes as weights for each pair of source and destination IPs (src_ip, dst_ip). The assignment of weights is contingent on whether src_bytes or dst_bytes are non-zero. The processing of pairs is governed by the conditions related to byte values.

For the IP pair weights: if only src_bytes are non-zero, they are added to the weight for the corresponding src_ip to dst_ip pair. Conversely, if only dst_bytes are non-zero, they are added to the weight for the dst_ip to src_ip pair. If both byte values are non-zero, weights are added for both directions. The accumulated weights and labels for each IP pair are then converted into a new DataFrame.

For our future study we would be estimating weight using duration: a function is defined to convert the values in the duration column to floats, rounding them while gracefully handling any non-numeric entries by returning None. This function is then applied to the duration column of the DataFrame, and any rows resulting in None are removed. The duration is summed for rows corresponding to unique source and destination IP pairs (src_ip, dst_ip). A dictionary is initialized to store these accumulated durations along with their respective labels. Finally, this data is converted into a new DataFrame, focusing on the summed durations and labels.

4.2 Graph Creation

Using Python's CSV library and the data analysis tools provided by Pandas and NetworkX, a rudimentary network graph was made using NetworkX.MultiDiGraph for basic analysis. MultiDiGraph was the most appropriate solution because of its ability to accurately account for parallel directed edges in a network during analysis.

The src_ip and dst_ip have different bytes transferred on each transaction, making the weight on each direction different. Eg. A user can ask google for a webpage which takes 100 bytes of data (src_ip to dst_ip) but google on sending reply gives 500 bytes of data (dst_ip to src_ip). Making a clear distinguishable edge to notify which node can or cannot be a router. (The one which sends most data to all nodes)

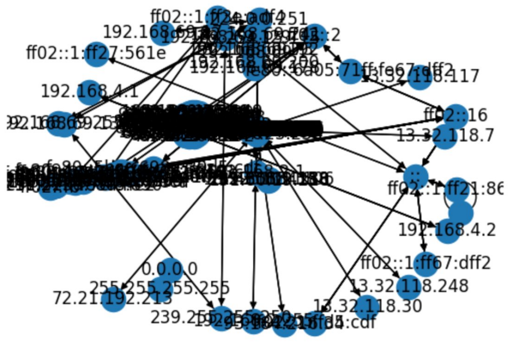


Figure 2: A network graph drawn using networkx.draw()

In an attempt to use NetworkX to visualize this data, the generated graph proved difficult to read and slow to create. Additionally, the static graph proved to be a barrier to comprehension. A search for interactive graphs led to using PyVis (Fig 3), a python library that generates HTML files to view a given network interactively in a web browser.

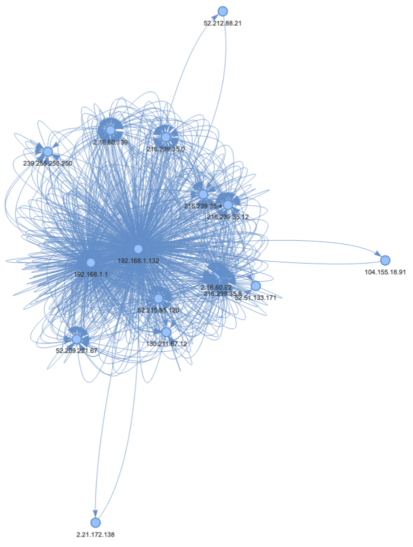


Figure 3: A network graph drawn using the PyVis interactive web view

PyVis graphs proved useful for interactive, dynamic viewing of the network graph, however, on graphs with hundreds of nodes and edges, performance heavily suffered. This problem led to refactoring the data using Pandas, and using a tool called Gephi, separating from Python and NetworkX.

Using Gephi, we were able to import the graphs via 2 MAIN methods: One was to take the graph that NetworkX created and convert it to Gephi format (.gexf) which you can directly import your csv data. The Network import failed and didn't give appropriate results, however the csv data was as required.

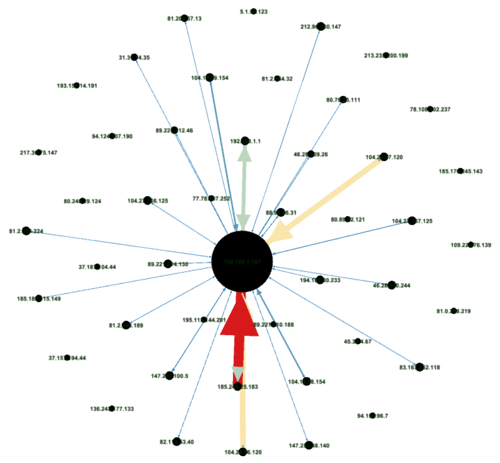


Figure 4: A figure drawn using Gephi

Gephi is a useful tool that combines interactive graph viewport with many modules to analyze data in the graph and calculate metrics per node or on the entire network.

The metrics calculated can also be used to filter the graph and remove segregation.

The most common layout we used and found convienient to use was the Fruchterman Reingold Layout which is majorly used to consider a force (in our case the weights) between any two nodes. This helped us gather a few information on the graph.

5. Findings

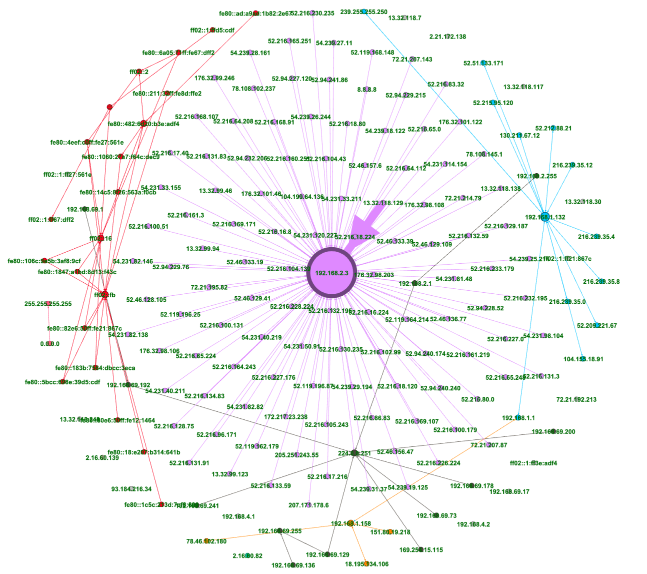


Figure 5: Benign traffic

Through analyzing the graph created on Gephi, some details are readily visible. The node's size is directly proportional to its combined in-degree and out-degree. A larger node, like 192.168.2.3 in the center of the graph, likely serves as a router, network switch, or web server given its high degree. Smaller nodes with degrees less than 10 are likely client computers or IoT devices.

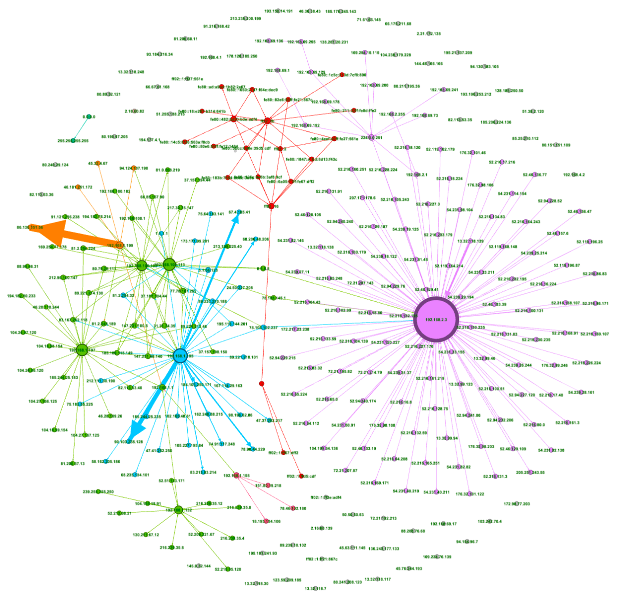


Figure 6: Malicious traffic (298 nodes, 505 edges)

A node's edge size is also proportional to the edge weight. In figure 4, larger arrows replace normal edges with yellow to show inter-quartile sections, blue to show minimum and red to show maximum weight in the graph. Similarly for Figure 5, our clean Network, the Purple edge in between indicated the similar high transfer of bytes to that particular node.

Figure 6, shows the addition of 7 malicious network within our clean network. The netwro here is separated based on modularity community and coder coded accordingly. The red edge crossing the middle of the network demonstrates the IPv6 network.

2 of the clean networks is clearly visible in Purple and Red color, where as the Third Network is mixed with the malicious traffic on which is seen below the Blue colour Network. This can indicate that the clean network is within hops connected to the malicious nodes and is in danger of getting affected.

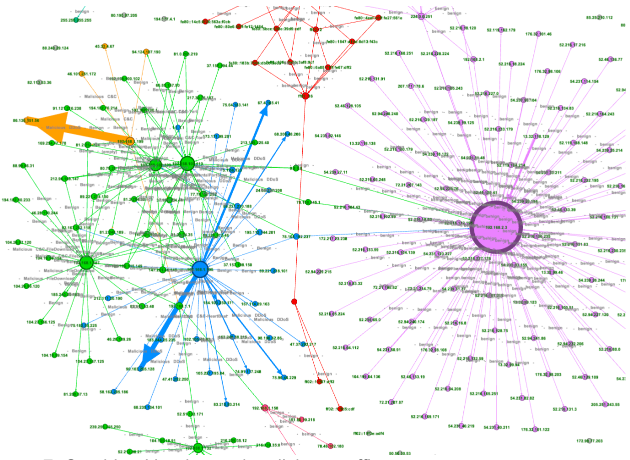


Figure 7: Combined benign and malicious traffic

On observing the left most part of the graph (which consists of the major malicious traffic) we see edges with label DDOS relatively thick edges.

Some nodes stand out as hubs for inter-network communication. For example, in Figure 5, we have multiple clearly visible nodes that serve as routers for sub-networks. These are high-value targets for malware, because routers often bridge two networks, giving access to more potential victim computers. For example, in Figure 8, node 78.108.102.237 connects the pink cluster (benign network) with the green cluster (malicious network).

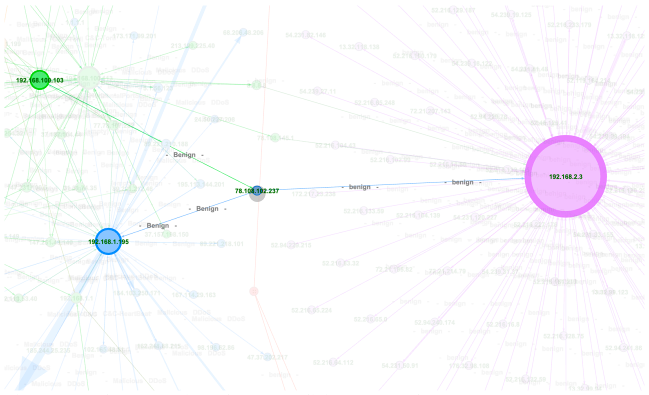


Figure 8: Node Connecting Clean & Malicious Network

Network metrics can also be calculated in Gephi. The following metrics were created using the data shown in figure 6:

Diameter: 7 (network is partially segmented)

Average Path length: 3.51 (network is partially segmented)

Modularity: 0.572

Number of Communities: 65 (6 colored)

Average Degree: 1.695

5.1 Limitations

IPv4 and IPv6 networks appear separate, when in reality, a single host will often use more than one name: one for IPv4 and another for IPv6. Without more in-depth analysis of the data prior to extraction, a single host with more than one IP will appear as separate hosts, with one IP address each.

6. Conclusions

Complex network analysis on computer network traffic reveals interesting metrics not readily available when reading through network packet captures and logs. Using rudimentary data available in any network traffic capture, a graph of network communication is easy to create to easily visualize and understand the important hosts in a network.

More complex analysis will need to be done to reveal statistics like node behavior or traffic flow patterns. Future research can include timestamps from the packet capture logs, to enable a time study of malware spreading across the computer network and calculate metrics on how different types of malware behave over time.

REFERENCES

[1] Yanchen Qiao, Xiaochun Yun, and Yongzheng Zhang. 2016. How to automatically identify the homology of different malware. In Proceedings of Trustcom/BigDataSE/ISPA. IEEE, Los Alamitos, CA, 929–936..

[2] Samaneh Mahdavifar and Ali A. Ghorbani. 2019. Application of deep learning to cybersecurity: A survey. Neurocomputing 347 (2019), 149–176

[3] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[4] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY..

[5] Anastopoulos, V. (2022). Using social network analysis for cyber threat intelligence.ResearchGate.https://www.researchgate.net/publication/362052767_Using_Social_Network_Analysis_for_Cyber_Threat_Intelligence

[6] Zola, F., Segurola-Gil, L., Bruse, J. L., Galar, M., & Orduna-Urrutia, R. (2022). Network traffic analysis through node behaviour classification: a graph-based approach with temporal dissection and data-level preprocessing. Computers & Security, 115, 102632. https://doi.org/10.1016/j.cose.2022.102632

[7] Sebastian Garcia, Agustin Parmisano, & Maria Jose Erquiaga. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set]