2024 4th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME) | 979-8-3503-9118-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICECCME62383.2024.10797094

Proc. of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME 2024)
4-6 November 2024, Male, Maldives

Augmenting DNS-Based Security With NetFlow

Kevin Sam Tharayil
School of ECE
Georgia Institute of Technology
Atlanta, USA
kevinsam@gatech.edu

Panagiotis Kintis
School of ECE
Georgia Institute of Technology
Atlanta, USA
kintis@gatech.edu

Angelos D. Keromytis
School of ECE
Georgia Institute of Technology
Atlanta, USA
angelos@gatech.edu

***Abstract*—As cyber threats become more advanced and prevalent, network operators use a variety of tools and techniques to detect and defend against these attacks. Due to the importance of the domain name system or DNS in almost all internet communications, security controls that are based on DNS are often used by network operators as a first line defense to detect and block malicious network traffic. In this work we present how DNS data can be augmented by combining it with its corresponding network traffic data that is often collected from a different point in the network. This will allow operators to exactly identify hosts within the network that have resolved malicious domains and can identify exactly how much communication happened with the malicious domain. Both of these are not possible using only the passive DNS data that is collected from the network. In this paper we demonstrate this process using DNS data and network traffic data collected from a university network. To this aim, we first identify and measure clock offset between the two datasets. After accounting for the clock offset, we identify traffic to malicious domains by augmenting the DNS data with network traffic data. Furthermore, we do an in-depth analysis to identify the type of malice of the malicious domains which we identified in the data.**

***Index Terms*—DNS, Malicious Traffic, NetFlow**

I. Introduction

Cyber threats are becoming more sophisticated and widespread today. These include malware, phishing, ransomware, advanced persistent threats (APTs) etc., which can cause significant damage to individuals and organization's cyber assets. Hence organizations employ a variety of tools and techniques deployed at various points across their network to detect and defend against such cyber threats. Network operators also monitor and collect various types of network data for further analysis.

Among the various tools used by network operators to protect their networks, one of the most important is the suite of security measures that leverage the Domain Name System (DNS). DNS is a critical component of the internet which is used to convert human-readable domains names to IP addresses. Since most of the activities on the internet starts with a DNS request, network operators use this as a effective first line defense in detecting and blocking communication with malicious entities.

This work is supported by Defense Advanced Research Projects Agency under Contract #HR001120C0155

Applications send DNS requests to DNS recursive servers provided by the network which resolves the request. Network operators use this as a bottle-neck to monitor and block communication to potentially malicious entities and for policy enforcement. In addition to real-time monitoring, organizations often collect this DNS data for further analysis and auditing purposes as well. It is common practice for DNS data to be collected from above the recursive server. While collecting DNS data in this manner provide storage efficiency, we lose information about the exact hosts within the network that sent the DNS request. This is because above the recursive every DNS request is sent by the recursive itself.

In this work we demonstrate how we can augment this DNS data collected from a network with the network traffic data from the same network. This approach is novel compared to traditional DNS-based security measures, as it offers network operators deeper insights by pinpointing the exact host that resolved a malicious domain and determining the precise volume of traffic sent to it. We do this using the DNS and network data collected from a university network. Often in large networks DNS data and network traffic data are collected from different points in the network. So it is common to have clock synchronization issues between the two data collection points. Hence we first measure and correct any potential clock offset between these two datasets before correlating DNS data with its corresponding network traffic.

The contributions of this paper are as follows:

- Using the DNS data and network traffic data collected from a university, we present a technique to identify and measure the clock offset between the two data collection points.
- After correcting the clock offset, we demonstrate how DNS data can be augmented with network traffic data to identify the exact hosts that resolved malicious domains and the amount of traffic that was sent to these domains.
- We carried out an in-depth analysis to identify the nature of malice of the malicious domains that we identified in our data.

II. Background

This section provides an overview of some of the key concepts relevant to this paper.

A. NetFlow

NetFlow is a widely used standard developed by Cisco for summarizing traffic between two IPs [1]. Each flow represents the packets send from a source IP to a destination IP that share some common characteristics. In our case, each flow is a summary of all the packets that have the same source and destination IP address, port number and protocol. The flow also keeps track of the number of packets and total bytes sent. Usually on enterprise networks NetFlow is enabled on network devices like switches and routers to collect network traffic data. While raw network packets can provide more granular information and details, network operators often prefer collecting NetFlow as opposed to raw packets as NetFlow has a significantly smaller storage footprint. NetFlow data is used for a wide variety of applications including, but not limited to, network monitoring, traffic analysis, intrusion detection, network behavioral analysis etc.

B. Domain Name System

The Domain Name System or DNS is one of the most fundamental component of today's internet. DNS is primarily used to translate or resolve user-readable domain names to IP addresses. DNS has become a crucial component of the internet since most of the resources on the internet today are accessed using domain names. Hence most of the communications online start with a DNS resolution.

Typically, to resolve a domain name the DNS client on a user device sends a DNS request to a DNS recursive resolver in the network. The DNS request contains a *qname* field which is the domain name the client needs to resolve, say *foo.com* for example. When a recursive resolver receives a DNS request it first checks its cache to see if it already have the response for the request. If it is not available, the recursive handles the name resolution on behalf of the client by iteratively sending DNS request from itself to a series of DNS servers. The resolver first queries the DNS root name server which will respond with the IP address of the Top Level Domain (TLD) DNS server which is *.com* TLD in our case. The resolver then sends a request to the *.com* TLD server which then responds with the IP address of the nameserver of *foo.com*. Finally, the recursive resolver sends a query to the nameserver of *foo.com* which will return the IP address of *foo.com*. The resolver will then return this IP address to the DNS client that send the DNS query.

DNS responses are in the form of resource records. Every record will have an *rtype* field which specifies the type of value that is contained in the *rdata* field. Type A record indicates that the *rdata* field contains an IPv4 addresses and type AAAA record indicates that the *rdata* field contains an IPv6 addresses. Another important field in the resource record is the *time-to-live* field or *ttl*. This specifies how long each record is valid for.

Once a resolver resolves a domain the response is cached for the duration specified in *ttl*. Any subsequent queries to the same domain name during this time period will be served directly from the cache by the resolver without further queries to any DNS servers. Caching of DNS responses happen on the client side at the OS level and application level as well [2]. Once the user application have the IP address of the domain name it can establish communication with the online resource.

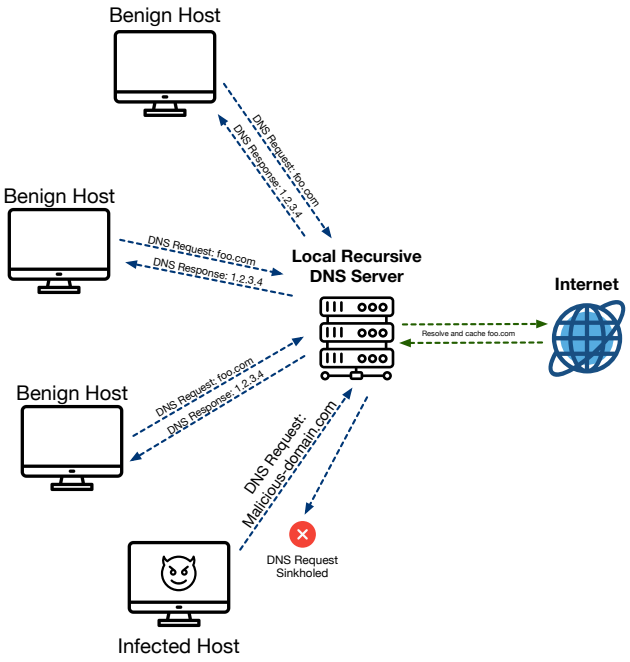

Fig. 1. Local DNS recursive servers in enterprise networks are often configured to sinkhole DNS requests to malicious domains thereby preventing communication.

C. DNS Based Security Controls

Since a DNS resolution is the starting point of most of the communication online, it provides an ideal bottleneck to monitor network communications and to enforce security policies. Hence, devices on large enterprise networks are typically configured to use the local DNS recursive servers within the network which are under the control of the organization's network operators instead of external recursive servers. The operators will configure these servers to identify potentially malicious DNS requests and will block such requests thereby preventing further communications with a malicious entity (Figure 1).

1) Collecting Passive DNS Data: Passive DNS data is the collection of DNS requests and responses observed in a network. Usually it is collected from a point above the recursive or from a point below the recursive. Collecting DNS data from a point below the recursive means logging every single DNS request and response in the network. Here the source address of each request will be the exact device that send the request and the destination will be the IP address of the recursive DNS server in the network to which the device send the request to and vice-versa for the responses. When the DNS data is collected from a point above the recursive, data is logged only when there is a cache miss at the DNS recursive. Here all the request originate from the DNS recursive servers in the network.

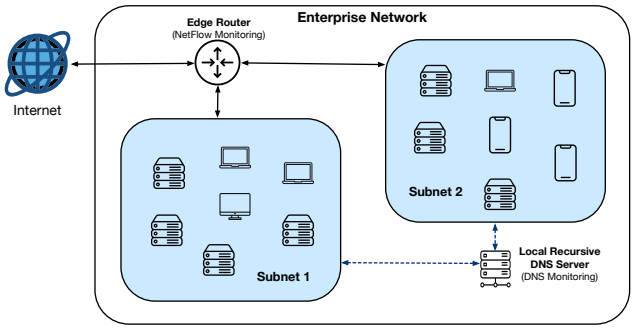

Fig. 2. A representation of an enterprise network. These networks often have its own local DNS recursive servers where DNS is monitored. The DNS data is collected from the recursive server and the NetFlow is collected from various network monitoring points.

While collecting DNS data from below the recursive is the more granular and comprehensive approach, the data volume is too high to be practical. For example, if the DNS data is collected from above the recursive, the response of a domain *example.com* with a *ttl* of 60 minutes will be logged only once in 60 minutes when the first device in the network resolved it. In the following 60 minutes every subsequent request for *example.com* will the served from the cache. Depending on the size of the network there could be dozens or even hundreds of requests for a given domain that will be served from the cache. Collecting the DNS data from below the recursive will require logging all these request making it impractical. While this will provide exact information about all the devices that resolved a given domain, network operators prefer collecting the data from above the recursive due to its manageable size. Additionally, in [3], the authors talks about the preference to collect the DNS data from above the recursive as it protects end user privacy.

D. Augmenting DNS data with NetFlow Data

As mentioned before, DNS data is usually collected from points above the recursive. This comes at the cost of losing information regarding the exact device(s) that resolved a particular domain. This is because from a collection point above the recursive, every request is sent by the recursive itself on the behalf of the devices in the network that originally sent the DNS requests. Hence during an analysis when a network operator needs to identify the exact device that tried to resolve a malicious domain, we need to enrich the DNS data with the network data.

As shown in Figure 2, in large networks it is common for network data and DNS data to be collected from different points. Usually network data is collected from edge routers as flows and DNS data is collected at the recursive as mentioned before. So enriching DNS data with network data required combining two related but disjoint datasets. When combining two datasets like these it is important to ensure that there is no clock offset between the two datasets. In our case synchronization issues between netflow collection and DNS collection can lead to inaccuracies.

Let us recall that every flow (which is the representation of the communication between two devices) is preceded by a DNS resolution and that each resolution will have time interval in which it is valid. Every flow have a timestamp which is the timestamp of the first packet in that flow. Hence, ideally the timestamp of every flow should fall within the valid time interval of its corresponding DNS resolution. In this paper we call these valid flows. But due to clock offsets in data collection sometimes we see flows with timestamps before or after the valid duration of its corresponding DNS resolution. We call such flows preceding flows and trailing flows respectively. In this paper we observe these flows to measure the clock offset between our two datasets so that it can be corrected for accurate results.

III. Related Work

Various work have shown different applications of DNS data and Netflow data. Maghsoudlou et. al. [4] developed and deployed a system that can correlate netflow and DNS data in real time on data from a European ISP. They used this technique primarily for network planning purposes and also to identify traffic to malformed domains. The authors in [5] studied over 1.6 trillion DNS transactions to characterize DNS deployments and traffic patterns. Khalil et. al. took advantage of the dynamic nature and associations among malicious domains to identify more malicious domains from a set of existing known ones using passive DNS data [6]. Similarly passive DNS data was used by Bilge et. al. in [7] to detect malicious domains by extracting various features from DNS traffic which helps in characterizing their properties. Many works have also shown how passive DNS data can be used to detect internet abuse [8] and other suspicious internet traffic [9].

NetFlow data have also been used for various security applications [10] like intrusion detection [11] and for studying traffic patterns [12]. There is a rich body of academic literature on DNS based security. This includes various security controls that can be implemented using DNS [13], [14] and also controls ensuring the integrity of the DNS infrastructure itself [15], [16].

IV. Dataset and Methodology

In this section we talk about the different datasets used in this paper and the methodology we used to link the network traffic data with its corresponding DNS resolutions.

A. Dataset

For this paper we used the network traffic data and DNS data collected from an American university network from 1 May 2023 to 31 May 2023. Passive DNS data is collected from points above the various DNS recursive servers located in the network. In our DNS data, each day on average we had approximately 265M DNS responses.

The network traffic data is collected in the NetFlow format by the IT department using various sensors located throughout the university network. It is well known that real world

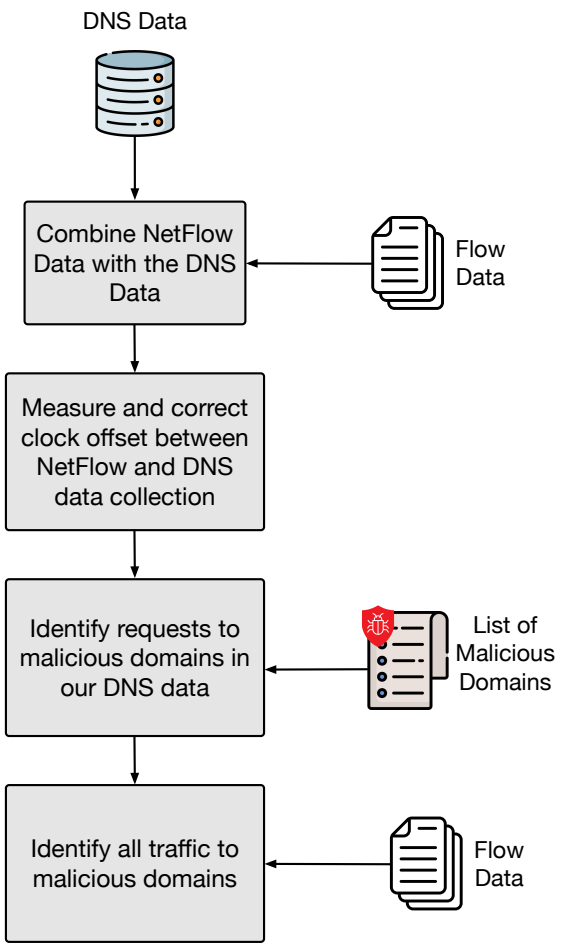

Fig. 3. Visual representation of our methodology. We process Passive DNS and NetFlow data in four steps and we integrate a domain name block-list.

network data is extremely noisy. Additionally, since we are using DNS data collected locally from our network we need to limit the scope of our analysis to network traffic from our own network. To achieve this, we first used various filters to filter out the noise from the network data and also removed all the incoming traffic. After this data sanitization step we are left with approximately 2.22M flows on average each day.

Finally, to identify malicious domains, we used the *Multi ULTIMATE* DNS blocklist from [17]. This list is compiled from over 400 sources and includes mroe than 622,000 domains associated with Ads, Affiliate, Tracking, Metrics, Telemetry, Phishing, Malware, Scam, Free Hoster, Fake, Coins etc.

B. Methodology

Our methodology consists of two distinct processes that help prepare the data before we correlate it. As mentioned earlier, identifying traffic and corresponding DNS requests may be cumbersome, especially as data grows, hence, we are following the process outlined below to increase our accuracy.

In order to combine the DNS and NetFlow data together, we need to first transform the raw data into structures that can be performant in the scale of hundreds of millions of events per day. Therefore, we first convert the raw DNS data into key-value pairs, where we retain only relevant data points that will help us later. These include the domain names, the IP addresses each domain name points to, and the valid time intervals of the resolutions. This will make identifying the associated DNS resolutions for each of the flows faster and more efficient. Then we follow a four step process to first measure clock offset and then identify traffic to malicious domains as shown in Figure 3.

The first step is to combine the NetFlow data with the DNS data to measure potential clock offset. We process data on a daily basis, to correlate the DNS data with the NetFlow data. For each network communication event (network flow) in a given day, we pivot into the DNS key-value store for that same day, to identify the domains that have pointed to the remote side of the communication event (remote IP address, as not the local network IP). In most cases that would suffice to merge the DNS and NetFlow data. However, we have identified several cases in our data where a domain resolution may have been cached or used by a system for more than 24 hours. This case would affect the way we perform the merger, creating more flows that do not have an associated communication event. We solve this problem by increasing the time window we are working with to the previous day of the communication event. After correlating the DNS data with the NetFlow data, we can classify the flows with a corresponding domain name resolution into three categories. Flows that started within the valid time interval of a resolution are termed valid flows. Preceding flows (preceding traffic) are those that occur before the associated domain resolution, while trailing traffic refers to the traffic that occurs after a resolution has expired.

The next step is to accurately measure the time difference between the valid interval of the DNS resolution and the timestamp of the communication event in the case of preceding and trailing flows as these could potentially be caused by clock offset between the two datasets. Aggregating the data allows us to get a better understanding of the underlying distributions. To this aim we compile this time delta data together and plot a cumulative distribution function graph or a CDF graph. This will show us the cumulative probabilities associated with a variable which in our case is the time delta. Figure 4 (left) shows the CDF of the time delta in the preceding and trailing traffic.

The high cumulative probability for the lower values of time delta for the preceding traffic shows a clear trend. Focusing more on this region in Figure 4 (right) clearly shows a clock offset of 3 second or less between the NetFlow data and DNS data for 70% of the preceding traffic. Thus we are able to exactly measure the clock offset.

Once we have successfully measured and corrected the clock offset between the two datasets, we are able to start working towards identifying malicious flows. We assemble the lists mentioned earlier into one block-list, which we can now correlate with the DNS data. At this point, we have identified malicious domains which we can then attribute to communication events.

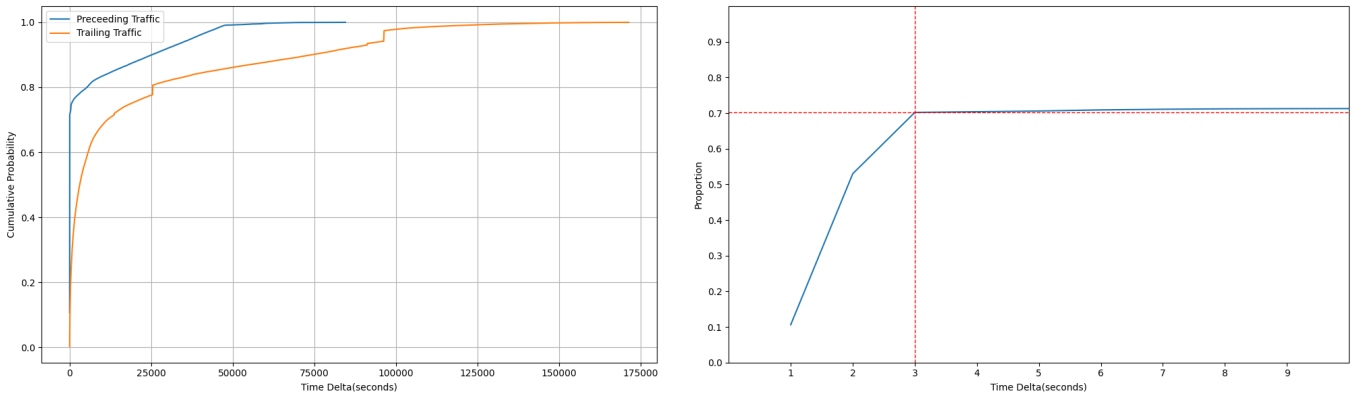

Fig. 4. Clock offset distribution between DNS and NetFlow data.

Therefore, at the final step, we take advantage of the block-lists that include IP addresses and the IP addresses we identified as related to malicious domains in the previous step, and annotate the network communication events. Hence, we now have flows that we know are related to malicious activity. In the following section we present our findings and will take a closer look to identify the type of malware the malicious domains we identified in our data are associated with.

V. Results

Using the methodology described in the previous section, out of the 2.22M flows each day we were able to correlate 88.62% of flows with a corresponding DNS resolution. Using the DNS blocklist we were able to identify traffic to 8176 unique malicious domain from the entire dataset. A total of over 2.7M flows in the dataset were traffic to these malicious domains and this traffic was generated by 609 hosts in the network.

A. Identifying the Nature of Malice

To get a better understanding of the exact nature of these malicious domains we identified in out data we used the online service VirusTotal. The VirusTotal API lets users query files, domain names, IP addresses, etc. and get analysis report which include threat reputation and analysis details produced by over 70 antivirus products and other security tools and datasets. We used the VirusTotal *domain* API endpoint to get the *communicating_files* relationship of these 8,176 malicious domains that we found in our results. This will return a list with the details of all the files that have been reported to communicate with these domains.

Among other details for each file, the API returns the number of votes each file got from the security community reporting it as malicious. Out of the 8,176 malicious domains we queried, 527 domains had at least one malicious communicating file. In total we identified 3,917 unique malicious files that communicated with these 527 domains. Similar to the way we queried the *domain* endpoint to identify the files that are associated with the malicious domains we queried the *files* endpoint to get the report for each of the 3,917 malicious communicating files. This endpoint provides other data points around the file queried, including the *malware_classification* field. The *malware_classification* field includes tags in the value that help us identify the type of malware each file is associated with. VirusTotal assigns these tags based on the behaviour observed during sandbox execution of the files. Note that identifying the exact malice nature of every file is difficult. In that case VirusTotal gives them generic classifications like "malware" or "unknown". The different types of malware classification tags we observed in the data are shown in Table I.

TABLE I.
Tags given to files by sandboxes indicating the type of malware

Malware Type	Count
TROJAN	755
EVADER	701
STEALER	234
RANSOM	179
RAT	114
SPREADER	111
PHISHING	72
GREYWARE	59
EXPLOIT	57
ADWARE	42
BANKER	24

As shown in the table the top three malware type tags that we identified are trojans, evaders and stealer malware. Trojans are malware that are disguised as legitimate software. Users are tricked into opening them thereby loading and executing the malware on their device [18]. Some malware perform evasive tactics when it detects that it is being run in a sandbox. When sandboxes detect such evasive behaviour these files are labeled as evaders. Stealers are malware that are designed to target and exfiltrate data. Stealers commonly target data like browser data, banking data, system information etc.

VI. Discussion

If our technique is implemented in real-time traffic, on a network that already use DNS based security controls, DNS

requests to malicious domains are sink-holed thereby preventing any further communication. In this scenario the benefit this technique provide is to identify the potentially infected host that tried to resolve a malicious domain. On the other hand network operators often carryout forensic investigation on networks to identify hosts that communicated to malicious domains that were only identified as malicious at a later stage. An example for this are stealthy attacks from advanced persistent threats or APTs where the indicators of compromise like domain names are only identified weeks after the attack. In this case, the idea of combining network traffic data with DNS data is especially powerful as it can exactly identify the host that tried to resolve the malicious domain and also get details of exactly how much data was transferred. In these scenario since the domain is reported as malicious only at a later stage, the DNS based detector employed in the network will not be able to sinkhole the DNS request and host will be able to communicate with malicious domains.

In this work we were able to measure clock offset based on preceding flows. While clock offset can also cause trailing flows, we also note that trailing flows can also be cause by recursive servers or by applications not honouring the *ttl* associated with each resource record.

VII. Conclusion

In this paper, we demonstrated how augmenting DNS data with NetFlow data can significantly enhance network operators' ability to identify precise details of traffic to malicious domains. Our study involved collecting and analyzing datasets from a university network over one month. The process began with identifying and measuring the clock offset between the two datasets, a critical step given the common clock synchronization issues in large networks. This alignment ensured accurate correlation of DNS queries with the corresponding network traffic data.

Through this augmented analysis, we successfully pinpointed the exact hosts within the network that resolved malicious domains and quantified the volume of communication with these domains. This level of detail is unattainable using passive DNS data alone, which lacks host-specific information due to the aggregation at the recursive server level.

Furthermore, after identifying traffic directed to malicious domains, we utilized VirusTotal for an in-depth analysis to ascertain the nature of the threats posed by these domains. This comprehensive examination revealed that trojans, evaders, and stealer malware were the predominant types of malware associated with the malicious domains detected in our dataset. Such detailed threat classification is invaluable for network operators, as it informs more targeted and effective defensive measures.

The methodology and findings presented in this paper underscore the enhanced security insights that can be achieved through the integration of DNS and network traffic data. By addressing clock synchronization issues and leveraging both data types, network operators can significantly improve their detection and response capabilities against sophisticated cyber threats. Our approach provides a robust framework for real-time and retrospective security analysis, thereby bolstering an organization's overall cybersecurity posture.

References

[1] B. Claise, "Cisco systems netflow services export version 9," Tech. Rep., 2004.

[2] B. Imana, A. Korolova, and J. Heidemann, "Enumerating privacy leaks in dns data collected above the recursive."

[3] J. M. Spring and C. L. Huth, "The impact of passive dns collection on end-user privacy," *Securing and Trusting Internet Names*, 2012.

[4] A. Maghsoudlou, O. Gasser, I. Poese, and A. Feldmann, "Flowdns: correlating netflow and dns streams at scale," in *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies*, 2022, pp. 187–195.

[5] P. Foremski, O. Gasser, and G. C. Moura, "Dns observatory: The big picture of the dns," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 87–100.

[6] I. Khalil, T. Yu, and B. Guan, "Discovering malicious domains through passive dns data graph analysis," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 663–674.

[7] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis." in *Ndss*, 2011, pp. 1–17.

[8] S. Torabi, A. Boukhtouta, C. Assi, and M. Debbabi, "Detecting internet abuse by analyzing passive dns traffic: A survey of implemented systems," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3389–3415, 2018.

[9] K. R. Barbosa, E. Souto, E. Feitosa, and K. El-Khatib, "Identifying and classifying suspicious network behavior using passive dns analysis," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*. IEEE, 2015, pp. 160–167.

[10] S. Choudhary and B. Srinivasan, "Usage of netflow in security and monitoring of computer networks," *International Journal of Computer Applications*, vol. 68, no. 24, 2013.

[11] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Netflow datasets for machine learning-based network intrusion detection systems," in *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10*. Springer, 2021, pp. 117–135.

[12] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju, "Visflowconnect: netflow visualizations of link relationships for security situational awareness," in *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, 2004, pp. 26–34.

[13] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through dns data analysis," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[14] M. Trevisan, I. Drago, M. Mellia, and M. M. Munafo, "Automatic detection of dns manipulations," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 4010–4015.

[15] R. Perdisci, M. Antonakakis, X. Luo, and W. Lee, "Wsec dns: Protecting recursive dns resolvers from poisoning attacks," in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2009, pp. 3–12.

[16] S. Adiwal, B. Rajendran, S. D. Sudarsan *et al.*, "Dns intrusion detection (did)—a snort-based solution to detect dns amplification and dns tunneling attacks," *Franklin Open*, vol. 2, p. 100010, 2023.

[17] hagezi, "dns-blocklists," https://github.com/hagezi/dns-blocklists, 2024, accessed: 2024-06-05.

[18] M. R. Faghani and U. T. Nugyen, "Modeling the propagation of trojan malware in online social networks," *arXiv preprint arXiv:1708.00969*, 2017.