

Global Analysis with Aggregation-based Beaconing Detection across Large Campus Networks

Yizhe Zhang
University of Virginia
Charlottesville, Virginia, USA
yz6me@virginia.edu

Molly Buchanan
University of Virginia
Charlottesville, Virginia, USA
mkb4vb@virginia.edu

Hongying Dong
University of Virginia
Charlottesville, Virginia, USA
hd7gr@virginia.edu

Donald E. Brown
University of Virginia
Charlottesville, Virginia, USA
deb@virginia.edu

Alastair Nottingham*
University of Virginia
Charlottesville, Virginia, USA
atn5vs@virginia.edu

Yixin Sun
University of Virginia
Charlottesville, Virginia, USA
ys3kz@virginia.edu

ABSTRACT

We present a new approach to effectively detect and prioritize malicious beaconing activities in large campus networks by profiling the server activities through *aggregated signals* across multiple traffic protocols and networks. Key components of our system include a novel time-series analysis algorithm that uncovers hidden periodicity in aggregated signals, and a ranking-based detection pipeline that utilizes self-training and active-learning techniques. We evaluate our detection system on 10 months of real-world traffic collected at two large campus networks, comprising over 75 billion connections. On a daily average, we detect 43% more periodic domains by aggregating signals across multiple networks compared to single-network analysis. Furthermore, our ranking pipeline successfully identifies 1,387 unique malicious domains, out of which 781 (56%) were unknown to the major online threat intelligence platform, VirusTotal, at the time of our detection.

CCS CONCEPTS

• Security and privacy → Network security; Intrusion detection systems.

KEYWORDS

intrusion detection, self-training, active learning

ACM Reference Format:

Yizhe Zhang, Hongying Dong, Alastair Nottingham, Molly Buchanan, Donald E. Brown, and Yixin Sun. 2023. Global Analysis with Aggregation-based Beaconing Detection across Large Campus Networks. In *Annual Computer Security Applications Conference (ACSAC '23)*, December 04–08, 2023, Austin, TX, USA. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3627106.3627126>

*Alastair Nottingham is now with Peraton Labs.



This work is licensed under a Creative Commons Attribution International 4.0 License.

ACSAC '23, December 04–08, 2023, Austin, TX, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0886-2/23/12.
<https://doi.org/10.1145/3627106.3627126>

1 INTRODUCTION

Modern sophisticated cyber attacks often rely on command and control (C&C, or C2) communications to remotely control infected endpoints (e.g., APT29 [42], the Sony hack [92]). Adversaries typically establish stealthy *outbound* communication channels from compromised internal hosts (bots) to external C2 infrastructures (bot masters) using common application protocols like HTTP and TLS [24]. In a centralized botnet architecture, the bot master relies on each bot to announce its presence and *regularly* connect to the external C2 server for commands and upgrades [91]. These timed connections - also known as 'beacons' - are pre-programmed in the malware [62] and exhibit periodic communication pattern in network traces [6, 9, 43]. Consequently, defenders can identify the compromised host by analyzing these periodic time-series patterns.

While not all malware exhibits periodic communication patterns, detecting beaconing activity is yet one of the most effective ways for threat hunting [18] given its widespread usage. Research has found that over 90% of malware families manifest periodic behavior [36]. Notably, beaconing malware such as Andromeda [7], Conficker [54], Zeus [8], and QBot [16] continue to be among the most active botnets in recent years [5, 23, 40, 65]. In addition, beaconing detection is effective even when the packet payload is encrypted (e.g., TLS), making it an ideal solution for campus networks where deep packet inspections and TLS interceptions are not feasible due to high traffic volume.

Nevertheless, accurately detecting *malicious* beaconing traffic poses challenges for two reasons: (i) detection may fail due to missing or irregular signals, and (ii) benign programs can also exhibit periodic behaviors like software updates. In large campus networks, these challenges are further exacerbated by the difficulty of capturing continuous traffic for most devices, as individuals can join/leave the network at anytime, resulting in fragmented signals.

In our work, we propose a new approach to address these challenges at each step of the detection process, outlined as below:

Step 1: Global analysis to reconstruct beaconing signals. Inter-organizational cooperation in cyber defense typically *shares* attack information *after* detection. Indicators of Compromise (IoCs) are distributed through threat intelligence platforms such as STINGAR [75], enabling feed subscribers to update their firewall rules accordingly. However, these solutions primarily take place *after* attack detection within individual networks, with no integration of data across multiple organizations. In contrast, our approach,

referred as *global analysis*, integrates data *at the signal level* through collaboration across institutions, reconstructing a more comprehensive view of the beaconing activity.

Step 2: Novel periodicity detection algorithm to handle noisy data. While aggregating data in *global analysis* helps in reconstructing missing signals, it introduces additional noises, particularly with the high volume of traffic from multiple organizations. To tackle this challenge, we develop a new periodicity detection algorithm (Section 4) that employs Empirical Mode Decomposition [34], a data-adaptive multi-resolution technique. Our algorithm effectively captures periodicity and regularity of signals in much noisier scenarios, allowing us to leverage data across multiple campus networks (*global analysis*) to enhance detection performance.

Step 3: Ranking pipeline to prioritize most suspicious activities with limited labels. While the previous two steps aim at identifying periodic activities in the presence of missing signals, they lead to significantly more identified beaconing domains, which may or may not be malicious. Dealing with this increased volume poses challenges for malicious beaconing activity detection due to the limited availability of ground-truth labeling resources [30] and the sheer volume of traffic.

Towards this end, we design a ranking pipeline that employs *self-training* [83], a state-of-the-art semi-supervised machine learning technique, to reduce model bias through data re-balancing in the early training phase. In addition, we incorporate *active learning* into the daily detection pipeline that ranks periodic domains from the most suspicious to the least suspicious and learns from external oracles to validate the top-ranked domains. This approach empowers the model to gradually improve the performance and keep updated throughout the time (Section 5).

Deployment and evaluation. Our beaconing detection system is deployed at (i) two individual campus networks, and (ii) a *global analysis cluster* where data aggregation is performed. We systematically evaluate the detection results in these two setups using real-world traffic collected from two campus networks (10 months, 75B events). Our key evaluation results are as follows:

- Performing *global analysis* across the two campus networks allows the system to detect 65.32% more malicious domains than running the system on each individual network separately.
- On a daily average, about 77 suspicious beaconing cases were reported in the *global analysis* setup, out of which 72 (93.5%) are confirmed truly malicious by VirusTotal, a widely used threat intelligence platform [93].
- Our system detects 509 malicious domains that were unknown to VirusTotal at the time of detection. Additionally, we quantify the extent of VirusTotal’s query delay, offering valuable insights to the research community.

In summary, our key contributions are as follows: (i) we develop a system that performs novel *global analysis* leveraging data across multiple organization, (ii) we employ new *data-adaptive multi-resolution techniques* to detect periodicity patterns in the presence of large noises, (iii) we introduce *self-training* into the semi-supervised active-learning pipeline to perform detection on large volume of traffic with limited labels and human involvement, and (iv) we run our system on the *largest* dataset of network traffic known to date, demonstrating the efficacy of our detection system in real-world scenarios.

Artifact. While we cannot release the campus traffic data used in our main evaluation due to legal and privacy rules, we have made our source code and simulation for the periodicity detection algorithms publicly available: <https://github.com/yzzhn/bcndetection>.

2 MOTIVATION AND CHALLENGES

Prior works [4, 14, 19, 29, 32, 33, 37, 78, 88] typically reconstruct the time series of network connections based on each {source, destination} pair. We categorize these algorithms as *fine-grained* detectors as their underlying shared concept is to build as precise a time series as possible by representing each individual {source, destination} pair in a finer granularity. For example, the source and destination can be represent by {IP, port, MAC (Media Access Control) address, user agent, device identifier} and {IP, port, full domain name, top-level domain name, AS number, URL} respectively [33]. We identify two main issues when applying *fine-grained* algorithms to real-world campus network data:

- *Difficulties in campus network host tracking.* Counter-intuitively, the campus Security Operations Center (SOC) often does not have full visibility into, or control over, internal device tracking. In practice, much of the traffic log data collected at our campus lacks corresponding MAC addresses due to complex Network Address Translation (NAT) deployments, limited visibility into uninstrumented subnets, and record loss when logging infrastructure is under heavy load.
- *Vulnerabilities to common attacker evasion techniques.* Such detection algorithms can be easily evaded by common attacker evasion techniques, such as DNS (Domain Name System) fast fluxing [56], or by placing the C2 server in a cloud environment [17] where a domain can be associated with multiple IP addresses.

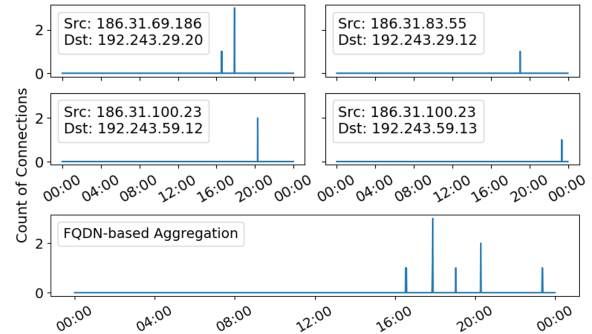


Figure 1: Periodic pattern is not evident in *fine-grained* analysis. IPs are anonymized.

Due to above issues, time series reconstructed by *fine-grained* detectors are often *broken* in reality, resulting in periodicity detection failure. We illustrate an example of incomplete periodic signals observed in real campus network traffic in Figure 1. The upper four figures show the time series of four {source, destination} pairs, where all destination IPs host the same malicious domain. Interestingly, though using any individual time series alone is insufficient for periodicity detection, we observe that aggregating all signals together based on the shared Fully Qualified Domain Name (FQDN) generates a periodic pattern, as shown in the bottom of

Figure 1, thus making it possible to detect such beaconsing behavior. We further observe that by aggregating signals using shared domain names across protocols and organizations, we can uncover periodic patterns that were not evident in any single protocol or network.

These observations motivate our design of a new *aggregation-based* beaconsing detection system with the ability to reconstruct complete time series across multiple traffic protocols and networks. However, such approach is non-trivial and poses several challenges:

- *Increased noise in aggregated signals compared to fine-grained time series*: we develop a new periodicity detection algorithm to unfold hidden periodicity against various noise in Section 4.
- *Distinguish between benign and malicious periodic network activities*: we build a ranking system with semi-supervised learning that prioritizes the most suspicious activities for further human analysis and SOC investigation in Section 5.

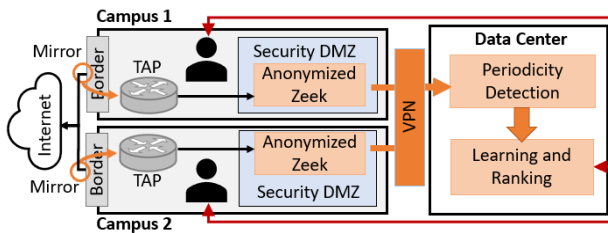


Figure 2: Beaconsing detection system across two large campus networks.

3 DATASET AND SYSTEM ARCHITECTURE

We first provide an overview of our data collection process conducted at two large, disjoint university campuses, namely, the University of Virginia (referred to as Campus1) and Virginia Tech (referred to as Campus2). Subsequently, we outline the key components of our beaconsing detection system to address specific challenges posed by campus networks.

3.1 Campus Network Traffic Collection

We cooperate with each university’s information security department for data collection and processing. In each network, raw border traffic is mirrored via a Gigamon network TAP (test access point) to a cluster in the secure DMZ (demilitarized zone). The mirrored traffic is then parsed by Zeek software [89] into a collection of logs. To protect user privacy, these Zeek logs are anonymized before transmission to the cluster where our beaconsing detection system runs. The data collection process is illustrated in Figure 2 and detailed in prior work [58].

Ethics consideration. The anonymization process has passed IRB review and is conducted in a secure cluster accessible by only a few designated personnel within the university (e.g., the SOC analysts). We, the researchers, are restricted to using anonymized data and cannot access the raw data. We responsibly disclose our findings to campus Security Operations Centers (SOCs).

Choice of protocols. Zeek uses dynamic protocol detection (DPD) [90] to analyze protocols instead of determining protocols based on standard ports. We conduct beaconsing detection on Zeek

HTTP.log and SSL.log that captures HTTP and TLS-based traffic. HTTP and TLS-based traffic constitutes the majority of campus traffic and are majorly used by malware C2 connections [24]. Due to the campus DNS configuration and caching mechanism (i.e., requests sent to local subnet DNS servers are invisible to security analysts), DNS logs do *not* preserve the necessary time-series pattern for beaconsing detection, and are not utilized in this work.

Traffic statistics. Table 1 summarizes the total volume of HTTP and TLS-based traffic collected in the 10-month period (2020-06-01 to 2021-03-31) for both campus networks. Due to inevitable network and machine failure, around 3% of the traffic is missing from the datasets. In total, 75.23 billion connections are collected from both campus networks.

3.2 Beaconsing Detection System Overview

Our beaconsing detection system runs in the data storage and computing center, where it receives logs daily as illustrated in Figure 2. The system has two key steps: (i) a periodicity detection component to reconstruct time series signals via global analysis and identify periodic domains, and (ii) a semi-supervised learning and ranking pipeline to detect malicious beaconsing activities among the periodic domains with limited human involvement.

Aggregation-based periodicity detection. We *aggregate* traffic per server across all devices in both campus networks to construct time series that provides a more comprehensive view than using individual devices or individual campus networks alone. We further develop a novel algorithm based on Empirical Mode Decomposition (EMD) [34] and frequency-domain signal processing techniques to compensate for the noise due to aggregation. We describe the details of the algorithm and evaluate its efficacy using both simulated signals and real-world campus traffic in Section 4.

Table 1: Total volume of traffic logs.

	Log	Log Size (gzipped)	Connections	Data Coverage
Campus1	HTTP	0.8 TB	8.92 B	96.97%
	SSL	3.8 TB	34.05 B	97.56%
Campus2	HTTP	0.8 TB	7.82 B	97.22%
	SSL	2.2 TB	24.44 B	97.22%
Total	-	7.6 TB	75.23 B	-

Semi-supervised learning and ranking. Given the sheer volume of traffic produced by two campus networks, we develop a multi-phase semi-supervised learning pipeline, including (i) a novel self-training phase to tackle the highly-imbalanced dataset and mitigate model bias, and (ii) an active-learning phase to rank beaconsing domains from the most to the least suspicious and continuously “learn” the accuracy of the top-ranked domains from various sources with minimal human involvement. We describe the pipeline in further detail in Section 5.

3.3 Threat Model

Adversaries may compromise end host machines within a target network. Infected machines (bots) periodically contact the external C&C server(s) to report their status, request instructions, or convey other information based on the logic of the installed malware. The

border routers at which our data is collected are not compromised, nor is the secure computing center where anonymized network traffic data is stored and analyzed.

4 PERIODICITY DETECTION

We describe our simple yet novel time-series aggregation algorithm in Section 4.1. To mitigate the increased noise introduced by aggregation, we innovatively employ Empirical Mode Decomposition (EMD) to smooth the signal, and then uncover the hidden periodicity using traditional Fourier analysis. We compare our proposed algorithm with several representative works using both synthetic and real-world traffic in Section 4.2, and end this section by demonstrating the efficacy of global analysis in Section 4.3.

4.1 Algorithm Description

The theory behind the success of the aggregation-based time-series analysis is that the summation of multiple discrete periodic signals, regardless of the time intervals, is still periodic [66]. This property allows us to reconstruct a comprehensive server-side temporal pattern even when the compromised hosts exhibit different periodic patterns.

Time series aggregation. The communication pattern between a client (IP) C_i and a server (FQDN) S_j is represented by an integer sequence $X_{ij} = \{x_{ij}[n]\}$, where $x_{ij}[n]$ represents the count of connections from the client C_i to the server S_j at time interval n . Let C be the set of clients connecting to server S_j , the communication behavior of server S_j is represented by $S_j = \sum_{C_i \in C} X_{ij}$. Let $S'_j(O_k)$ be the *min-max normalized* time series of server S_j in organization O_k , and O be the set of all organizations, the aggregated time series for server S_j is defined as the min-max normalization of $\sum_{O_k \in O} S'_j(O_k)$. Figure 3a shows the *fine-grained* time-series recreation examples to server *www.youtube.com* in one campus, and Figure 3b illustrates the aggregated time series of the same server across all users at both campuses.

Comparing to *fine-grained* detectors, we only use IP address as a proxy to represent a user for coding implementation and do *not* differentiate between or rely on the actual devices behind each specific IP since only server-side time series are analyzed in the following periodicity discovery process.

Pattern decomposition. While server-based aggregation overcomes the host-tracking challenge described in Section 2, it introduces larger noises. To tackle the problem, we innovatively employ EMD to de-noise the aggregated signals. Unlike common Fourier/Wavelet transform, EMD is a multi-resolution adaptive method that requires no *a priori* knowledge. Although EMD has been applied to disciplines like biomedical science [59, 61] and solar physics [38], to the best of our knowledge, we are the first bringing this algorithm to a novel usage on network data.

The key component of EMD is to decompose the raw signal into a collection of intrinsic mode functions (IMFs) through an iterative procedure called sifting [34], described as below:

- (1) At iteration n , find the local extrema in the signal $f(t)$, where $f(t)$ is the raw time series of server S_j at the initial iteration.
- (2) Fit the local maxima and minima with an upper and a lower envelope using B-spline function, denoted respectively as $E_{up}(t)$

and $E_{low}(t)$, and compute the mean envelope $E_{avg}(t)$ as the arithmetic average of $E_{up}(t)$ and $E_{low}(t)$.

- (3) Assign the residual signal as the difference between the signal and the mean envelope, i.e. $R(t) = f(t) - E_{avg}(t)$.
- (4) Check if the residual signal satisfies the stopping criterion $\sum_t \frac{(R(t) - f(t))^2}{f(t)^2} < \epsilon$, where $\epsilon = 0.05$. When satisfied, we assign the IMF $imf_n(t) \leftarrow R(t)$, $f(t) \leftarrow f(t) - imf_n(t)$, and $n = n + 1$. Otherwise, we update the signal with $f(t) \leftarrow R(t)$.
- (5) Next, we repeat steps from 1 to 4.

We adapt the sifting process to our scenario. We characterize the communication pattern of server S_j with its first extracted IMF, denoted as I_j , for subsequent analysis. Figure 3c shows an example case of extracted I_j .

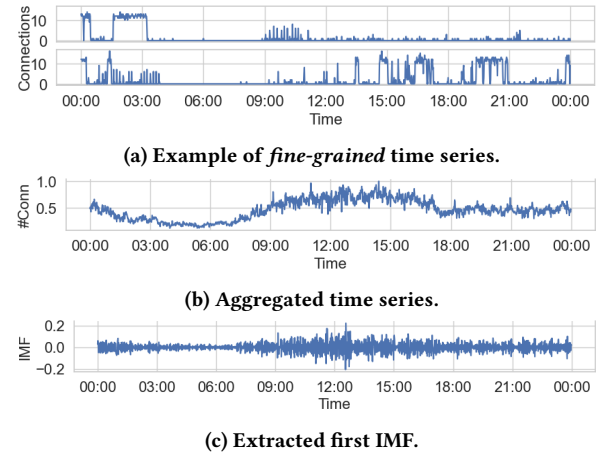


Figure 3: Time series construction, aggregation and pattern decomposition for “www.youtube.com”.

Periodicity uncovering. Next, we adopt two signal processing techniques - Discrete Fourier Transform (DFT) analysis and Auto Correlation Function (ACF) verification - to detect periodic activities [33, 64, 82].

We first employ the permutation-based approach [33] to identify candidate periods. Let I'_j be a random permutation of the original signal I_j . The random shuffling process destroys any periodicity residing in I_j , and thus even the maximum power of I'_j , denoted as p'_{max} , does not indicate any periodicity. We repeat the permutation 100 times, and define the power threshold (p_T) as the 95th largest p'_{max} (95% confidence level). Next, we extract any frequencies with power exceeding the threshold p_T in the periodogram $P(f_j)$ of the original signal I_j , and compute their corresponding period candidates \mathcal{P} . The original signal is considered non-periodic when no valid frequency is found. We further filter out period candidates caused by high frequency noise, by removing any periods $\mathcal{P}_i < I_{min}$, where I_{min} is the minimum time interval between two consecutive timestamps in the original time-series.

The final step is to verify the candidate periods \mathcal{P} using ACF that measures the similarity between a time series and a delayed copy of itself. For each candidate period $\mathcal{P}_i \in \mathcal{P}$, we consider it the true period of the original signal if its $ACF(\mathcal{P}_i)$ is the local maximum. The original signal I_j without any valid periods found after the

ACF verification process is considered as a non-periodic signal, and will be discarded.

4.2 Algorithm Evaluation

We conduct two sets of experiments (synthetic signal and real-world traffic evaluations) to compare the periodicity detection performance of *our new algorithm* with several representative studies [4, 14, 19, 32, 33, 37, 88], involving four classic methodologies (Table 2):

- *STATS-based* [88]: A statistic-based detection algorithm that computes the standard deviation and the mean of time intervals.
- *UPNSCA* [37]: A pure DFT-based beaconsing detection algorithm.
- *BAYWATCH* [33]: An algorithm combining periodogram analysis, ACF, and assumption testing.
- *RobustPeriod* [84]: A periodicity detection methodology leveraging wavelength transformation (WLTrans).

Table 2: A comparison of recent representative works.

Paper	Statistics	DFT/PSD	ACF	WLTrans	EMD
[14, 32, 88]	✓				
[4]			✓		
[19, 37]		✓			
[33]	✓	✓	✓		
[84]		✓	✓	✓	
This		✓	✓		✓

We implement *STATS-based*, *UPNSCA*, and *BAYWATCH* using Python, and use an unofficial open-source *RobustPeriod* Python implementation [63] in the following experiments.

4.2.1 Synthetic data evaluation. Due to the absence of ground truth in real-world data, we follow the standard evaluation process [33, 64, 82, 84] to assess the accuracy of the algorithm using synthesized data. We simulate three types of noises to emulate network disruptions and adversary counter-measurement strategies, e.g., injecting randomness and dropping connections, using a time series with a 10-minute period as the baseline.

- *Gaussian Noise (G)* introduces randomness by shifting the beaconsing activity time using a noise drawn from Gaussian distribution $\mathcal{N}(0, \sigma^2)$, with the standard deviation σ varying from 0% to 50% of the original periods P .
- *Insertion Noise (I)* injects random events following a Poisson process distributed around the original beaconsing activity within a time span varying from 0% to 20% of period P . We introduce this noise based on the observation [33] that, in real-world traffic, each beacon consists of a cluster of multiple connections within a short period, rather than a single connection.
- *Omission Noise* randomly drops each connections with the probability O , simulating logging failures or attackers intentionally disrupting connections to destroy periodic pattern.

We generate 100 independent time series for aforementioned noise types, and measure the *detection rate* $\gamma_d = \frac{m}{100}$ for all algorithms, where m is the number of time series successfully detected under each noise configuration.

Results. Figure 4 illustrates performance of periodicity detection algorithms against various noise levels for 10 minutes periodicity.

The detection rates of *STATS-based* and *UPNSCA* drop dramatically to 0 with a low noise level (2%) in all three setups, showing the inability of the two algorithms to identify periodicity with even tiny manipulation. Our algorithm outperforms all other algorithms in *all* experimental setups.

4.2.2 Real-world traffic evaluation. We next evaluate the above methods using one month (June 2020) of real-world HTTP traffic collected at Campus1.

Results. The unofficial implementation of *RobustPeriod* [63] typically requires around one minute to process a single time series (while ours completes the task in just 0.03 seconds). Given the daily volume of more than 58,000 HTTP FQDNs, we deem it unfeasible for practical real-world applications and therefore exclude it from our assessment of real-world traffic. Our method detects **13,839** unique periodic FQDNs (27.6% more as compared to *BAYWATCH*). Both *STATS-based* and *UPNSCA* detect **0** periodic domains suggesting their incapability of detecting periodicity in noisy real-world traffic.

Case study. Figure 5 shows the real-world beaconsing activities from a malvertising campaign discovered in Campus1 (details in Section 7), where our algorithm detects either a 4-minute or 10-minute periodicity for the campaign-related domains. We note that these time intervals are not strictly periodic but rather distributed around the actual periodicity. This observation highlights the importance of designing noise-tolerant periodicity detection algorithm and further demonstrates the robustness of our system.

Takeaway. Our periodicity detection algorithm outperform recent and classic periodicity detection methods in both experiments, demonstrating its effectiveness in identifying periodic signals in real-world campus network.

4.3 Efficacy of Global Analysis

We demonstrate the effectiveness of global analysis by running our proposed algorithm on daily HTTP and TLS traffic collected at two campus networks from June 2020 to April 2021. Note that we focus on detecting *all* periodic domains instead of *malicious* ones in this section. We will identify malicious periodic domains in Section 6.

Dataset notations. We construct four datasets as illustrated in Figure 6. In "Local" (LC1/LC2), traffic is analyzed separately at the two campus networks. In "Global" (GC1/GC2), traffic is aggregated on shared domains (if any) across the campus networks. Take GC1 dataset as an example, GC1 contains time series for domains (i) only observed at Campus 1 and (ii) observed at both Campus 1 and 2, where the traffic is aggregated to construct the time series. Note that the GC1 dataset has the same number of domains as the LC1 dataset, but each domain's global time-series may differ from the local time-series due to the process of aggregation.

Experimental setup. We deploy our proposed method on time series reconstructed in aforementioned four network datasets. To mimic the daily operations of a SOC, we collect and analyze the data every 24 hours during the 10 months period (June 2020 to April 2021). Our periodicity detector is deployed in a virtual machine with a 16-core 2.1GHz Intel Xeon Processor and 64GB memory. On a daily basis, the entire detection takes around 2 hours to identify periodic domains at both campuses.

Results. As shown in Table 3, we detect significantly more periodic domains in the "Global" datasets where time series are

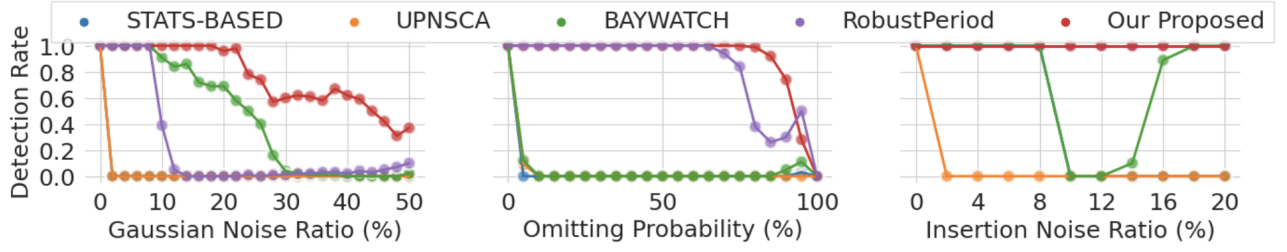


Figure 4: Comparison with existing algorithms. All subfigures have a shared y-axis representing the detection rate. In each of the three configurations, both STATS-based and UPNSCA detection rates exhibit a significant decline when subjected to low noise levels (2%). In the insertion noise simulation (on the right), both RobustPeriod and our proposed method achieve a 100% detection rate. However, our algorithm excels in accurately identifying the periodicity as 10 minutes, while RobustPeriod erroneously reports it as 5 minutes.

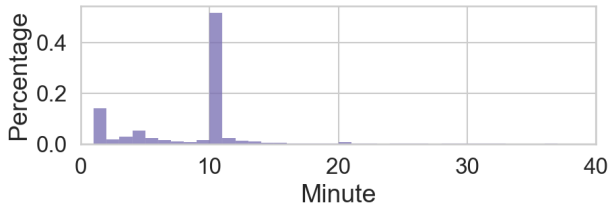


Figure 5: Time intervals observed in a malvertising cluster.

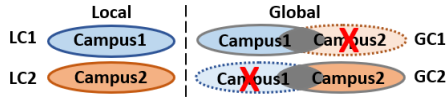


Figure 6: Dataset notation. Domains observed exclusively in one campus is not shared to the other campus.

aggregated across two campuses. Specifically, we detect **43.13%** and **66.48%** more periodic domains in GC1 and GC2, respectively, as compared to using time series from a single local network. Note

Table 3: Average count of distinct FQDNs per day.

	LC1	LC2	GC1	GC2
Periodic	12,246	9,190	17,528	15,310
Total	514,777	357,644	514,777	357,644

that domains observed exclusively in one campus are not shared with the other campus. Therefore, the global datasets (GC1/ GC2) contains the same number of FQDNs as the local ones (LC1/ LC2). The *global* detection benefits from the aggregation of FQDNs that are *mutually* visited by the two campus networks. We quantify the improvement by measuring the maximum spectrum power of time-series in LC1 and GC1 datasets. We find that more signals in the global dataset have higher spectral power (Figure 7), indicating a stronger dominant frequency in the signal and a higher probability that the signal is periodic. This suggests that the time-series at each individual network is likely inaccurate due to the incomplete profile of network communication patterns.

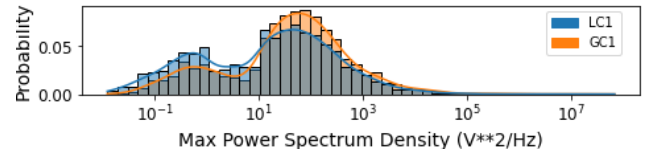


Figure 7: Maximum spectrum power comparison.

Takeaway. By aggregating signals across campus networks, we are able to detect 43% and 66% more periodic domains compared to detecting within Campus 1 or Campus 2 alone, respectively, suggesting the effectiveness of global analysis.

5 LEARNING AND RANKING PIPELINE

From Section 4.2.2, we see that more than 15K periodic FQDNs are detected daily in both campus networks. As it is impractical to manually investigate such a large number of cases for malicious activities, we develop a semi-supervised learning and ranking pipeline that ranks daily periodic domains from the most to the least suspicious, addressing challenges below.

Limited ground-truth labels. Given that real-world campus traffic lacks ground truth labels [30], we use VirusTotal [81] as an alternative to label a small portion of the dataset. In particular, we sample and query about 15% of all periodic domains in the starting phase of our training pipeline. VirusTotal is a threat intelligent platform that provides integrated malware labels, and is widely adopted in research community [93]. Lacking ground-truth labels is a common issue in the security domain that downgrades model performance [1]. To this end, we develop an *active-learning* pipeline to continuously learn from experts and improve model performance throughout the time.

Highly-imbalanced datasets. Network data is intrinsically highly-imbalanced; it mostly comprises benign or unknown connections with very few confirmed malicious cases. In total, we obtain more than 200K valid responses from VirusTotal, of which only 2.8% were flagged as malicious. Such highly-imbalanced datasets lead to critical training issues like model bias in favor of the majority (benign) class. We tackle this challenge by introducing a *self-training* phase prior to the active-learning pipeline in order to automatically re-balance the label distribution.

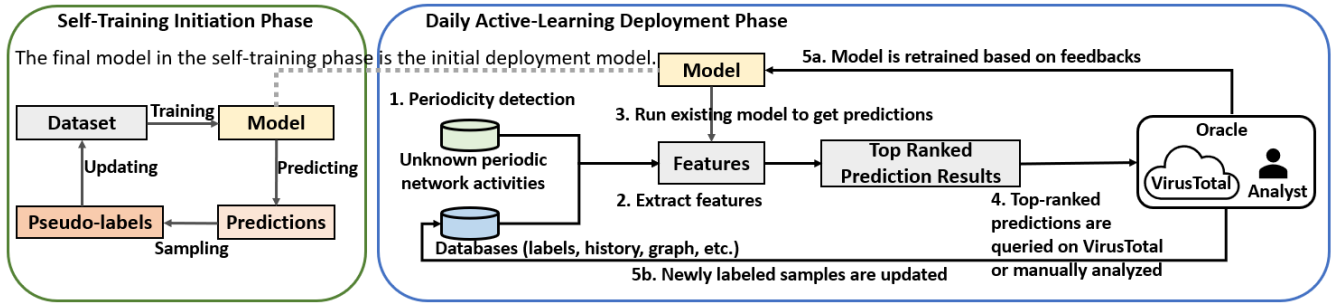


Figure 8: Two-phase learning and ranking pipeline. In the initiation phase, we employ self-training technique [83] with our new sampling strategy. We then deploy the model trained in self-training phase in active-learning framework to continuously improve the detection performance.

5.1 Feature Extraction

We integrate historical and topological data, along with our novel periodicity-based features, to provide a distinctive understanding of domain behavior and extract a comprehensive set of 70 features (elaborated in Appendix D), which we explain as follows:

- **Periodicity-based features.** To the best of our knowledge, we are the first to track the reputation of each periodicity and compute the overall statistics to depict behavior of domains with similar temporal patterns.
- **Graph-based features.** We utilize the Neo4j database [57] to construct a network graph by connecting periodic FQDNs via shared IPs and second-level domain names. For each periodic FQDN, its graph-based features summarize the neighboring behavior, such as the topological distance between one domain and its nearest malicious neighbors.
- **Historical features.** For both graph-based and periodicity-based features, we measure their historical statistics using a 10-day and 30-day sliding window respectively. These window sizes are determined to maintain the databases in a reasonable size without degrading the computing performance.
- **Other features.** We also extract widely adopted features such as entropy and domain popularity scores.

5.2 Pipeline Description

An overview of our two-phase learning and ranking pipeline for detecting malicious beaconsing activities is illustrated in Figure 8. In the initial phase, we employ CReST [83], a self-training technique, along with our novel sampling strategies, to address the imbalanced dataset challenge. In the subsequent phase, we implement an active-learning framework, typically used when ground truth labels are limited [70], to continuously update the model and detect daily malicious beaconsing activities.

Model definition. We use a 4-class random forest classifier [15] in both phases. Each class corresponds to the number of VirusTotal engines ($\#MalEng$) that detects a specific domain as malicious (i.e., classes 0–3). All domains with $\#MalEng \geq 3$ are classified as class 3. Given our observation that some benign websites are mis-classified as $\#MalEng = 1$ on VirusTotal, we consider a domain *malicious* when its $\#MalEng \geq 2$. In other words, the higher the $\#MalEng$ is, the more likely that the domain is malicious.

Self-training phase. We employ CReST [83] training process as shown in Figure 8 on the left. Our initial dataset consists of 10-day periodic domains, comprising a total of 44K unique FQDNs, out of which 106 domains are malicious based on labels from VirusTotal ($\#MalEng \geq 2$). Prior study [83] observes that training the model on a highly-imbalanced dataset generates high-precision predictions for the minority classes. Utilizing this property, we improve the model performance by iteratively retraining it with “pseudo-labels” sampled from predictions of non-dominant classes. To mitigate mislabeling issues of VirusTotal, we develop a new sampling strategy: (i) the sampling rate $\alpha = 1$ is applied to predictions with $\#MalEng \geq 2$, as larger classes indicate a more malicious behavior, (ii) $\alpha = 0.05$ is used for non-benign samples (i.e. class 1), and (iii) $\alpha = 0$ is set for benign samples (i.e. class 0). We iteratively train the model till no further “pseudo-labels” are predicted or sampled, and deploy the trained model in the next active-learning phase. We demonstrate the efficacy of our self-training process in Appendix A.

Active-learning phase. We design a daily active-learning framework (Figure 8 on the right) to reduce model performance degradation caused by the lack of ground-truth labels. The iterative process is: (i) we perform periodicity detection on collected network data daily, (ii) we extract features of the periodic FQDNs and run the model for prediction, (iii) based on prediction results, we rank the FQDNs and send top-ranked suspicious ones to the oracle for feedback, and (iv) we retrain the model and update databases based on the feedback. Specifically, the ranking algorithm is based on two criteria: (i) the prediction probability of an FQDN being malicious (predicted as class 2 or 3), and (ii) whether this FQDN has been seen and verified in the past. For example, a *new* FQDN predicted as class 3 with high model confidence will be assigned the highest priority when sent to the oracle. Our oracle consists of automated VirusTotal querying and manual verification, performed by two domain experts. When the prediction result differs from the VirusTotal label (e.g., predicted as malicious while labeled as benign on VirusTotal), the case will be sent to the domain experts for manual verification through online security reports such as MalwareBytes [53] and ThreatCrowd [79]. The two domain experts will compare their findings for consistency. We evaluate the pipeline including verification process in Section 6.1.

In summary, self-training phase is performed only once during the initiation, while active-learning and ranking pipeline are executed and updated continuously each day.

6 EVALUATION

We evaluate our beaconing detection system on 10 months (June 2020 to April 2021) traffic from two campus networks in both local and global setups, as follows:

- **User-centric performance:** Given the resource constraint in real-world SOC operations, the primary goal is to minimize False Positives and to maintain a reasonable number of cases for manual verification. In other words, it is more important to prioritize truly malicious activities for SOC analysts instead of targeting to minimize False Negatives. Such preference has been confirmed in recent studies that interviewed security analysts [3, 39].
- **Overall model performance:** we supplement the above evaluation by obtaining labels for all domains, which enables evaluation on overall model performance using metrics including *accuracy*, *precision* and *recall*. However, given the sheer amount of traffic, it is unrealistic to query all domains on VirusTotal for the whole 10 months. Thus, we randomly sampled 10% data from Jan. to Apr. 2021 (last 3 months of evaluation where the model is more stabilized) for this evaluation.

In addition, we analyze the model performance with its most important features to provide further insights. Furthermore, we quantify the pipeline's capability on detecting *new* malicious domains ahead of VirusTotal (Section 6.2).

Note that we do not compare our final results to the aforementioned four periodicity detection algorithms. STATS [88] and UPNSCA [37] fail to identify any beaconing domains in real-world campus network; BAYWATCH [33] uses HTTP URL to classify malicious activities, making it infeasible for TLS traffic (the dominant traffic in campus); and RobustPeriod [84] does not perform any anomaly detection and runs way too slow on large volume datasets.

6.1 Global Pipeline v.s. Local Pipeline

We run two active-learning pipelines on our datasets, defined as:

- *Global pipeline:* aggregating traffic signals and computing features using global datasets (GC1/GC2).
- *Local pipeline:* aggregating traffic signals and computing features only *within* each campus network, i.e., LC1/LC2.

We use 20 days of traffic to build graph database, and 10 days for self-training initiation. Starting 2020-07-01, the daily traffic is processed through the two active-learning pipelines.

Runtime. Zeek logs are generated in real time at the network border, anonymized and transferred to the data center on a daily basis, where we perform our daily analysis in 3-4 hours (2 hours for periodicity detection, and 1 hour for feature generation and model prediction).

Results. Figure 9 shows the detection results for both local (LC1) and global (GC1) pipelines during the 9-month period in Campus1. We observe similar results for Campus 2, which is shown in Appendix B. Three drops (2020-09-26, 2020-12-05, and 2020-12-23) in the figure are due to network/machine failures during the data collection. We also notice the lower volume of traffic during summer and winter breaks.

Each line in the graph corresponds to the following:

- (Global) Detected cases: total number of detected domains per day using the global pipeline.
- (Global/Local) Malicious cases (verified): among the detected cases, the number of cases that have been verified as malicious on VirusTotal as of March 2022.
- (Global/Local) Malicious cases (original): among the detected cases, the number of cases that were labeled as malicious on VirusTotal on or close to the day of detection.

User-centric performance (detected cases and FP). Table 4 summarizes average daily statistics of detection results for both *local* and *global* pipelines. The *global pipeline* detects an average of 65.32% more malicious domains per day than the standalone *local pipeline* with an accuracy of 93.49%. On a daily average, 3-5 false positive cases in top-ranked results cannot be verified by the time of detection, corresponding to 7% and 6% of top-ranked alerts in *local* and *global* pipelines, respectively.

At the end of evaluation, the *local pipeline* detects 1,147 unique malicious domains, among which 610 (53.1%) were *not* flagged on VirusTotal *at the time of detection* but are updated as malicious later in time. The *global pipeline* identifies 1,387 unique malicious domains, among which 781 (56.3%) were *not* flagged on VirusTotal *at the time of detection* but are updated as malicious later. In total, the *global pipeline* detects 240 unique malicious domains that are invisible in the single network evaluation.

Table 4: Average daily detection.

	Local Pipeline	Global Pipeline	Diff.
Detected	46.93	77.15	30.22
Malicious (Original)	28.79	42.44	13.65
Malicious (Verified)	14.84	29.69	10.22
Malicious (Total)	43.63	72.13	28.50
Unknown	3.30	5.02	1.72
Accuracy	92.97%	93.49%	-

User-centric performance (cases for manual verification).

We measure the number of cases in the *global pipeline* that require manual verification to evaluate the level of necessary human involvement in our experiments. On average, 10 cases are reported to analysts (oracle) for further investigation per day. Throughout the experiments, on 73% of the days, the pipeline generates 10 or fewer cases for manual verification.

Table 5: Global pipeline scores.

	Accuracy	Precision	Recall
Original Analysis	0.9957	0.9464	0.9969
Retrospective Analysis	0.9701	0.9675	0.9701

Overall model performance. We report model performance in Table 5. As mentioned previously, we randomly sampled 10% of all domains from the last 3 months (Jan-Apr 2021) to query VirusTotal due to the sheer number of domains in our dataset. *Original analysis* shows the average daily model scores of the *global pipeline* model based on VirusTotal query results during Jan-Apr 2021. We further conduct a *retrospective analysis* in December 2022, roughly 2 years after the original detection, to re-query the same set of domains.

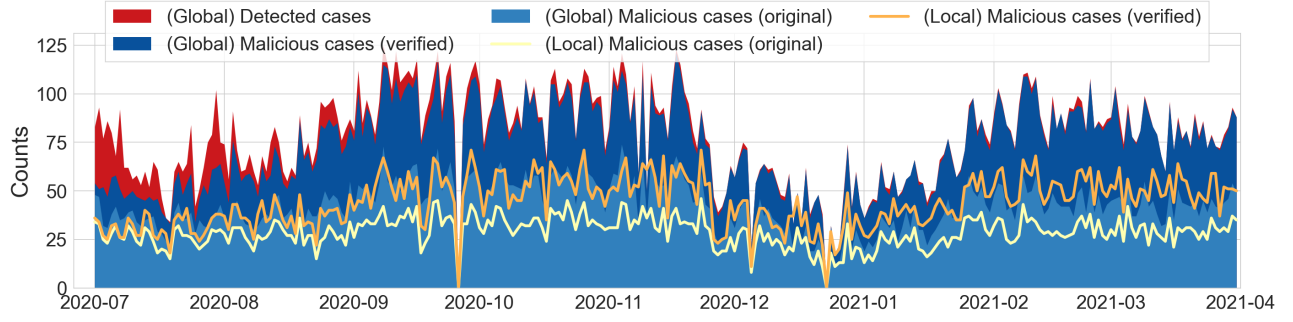


Figure 9: Comparison of active-learning and ranking based detection. X-axis shows the date, and y-axis shows the count of FQDNs. Three drops (2020-09-26, 2020-12-05, and 2020-12-23) are due to network/machine failures during the data collection. The red area illustrates the daily detected cases by the *global pipeline*. The light blue area depicts the count of cases that can be verified on or close to the day of detection, and the dark blue area shows the amount of cases that can be verified as of March 2022. Yellow and orange lines are the baseline results in the *local pipeline*.

We can see that the model scores remain high. Although only 10% data is sampled, we calculate the margin of error below 1% with 95% confidence interval.

Table 6: Most important 10 features.

Rank	Global	Local
1	avglen2malFQDN	avgMalEng
2	minlen2malFQDN	maxMalEng
3	maxlen2malFQDN	max_ipDomMalEng
4	dom_level	avglen2malFQDN
5	subdom_entropy	minMalEng
6	dom_subcnt	minlen2malFQDN
7	occ	malEng_ratio
8	freq	maxMalEng
9	cisco_max_period	malFQDN_ratio
10	hist_malscore_mean_period	cntFQDN

*Detailed feature descriptions can be found in Appendix D.

Model interpretation. Table 6 lists the key features ranked by their *Gini* importance. The features presented in bold depict domain behavior at a *broader* topological and temporal scale, while the remaining features focus more on small-scale behavior. For instance, *avglen2malFQDN* measures the average topological distance between one periodic FQDN and all connected malicious nodes within four hops in the graph database. On the other hand, *avg-MaxEng* only calculates the maliciousness of FQDNs that share the same second-level domains (one-hop neighbors). The *global* model also assigns greater weights on the periodicity features such as *cisco_max_period* and *hist_malscore_mean_period* that measure the popularity and historical maliciousness of the detected domain's period. Given that the *global* pipeline significantly outperforms the *local* pipeline, these observations emphasize the importance on characterizing topological and temporal patterns of domains for enhanced identification of suspicious beaconsing activities.

6.2 Assessing VirusTotal's Searching Delay

Previous research [60] illuminate the differentiation between VirusTotal's querying and scanning mechanisms, while also quantifying the inherent delay within VirusTotal through active scanning of

38 phishing websites. More specifically, a search request involves querying the VirusTotal database for its latest scan results, whereas a scan request triggers vendors to scan the requested website. Nevertheless, conducting active scans for every domain becomes impractical within a campus environment, primarily due to the substantial volume of network traffic and budgetary constraints. In practice, campus Security Operations Centers (SOCs) typically commence their investigations by first identifying suspicious targets and subsequently performing VirusTotal searches to gather additional information.

The operational dependence on VirusTotal has prompted us to investigate the extent of detection delay when using VirusTotal's search mechanism.

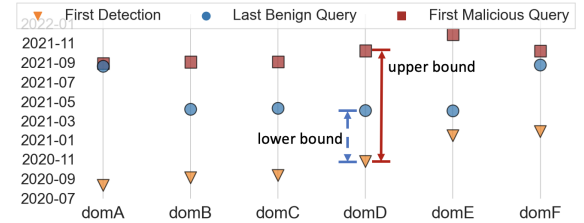


Figure 10: Example of early detection. We use short names for domains here and show actual domain names in Table 8.

Methodology. To answer the forementioned questions, we retrieve three important dates of domains that fall within the category represented by the dark blue area in Figure 9. These domains were not initially identified as malicious by VirusTotal at or immediately after our system's detection, but were later updated as malicious. We explain the three dates as follows:

- **First Detection Date:** the date when our system first detected the domain as malicious.
- **Last Benign Query Date:** the latest date that we receive a benign label from VirusTotal for a given domain.
- **First Malicious Query Date:** the date when we initially observe the domain classified as malicious on VirusTotal.

We acknowledge that acquiring the precise date when a domain shifts from benign to malicious on VirusTotal poses challenges, primarily because we are unable to perform daily queries continuously due to query limitations.

With the three important dates, we can compute the *upper bound* and the *lower bound* to provide an estimate on the number of days that VirusTotal lags behind our system. This estimation is presented in Figure 10 and is described as follows:

- *Lower bound*: the difference between the *First Detection Date* (yellow triangle) and the *Last Benign Query Date* (blue circle).
 - *Upper bound*: the difference between the *First Detection Date* (yellow triangle) and *First Malicious Query Date* (red square).
- In other words, for a given malicious domain, the *lower bound* measures the minimum number of days that our system detects ahead of VirusTotal, while the *upper bound* measures the maximum.

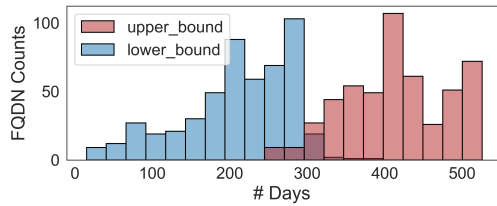


Figure 11: Searching delays of VirusTotal.

Results. We successfully retrieved the three important dates for 509 distinct malicious domains detected by our system. Our failure on retrieving the remaining cases is due to the lack of *Last Benign Query* as those domains return with a “malicious” label in the first query on VirusTotal. Table 7 shows statistics on the *upper* and *lower bound*, and Figure 11 shows the histogram for all 509 domains. On average, the number of days that VirusTotal lags behind our system falls between 211 and 414 days.

Table 7: Statistics on the *upper* and *lower bound*.

	Min.	Max.	Mean	Stdev.
Lower Bound	15	374	211.44	70.35
Upper Bound	248	526	414.45	65.63

Takeaway. VirusTotal’s search mechanism exhibits an average delay of at least 211 days in comparison to our system.

7 CASE STUDIES

We now take a deeper look into malicious cases detected by our system. We start with dissecting the 6 example domains shown in Figure 10 that are detected by our system earlier than VirusTotal, shown in Table 8. We find that they belong to different categories or malware families (identified by online threat intelligence platforms), with the top two being *NjRAT* and *Malvertising*, which we investigate next.

Malvertising network. Our system detects 46 domains belonging to the same malvertising cluster. The 46 domains periodically connect to two IPs, 216.21.13.14 and 216.21.13.15, which are found to be malicious by threat intelligence platforms [46, 47]. All the 46

Table 8: Verified malicious detections.

Short	Domain Name	#. MalENG	Category
domC	regul[...].anr{dot}top	8	NjRAT
domE	omareeper{dot}com	7	NjRAT
domD	dmnsg{dot}com	5	Malvertising
domB	outseeltor{dot}com	3	Fraud
domA	catest[...].ray{dot}com	3	Adware
domF	6v5f3l{dot}com	1	Browser hijacker

domains are DGA-formatted, with an average lifetime of 1 day. The complete set of domain names are shown in Appendix C Table 10. Our system successfully uncovers either a 4-minute or 10-minute periodicity for the majority of these domains, as discussed in Section 4.2, demonstrating the robustness of our system in detecting periodicity even in the presence of noises.

NjRAT. We detected an NjRAT [45] and riskware-involved cluster, consisting of 65 distinct malicious domains and 52 distinct IPs in 4 subnets (/24) hosted in Netherlands and the United Kingdom. Among the 65 domains, 30 (46%) are already identified as malicious by VirusTotal since the beginning, while 33 (51%) are first detected by our pipeline and later verified through VirusTotal and manual analysis. We show the detailed cluster in Figure 14 in Appendix C.

8 RELATED WORK

Prior work focusing on detecting botnet C&C communications through periodic behavior fall into three categories: statistical analysis, spectral analysis, and machine-learning based approaches with temporal features. Statistical analysis [25, 28, 29, 32, 72] detects beaconing activities by computing statistics of time intervals between consecutive connections, whereas spectral studies [4, 6, 33, 37] analyze the time-series in frequency domains utilizing DFT and periodogram. Researches that adopt machine-learning techniques [12, 14, 51, 78] extract features from communication patterns and combine artificial intelligence to classify periodic activity. Besides beaconing, periodicity detection has also been widely studied in signal processing and data mining areas to identify similarities in data over time [21, 22, 51, 52, 64].

Though the lack of ground-truth labels remains a serious but unexplored issue in network security, similar problems have drawn attention in semi-supervised learning area. Assuming that the ground-truth labels can always be obtained from teachers (oracles), researchers studying active-learning proposed various querying strategies to improve model performance [20, 31, 41, 44, 70, 71], from which some techniques have also been adopted in network intrusion detection and binary analysis [26, 27, 50, 69, 80, 87]. However, none of above studies demonstrate the model capability in dealing with enormous amount of data captured from large-scale campus networks. Meanwhile, other semi-supervised learning approaches [10, 11, 48, 49, 55, 67, 73, 85, 86] developed strategies such as *pseudo-labels* to improve data quality without reaching to external resources. Although these algorithms have made significant improvements in other research areas like computer vision, careful design is required to adopt such techniques in network attack detection. Besides above techniques, unsupervised classic clustering methods, such as K-Nearest Neighbors (KNN), are also commonly

used to detect suspicious network activities in case of insufficient labels [2, 68, 76, 77].

9 DISCUSSIONS

Applicability. Although our system is evaluated on Zeek HTTP and SSL logs, it is generally applicable to other types of log and traffic, e.g., UDP [74], as the core component of our beaconsing detection is based on time-series analysis on aggregated signals. The beaconsing detection algorithm can also be adapted to various sampling frequencies and time scales to conduct hourly, daily, or even weekly detection.

DGA. Attackers can leverage Domain generation algorithm (DGA) to evade beaconsing detection. In an extreme case, the C&C server may use different hostnames for each connection, forcing the system to generate a new time-series for each DGA domain. However, such aggressive DGA behavior is rare in real-world scenarios as it increases the probability of C&C servers being detected. In fact, researchers [13] have found that DGA domains typically last for 1.2 days on average, which is more than sufficient for our detection.

Privacy concerns in sharing campus traffic. While we have successfully arranged data sharing between two campuses and we believe such agreement can be extended to include more campuses, we understand that some campuses may not be willing to share raw Zeek logs due to privacy concerns. Our detection relies on *aggregated time series* for external FQDNs, which means that campuses can run our aggregation algorithm internally first, and only share the aggregated time series which do not reveal any information on individual connections. Such time series will then be aggregated further across campuses.

Limitations. Our beaconsing detection relies on the presence of periodic beaconsing activities (despite being incomplete, irregular, or noisy), which is common in C&C communications. We cannot detect *all* possible malicious domains or bot activities, especially when the C2 communication is completely random. However, we argue that our system is an effective approach in many cases: 1) periodic beaconsing is common in the majority of the botnets and malwares [36], and 2) time-series analysis is a general method that do not heavily rely on traffic metadata that require deep packet inspection (DPI), which usually is not possible at campus networks.

Another limitation of the system is the reliance on the accuracy of the “teacher” labels during the active-learning pipeline. Although we have done our best in performing manual investigation when there is discrepancy between the detection and the VirusTotal labels, we are limited by the information we have access to and there may still exist cases which are falsely labeled. One potential improvement is tighter collaboration with the SOC analysts, who have access to more restricted/protected data that can help provide more context to determine whether a newly-observed suspicious activity is indeed malicious.

10 CONCLUSION

Our novel approach for detecting and ranking malicious beaconsing activities in large campus networks by utilizing aggregated signals and innovative time-series analysis has demonstrated remarkable efficacy. Through extensive evaluation in real-world scenarios, we

identified a significant increase in beaconsing detection when compared to single-network analysis. We believe that our work sheds light on the efficacy of global analysis and we hope to draw the attention of the community towards building a data sharing and defending consortium.

ACKNOWLEDGMENTS

We thank the anonymous referees for their valuable comments and helpful suggestions.

This work is supported by the National Science Foundation (NSF) under Grant No. CNS-2154962 and CNS-2319421. The research reported in this document/presentation was performed in connection with contract number W911NF18-C-0019 with the U.S. Army Contracting Command - Aberdeen Proving Ground (ACC-APG) and the Defense Advanced Research Projects Agency (DARPA). The views and conclusions contained in this document/presentation are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of ACC-APG, DARPA, or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer’s or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

- [1] Sebastian Abt and Harald Baier. 2014. Are we missing labels? A study of the availability of ground-truth in network security research. In *2014 third international workshop on building analysis datasets and gathering experience returns for security (badgers)*. IEEE, 40–55.
- [2] Mohiuddin Ahmed and Abdun Naser Mahmood. 2015. Network traffic pattern analysis using improved information theoretic co-clustering based collective anomaly detection. In *International Conference on Security and Privacy in Communication Networks: 10th International ICST Conference, SecureComm 2014, Beijing, China, September 24–26, 2014, Revised Selected Papers, Part II* 10. Springer, 204–219.
- [3] Bushra A Alahmadi, Louise Axon, and Ivan Martinovic. 2022. 99% False Positives: A Qualitative Study of {SOC} Analysts’ Perspectives on Security Alarms. In *31st USENIX Security Symposium (USENIX Security 22)*. 2783–2800.
- [4] Giovanni Apruzzese, Mirco Marchetti, Michele Colajanni, Gabriele Gambigliani Zoccoli, and Alessandro Guido. 2017. Identifying malicious hosts involved in periodic communications. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*. IEEE, 1–8.
- [5] Ionut Arghire. 2022. QBot Malware Infects Over 800 Corporate Users in New, Ongoing Campaign. <https://www.securityweek.com/qbot-malware-infects-over-800-corporate-users-new-ongoing-campaign/>.
- [6] Basil AsSadhan, José MF Moura, and David Lapsley. 2009. Periodic behavior in botnet command and control channels traffic. In *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, 1–6.
- [7] Avast. 2023. Andromeda under the microscope. <https://blog.avast.com/andromeda-under-the-microscope>.
- [8] Avast. 2023. The Zeus Trojan — What It Is, and How to Remove and Prevent it. <https://www.avast.com/c-zeus>.
- [9] Michael Bailey, Evan Cooke, Farnam Jahanian, Yunjing Xu, and Manish Karir. 2009. A survey of botnet technology and defenses. In *2009 Cybersecurity Applications & Technology Conference for Homeland Security*. IEEE, 299–304.
- [10] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. 2019. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785* (2019).
- [11] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249* (2019).
- [12] Leyla Bilge, Davide Balzarotti, William Robertson, Engin Kirda, and Christopher Kruegel. 2012. Disclosure: Detecting Botnet Command and Control Servers through Large-Scale NetFlow Analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference (Orlando, Florida, USA) (ACSAC '12)*. Association for Computing Machinery, New York, NY, USA, 129–138. <https://doi.org/10.1145/2420950.2420969>

- [13] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. 2011. EX-POSURE: Finding Malicious Domains Using Passive DNS Analysis.. In *Ndss*. 1–17.
- [14] Yessine Borchani. 2020. Advanced malicious beaconing detection through AI. *Network Security* 2020, 3 (2020), 8–14. [https://doi.org/10.1016/S1353-4858\(20\)30030-1](https://doi.org/10.1016/S1353-4858(20)30030-1)
- [15] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.
- [16] CISA. 2020. Qbot/Qakbot Malware Report. <https://www.cisa.gov/stopransomware/qbotqakbot-malware-report>.
- [17] Angelo Comazetto. 2011. *Botnets: The dark side of cloud computing*. Technical Report. Technical Report, Boston, USA.
- [18] AT&T Cybersecurity. 2021. Stories from the SOC – Beaconing Activity. <https://cybersecurity.att.com/blogs/security-essentials/stories-from-the-soc-beaconing-activity>.
- [19] Yael Daihes, Hen Tzaban, Asaf Nadler, and Asaf Shabtai. 2021. MORTON: Detection of Malicious Routines in Large-Scale DNS Traffic. In *European Symposium on Research in Computer Security*. Springer, 736–756.
- [20] Sanjoy Dasgupta. 2005. Analysis of a greedy active learning strategy. *Advances in neural information processing systems* 17 (2005), 337–344.
- [21] Mohamed G Elfeky, Walid G Aref, and Ahmed K Elmagarmid. 2005. Periodicity detection in time series databases. *IEEE Transactions on Knowledge and Data Engineering* 17, 7 (2005), 875–887.
- [22] Mohamed G Elfeky, Walid G Aref, and Ahmed K Elmagarmid. 2005. WARP: time warping for periodicity detection. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 8–pp.
- [23] Center for Internet Security. 2022. Top 10 Malware September 2022. <https://www.cisecurity.org/insights/blog/top-10-malware-september-2022>.
- [24] Sean Gallagher. 2021. Nearly half of malware now use TLS to conceal communications. <https://news.sophos.com/en-us/2021/04/21/nearly-half-of-malware-now-use-tls-to-conceal-communications>.
- [25] Frederic Giroire, Jaideep Chandrashekar, Nina Taft, Eve Schooler, and Dina Papagiannaki. 2009. Exploiting temporal persistence to detect covert botnet channels. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 326–345.
- [26] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. 2009. Active learning for network intrusion detection. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*. 47–54.
- [27] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. 2009. Active Learning for Network Intrusion Detection. In *Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence (Chicago, Illinois, USA) (AISec '09)*. Association for Computing Machinery, New York, NY, USA, 47–54. <https://doi.org/10.1145/1654988.1655002>
- [28] Guofei Gu, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, and Wenke Lee. 2007. Bothunter: Detecting malware infection through ids-driven dialog correlation.. In *USENIX Security Symposium*, Vol. 7. 1–16.
- [29] Guofei Gu, Junjie Zhang, and Wenke Lee. 2008. BotSniffer: Detecting botnet command and control channels in network traffic. (2008).
- [30] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. 2022. Datasets are not enough: Challenges in labeling network traffic. *Computers & Security* 120 (2022), 102810.
- [31] Yuhong Guo and Russell Greiner. 2007. Optimistic active-learning using mutual information.. In *IJCAI*, Vol. 7. 823–829.
- [32] Mackenzie Haffey, Martin Arlitt, and Carey Williamson. 2018. Modeling, analysis, and characterization of periodic traffic on a campus edge network. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 170–182.
- [33] Xin Hu, Jiyong Jang, Marc Ph Stoecklin, Ting Wang, Douglas L Schales, Dhilung Kirat, and Josyula R Rao. 2016. BAYWATCH: robust beaconing detection to identify infected hosts in large-scale enterprise networks. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 479–490.
- [34] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu. 1998. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454, 1971 (March 1998), 903–995. <https://doi.org/10.1098/rspa.1998.0193>
- [35] Dan Hubbard. 2022. Cisco Umbrella, The Cisco Umbrella 1 Million. <https://umbrella.cisco.com/blog/cisco-umbrella-1-million/>
- [36] Ngoc Anh Huynh, Wee Keong Ng, and Hoang Giang Do. 2016. On periodic behavior of malware: experiments, opportunities and challenges. In *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 1–8.
- [37] Ngoc Anh Huynh, Wee Keong Ng, Alex Ulmer, and Jörn Kohlhammer. 2016. Uncovering periodic network signals of cyber attacks. In *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*. IEEE, 1–8.
- [38] Hybrid-Analysis. 2023. Hybrid-Analysis. <https://www.hybrid-analysis.com>.
- [39] SANS Institute. 2019. Common and Best Practices for Security Operations Centers: Results of the 2019 SOC Survey. <https://www.sans.org/media/analyst-program/common-practices-security-operations-centers-results-2019-soc-survey-39060.pdf>.
- [40] Interpol. 2020. ASEAN Cyberthreat Assessment 2020. https://www.interpol.int/content/download/14922/file/ASEAN_CyberThreatAssessment_2020.pdf.
- [41] Rong Jin and Luo Si. 2012. A bayesian approach toward active learning for collaborative filtering. *arXiv preprint arXiv:1207.4146* (2012).
- [42] Kaspersky. Last accessed: 2023. What's behind APT29? <https://www.kaspersky.com/enterprise-security/mitre/apt29>.
- [43] Sheharbano Khattak, Naurin Rasheed Ramay, Kamran Riaz Khan, Affan A. Syed, and Syed Ali Khayam. 2014. A Taxonomy of Botnet Behavior, Detection, and Defense. *IEEE Communications Surveys Tutorials* 16, 2 (2014), 898–924. <https://doi.org/10.1109/SURV.2013.091213.00134>
- [44] Punit Kumar and Atul Gupta. 2020. Active learning query strategies for classification, regression, and clustering: a survey. *Journal of Computer Science and Technology* 35, 4 (2020), 913–945.
- [45] Malwarebytes Labs. 2023. Backdoor:NJRat. <https://blog.malwarebytes.com/detections/216-21-13-15/>.
- [46] Malwarebytes Labs. 2023. Detections, 216.21.13.14. <https://blog.malwarebytes.com/detections/216-21-13-14/>.
- [47] Malwarebytes Labs. 2023. Detections, 216.21.13.15. <https://blog.malwarebytes.com/detections/216-21-13-15/>.
- [48] Samuli Laine and Timo Aila. 2016. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242* (2016).
- [49] Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, Vol. 3. 896.
- [50] Yang Li and Li Guo. 2007. An active learning based TCM-KNN algorithm for supervised network intrusion detection. *Computers & security* 26, 7–8 (2007), 459–467.
- [51] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. 2010. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1099–1108.
- [52] Zhenhui Li, Jingjing Wang, and Jiawei Han. 2012. Mining event periodicity from incomplete observations. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 444–452.
- [53] MalwareBytes. 2023. MalwareBytes. <https://www.malwarebytes.com/>.
- [54] Trend Micro. 2014. Threat Encyclopedia - CONFICKER. <https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/conficker>.
- [55] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.
- [56] Jose Nazario and Thorsten Holz. 2008. As the net churns: Fast-flux botnet observations. In *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 24–31.
- [57] Neo4j. 2023. Neo4j Graph Data Platform. <https://www.neo4j.com/>.
- [58] Alastair Nottingham, Molly Buchanan, Mark Gardner, Jason Hiser, and Jack Davidson. 2022. Sentinel: A Multi-institution Enterprise Scale Platform for Data-driven Cybersecurity Research. In *2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 252–257.
- [59] Ram Bilas Pachori. 2008. Discrimination between ictal and seizure-free EEG signals using empirical mode decomposition. *Research Letters in Signal Processing* 2008 (2008).
- [60] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. 2019. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *Proceedings of the Internet Measurement Conference*. 478–485.
- [61] Andrea Pigorini, Adenauer G. Casali, Silvia Casarotto, Fabio Ferrarelli, Giuseppe Baselli, Maurizio Mariotti, Marcello Massimini, and Mario Rosanova. 2011. Time–frequency spectral analysis of TMS-evoked EEG oscillations by means of Hilbert–Huang transform. *Journal of Neuroscience Methods* 198, 2 (2011), 236–245. <https://doi.org/10.1016/j.jneumeth.2011.04.013>
- [62] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. 2009. An analysis of conficker's logic and rendezvous points. *Computer Science Laboratory, SRI International, Tech. Rep* 36 (2009).
- [63] Aria Ghora Prabono. 2022. Unofficial Implementation of RobustPeriod: Time-Frequency Mining for Robust Multiple Periodicities Detection. <https://github.com/ariaghora/robust-period>
- [64] Tom Puech, Matthieu Boussard, Anthony D'Amato, and Gaëtan Millerand. 2019. A fully automated periodicity detection in time series. In *International Workshop on Advanced Analysis and Learning on Temporal Data*. Springer, 43–54.
- [65] Check Point Research. 2020. Exploring QBot's latest attack methods. <https://research.checkpoint.com/2020/exploring-qbots-latest-attack-methods>.
- [66] A. Restrepo and L.P. Chacon. 1998. On the period of sums of discrete periodic signals. *IEEE Signal Processing Letters* 5, 7 (1998), 164–166. <https://doi.org/10.1109/97.700917>

- [67] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems* 29 (2016), 1163–1171.
- [68] Suseela T Sarasamma, Qiuming A Zhu, and Julie Huff. 2005. Hierarchical Kohonen net for anomaly detection in network security. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35, 2 (2005), 302–312.
- [69] Naeem Seliya and Taghi M Khoshgoftaar. 2010. Active learning with neural networks for intrusion detection. In *2010 IEEE International Conference on Information Reuse & Integration*. IEEE, 49–54.
- [70] Burr Settles. 2009. Active learning literature survey. (2009).
- [71] Burr Settles, Mark Craven, and Soumya Ray. 2007. Multiple-instance active learning. *Advances in neural information processing systems* 20 (2007), 1289–1296.
- [72] Andrii Shalaginov, Katrin Franke, and Xiongwei Huang. 2016. Malware beaconing detection by mining large-scale dns logs for targeted attack identification. In *18th International Conference on Computational Intelligence in Security Information Systems*. WASET.
- [73] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685* (2020).
- [74] DO SON. 2018. New malware uses specially crafted UDP protocol for C&C Communications. <https://securityonline.info/new-malware-uses-specially-crafted-udp-protocol-for-cc-communications/>.
- [75] STINGAR. 2023. Shared Threat Intelligence for Network Gatekeeping and Automated Response. <https://stingar.security.duke.edu/>.
- [76] Ming-Yang Su. 2011. Using clustering to improve the KNN-based classifiers for online anomaly network traffic identification. *Journal of Network and Computer Applications* 34, 2 (2011), 722–730.
- [77] Iwan Syarif, Adam Prugel-Bennett, and Gary Wills. 2012. Unsupervised clustering approach for network anomaly detection. In *International conference on networked digital technologies*. Springer, 135–145.
- [78] Florian Tegeler, Xiaoming Fu, Giovanni Vigna, and Christopher Kruegel. 2012. Botfinder: Finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. 349–360.
- [79] ThreatCrowd. 2023. ThreatCrowd. <https://www.threatcrowd.org/>.
- [80] Jorge L Guerra Torres, Carlos A Catania, and Eduardo Veas. 2019. Active learning approach to label network traffic datasets. *Journal of information security and applications* 49 (2019), 102388.
- [81] VirusTotal. 2023. VirusTotal. <https://www.virustotal.com/>.
- [82] Michail Vlachos, Philip Yu, and Vittorio Castelli. 2005. On periodicity detection and structural periodic similarity. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 449–460.
- [83] Chen Wei, Kihyuk Sohn, Clayton Mellina, Alan Yuille, and Fan Yang. 2021. Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10857–10866.
- [84] Qingsong Wen, Kai He, Liang Sun, Yingying Zhang, Min Ke, and Huan Xu. 2021. RobustPeriod: Robust Time-Frequency Mining for Multiple Periodicity Detection. In *Proceedings of the 2021 International Conference on Management of Data*. 2328–2337.
- [85] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848* (2019).
- [86] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10687–10698.
- [87] Kai Yang, Jie Ren, Yanqiao Zhu, and Weiyi Zhang. 2018. Active learning for wireless IoT intrusion detection. *IEEE Wireless Communications* 25, 6 (2018), 19–25.
- [88] Yi-Ren Yeh, Tang Chen Tu, Ming-Kung Sun, Shih Ming Pi, and C-Y Huang. 2018. A malware beacon of botnet by local periodic communication behavior. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2. IEEE, 653–657.
- [89] Zeek. 2023. An Open Source Network Security Monitoring Tool. <https://zeek.org/>.
- [90] Zeek. 2023. Zeek Dynamic Protocol Detection. <https://docs.zeek.org/en/master/logs/dpd.html>.
- [91] Hossein Rouhani Zeidanloo and Azizah Abdul Manaf. 2009. Botnet command and control mechanisms. In *2009 Second International Conference on Computer and Electrical Engineering*, Vol. 1. IEEE, 564–568.
- [92] Kim Zetter. 2014. Sony Got Hacked Hard: What We Know and Don't Know So Far. <https://www.wired.com/2014/12/sony-hack-what-we-know>.
- [93] Shuofei Zhu, Jianjun Shi, Limin Yang, Boqin Qin, Ziyi Zhang, Linhai Song, and Gang Wang. 2020. Measuring and Modeling the Label Dynamics of Online Anti-Malware Engines.. In *USENIX Security Symposium*. 2361–2378.

A SELF-TRAINING EVALUATION

We conducted an evaluation on 10-day GC1 dataset to demonstrate the effectiveness of self-training by comparing it with an active-learning and a supervised-machine learning pipeline (baseline). By the end of the experiment, we measured the dataset label distribution, as shown in Figure 12. The self-training pipeline successfully detects 11.5% and 46.2% more malicious ($\#MalENG \geq 2$) domains comparing to the active-learning pipeline and baseline, respectively.

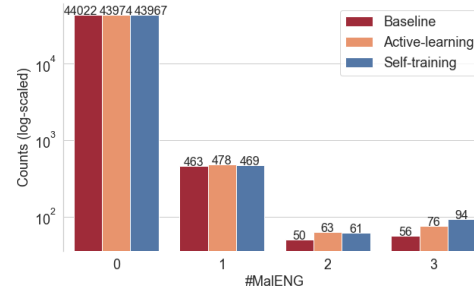


Figure 12: Class label distribution after 10 days training.

B LEARNING AND RANKING RESULTS

Figure 13 shows the active-learning and ranking results we obtained at Virginia Tech (Campus2) network. Similarly, we observe that the *global pipeline* outperforms the *local pipeline* throughout the whole evaluation process. As listed in Table 9, on a daily average, the *global pipeline* detects 19.47 more malicious domains than the *local pipeline*, nearly 54.3% of the total malicious activities detected by the standalone framework.

Table 9: Average daily detection.

	Local Pipeline	Global Pipeline	Diff.
Detected	39.17	58.69	19.52
Malicious (Original)	23.94	38.01	14.07
Malicious (Verified)	11.93	17.33	5.4
Malicious (Total)	35.87	55.34	19.47
Unknown	3.30	3.36	0.06

C CASE STUDY SUPPLEMENTS

Malvertising domains. Table 10 lists all 46 domains we found in the malvertising campaign as described in Section 7. All 46 domains are DGA-formatted with a 1-day DNS lifetime.

njRAT cluster. Figure 14 shows the network topology of the detected njRAT related cluster, consisting of 65 distinct malicious domains and 52 distinct IPs in 4 subnets (/24) hosted in Netherlands and the United Kingdom. We use red nodes to represent IP addresses, light blue nodes for malicious domains with an initial “malicious” label, dark blue nodes for pipeline-verified domains, and purple nodes for manual-verified domains. As discussed previously, among the 65 domains, 30 (46%) are with an original “malicious” label, and 33 (51%) are identified by our pipeline and are later verified through VirusTotal and manual analysis.

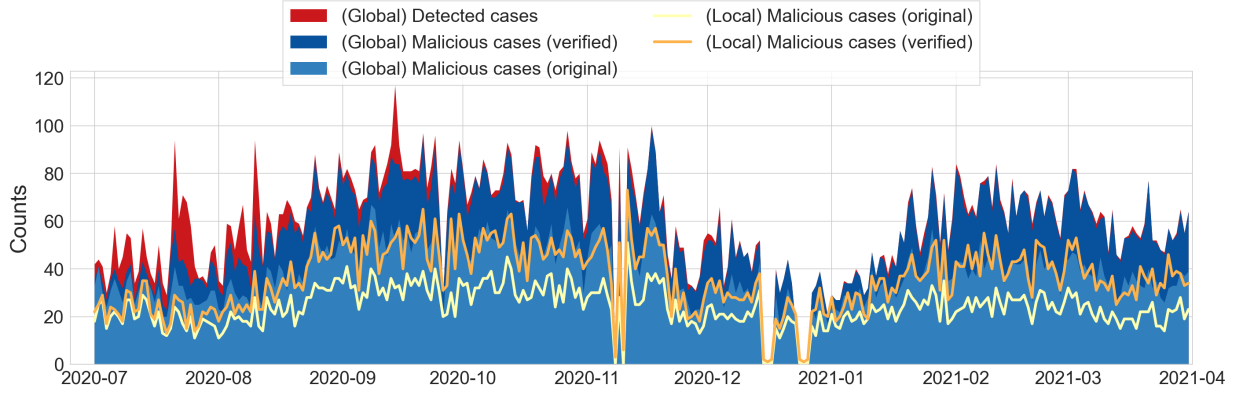


Figure 13: Comparison of active-learning and ranking based detection. X-axis shows the date, and y-axis shows the count of FQDNs. Four drops are due to network/machine failures during the data collection. The red area illustrates the daily detected cases by the *global pipeline*. The light blue area depicts the count of cases that can be verified on or close to the day of detection, and the dark blue area shows the amount of cases that can be verified as of March 2022. Yellow and orange lines are the baseline results in the *local pipeline*.

Table 10: Case study - malvertising domain names.

Domain Name	Domain Name
jcwuzktevijp{dot}com	tgvbwbjbnxz{dot}com
oegdmfjoqyyt{dot}com	pcdodues{dot}com
baitbrdwk{dot}com	comurzkcvttopr{dot}com
uszaodwya{dot}com	jcqwzssqks{dot}com
glqicmazu{dot}com	qbcpiertdje{dot}com
lgpumcja{dot}com	xcsypkkn{dot}com
luqafzbdscd{dot}com	qggypucwewn{dot}com
kfuiprtjuqntp{dot}com	sflywdvzyh{dot}com
sqhneencmysrk{dot}com	vnbugnwmwq{dot}com
eilczoszdrz{dot}com	zvxncxzgihb{dot}com
kjvboqnqoms{dot}com	pvcndecvyb{dot}com
boahwszwuebm{dot}com	dkypsidlj{dot}com
ebqfuphibj{dot}com	jnxkfldio{dot}com
hqlvmcrpb{dot}com	durpygckli{dot}com
hxslwibowjf{dot}com	nkjomdiztdy{dot}com
hicltgxzl{dot}com	dkdmygipll{dot}com
dlfclcey{dot}com	ixhparehw{dot}com
ciqguphollyj{dot}com	etzhporcxprf{dot}com
wwahelpfnhyx{dot}com	yhhbtavgpuo{dot}com
mqsrvkpcyzv{dot}com	ueuodgnrhb{dot}com
shqbsdjatunl{dot}com	krouekal{dot}com
rvedfxeljxo{dot}com	dtjvwspznqgtwf{dot}com
cbpsdvozwsbf{dot}com	umiyyuslde{dot}com

D FEATURES

We extracted 70 features, as listed in Table 11, to describe periodic domains from historical, temporal (periodicity) and topological (network graphs) characteristics. In general, features with an asterisk were calculated based on the graph database, and features labeled in blue were obtained based on historical information.

We describe our strategies on feature generation as follows, taking feature *maxCisco* and *avg_ipDomMalEngRatio* as two examples:

- **maxCisco.** We first updated our graph database with daily Cisco Umbrella score [35]. For each FQDN, we obtain the

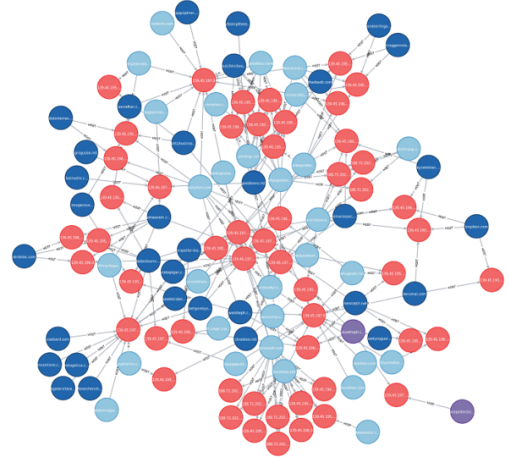


Figure 14: Case study - njRAT-related cluster topology.

maxCisco value by finding the maximum Cisco score of all FQDNs that were hosted on the same domain.

- **avg_ipDomMalEngRatio.** For each verified malicious FQDN in history, we recorded the number of anti-malware engines that detected it as malicious, denoted as *MalEng*. Then for each domain, we computed *DomMalEng* as the number of *MalEng* connected to it via {Domain, FQDN} tuples. We define *DomMalEngRatio* as *DomMalEng* divided by the total number of domains hosted on the specific IP. Let *I* be the set of IPs that a FQDN was hosted on in one day, the feature *avg_ipDomMalEngRatio* was computed as the mean of {*DomMalEngRatio*} for any *IP* ∈ *I*.

Table 11: Features extracted for active-learning pipeline.

Feature	Description	Feature	Description
psd_ratio	periodicity spectral power	freq	freq. being visited in last 30 days
dom_illegal	if domain in wrong format	occ	occurrence in last 30 days
dom_sld_entropy	SLD entropy	max_ipMalDomRatio*	ratio of max and total domain connected via IP
sub_dom_entropy	subdomain entropy	max_ipDomMalEngRatio*	ratio of max and total num. engines detecting as malicious connected via IP
dom_entropy	domain entropy	min_ipMalDomRatio*	ratio of min and total domain connected via IP
fqdn_entropy	fqdn entropy	min_ipDomMalEngRatio*	ratio of min and total num. engines detecting as malicious connected via IP
dom_level	domain level count	avg_ipMalDomRatio*	ratio of avg and total domain connected via IP
dom_length	domain length	avg_ipDomMalEngRatio*	ratio of avg and total num. engines detecting as malicious connected via IP
dom_tldcnt	domain tld length	sum_ipDomMalEng*	sum of engines detecting as malicious on domain hosted on IP
dom_sldcnt	domain sld length	max_ipDomMalEng*	max engines detecting as malicious on domain hosted on IP
dom_sub_count	subdomain count	min_ipDomMalEng*	min engines detecting as on domains hosted on IP
cntMalFQDNs*	malicious neighbor fqdn count	avg_ipDomMalEng*	avg engines detecting as malicious on domain hosted on IP
cntIP	IP count	minlen2malFQDN*	min topology length to nearest malicious fqdn
cntFQDN*	neighbor FQDN count	maxlen2malFQDN*	max topology length to nearest malicious fqdn
sumMalEng*	total num. engines detecting as malicious from neighbors	avglen2malFQDN*	avg topology length to nearest malicious fqdn
maxMalEng*	max num. engines detecting as malicious from neighbors	hist_malscore_min_period*	min historical malicious score for fqdn with same periodicity
minMalEng*	min num. engines detecting as malicious from neighbors	hist_malscore_max_period*	max historical malicious score for fqdn with same periodicity
avgMalEng*	avg num. engines detecting as malicious from neighbors	hist_malscore_mean_period*	mean historical malicious score for fqdn with same periodicity
maxCisco*	max cisco score from neighbors	hist_malscore_median_period*	median historical malicious score for fqdn with same periodicity
minCisco*	min cisco score from neighbors	hist_malscore_ratio_period*	ratio of domain with historical malicious score and the total with the same periodicity
avgCisco*	avg cisco score from neighbors	max_fqdn_period	max num. of fqdn with the same periodicity
sumMal*	sum history seen malicious fqdn from neighbors	min_fqdn_period	min num. of fqdn with the same periodicity
maxMal*	max history seen malicious fqdn from neighbors	mean_fqdn_period	mean num. fqdn with the same periodicity
minMal*	min history seen malicious fqdn from neighbors	std_fqdn_period	std. num. of fqdn with the same periodicity
avgMal*	avg history seen malicious fqdn from neighbors	min_period	min detected periodicity
malENG_ratio*	ratio of engines detecting as malicious from neighbors	max_period	max detected periodicity
malFQDN_ratio*	ratio of malicious and total fqdn from neighbors	mean_period	mean detected periodicity
sum_ipDom*	sum of domain connected via IP	std_period	std. detected periodicity
sum_ipMalDom*	sum of malicious domain connected via IP	cisco_min_period	min Cisco score on domain with same periodicity
avg_ipDom*	avg. domain connected via IP	cisco_max_period	max Cisco score on domain with same periodicity
avg_ipMalDom*	avg malicious domain connected via IP	cisco_mean_period	mean Cisco score on domain with same periodicity
max_ipDom*	max domain connected via IP	cisco_median_period	median Cisco score on domain with same periodicity
max_ipMalDom*	max malicious domain connected via IP	cisco_ratio_period	ratio of count of domains with a cisco score and total domains with same periodicity
min_ipDom*	min domain connected via IP	cisco_score	Cisco score
min_ipMalDom*	min malicious domain connected via IP	fqdn_popularity	local popularity score

¹Features marked with * are measured in the graph database.

²Features labeled in blue are measured with the historical information.

³We refer to Cisco Umbrella 1 Million List [35] for the universal popularity score.

⁴We define the campus popularity as the ratio of the number of IPs visiting the domain to the total count of observed IPs at campus on each day.

⁵We measure the frequency and occurrence of each periodic FQDN seen within last 30 days.