Visualizing Automatically Detected Periodic Network Activity

Robert Gove*
Two Six Labs

Lauren Deason†
PUNCH Cyber Analytics Group

ABSTRACT

Malware frequently leaves periodic signals in network logs, but these signals are easily drowned out by non-malicious periodic network activity, such as software updates and other polling activity. This paper describes a novel algorithm based on Discrete Fourier Transforms capable of detecting multiple distinct period lengths in a given time series. We pair the output of this algorithm with aggregation summary tables that give users information scent about which detections are worth investigating based on the metadata of the log events rather than the periodic signal. A visualization of selected detections enables users to see all detected period lengths per entity, and compare detections between entities to check for coordinated activity. We evaluate our approach on real-world netflow and DNS data from a large organization, demonstrating how to successfully find malicious periodic activity in a large pool of noise and non-malicious periodic activity.

Index Terms: Human-centered computing—Visualization—Visualization application domains—Information visualization; Human-centered computing—Visualization—Visualization design and evaluation methods; Security and privacy—Intrusion/anomaly detection and malware mitigation—Malware and its mitigation

1 INTRODUCTION

Malware infections often produce periodic network activity [1], such as exfiltrating data or communicating with a command and control (C2) server. However, many networks include a substantial amount of non-malicious periodic network activity [6] such as software update services. This can make it challenging to pinpoint the malicious periodic activity amongst non-malicious periodic activity. To address this problem, past research focused on giving users access to deep packet capture data to provide details on demand for each detected periodic signal [2]. However, on networks from large organizations, there will be too many periodic signals to investigate each signal individually. Further, packet capture data might not always be retained due to resource constraints. Users will need tools (possibly operating only on packet metadata) to provide information scent about which signals should be investigated further.

This paper presents 1) a novel algorithm for detecting periodic network activity along a wide spectrum of time horizons, and 2) a design study on the supporting visualization tool to explore the periodic activity detected by the algorithm. The visualization tool contains two novel components: A summary of periodic detections grouped by meaningful cybersecurity relevant attributes (e.g. domains, ports, and IPs), and a visualization design for seeing the multiple period lengths in each entity and comparing activity across entities to identify coordinated activity. We discuss lessons learned from our process of designing and refining the visualization system.

In Section 7 we present an example usage scenario run on 870 GB of Bro network data from a large organization. This data spans about 6 weeks of time, contains about 690 million records and represents 1,839,011 distinct entities in the Bro DNS data. In the netflow example referenced in the discussion section, the Bro Conn log data contains 9,951,969 distinct entities. The usage scenario demonstrates how to use our system to discover malicious activity within a large network. Example findings include a compromised endpoint periodically communicating with a malicious command and control (C2) server.

*e-mail: robert.gove@twosixlabs.com
†e-mail: lauren@punchcyber.com

2 RELATED WORK

Automatically Detecting Periodic Network Activity. The detection algorithm we propose draws from spectral analysis applications in various fields including analysis of online search queries, stock market indexes, and electrocardiogram data [17, 18] and easily generalizes to any discrete time series.

Nearly all studies conducting spectral analysis of network logs for the purpose of intrusion detection use datasets where known "attack" time series are artificially overlaid on top of (either real or synthetic) benign network traffic [2, 5, 9, 12], allowing for algorithmic design to be tailored to the characteristics of the known malicious signal. By contrast, we apply our algorithm to network logs obtained from a large partner organization, where presence of attacks, as well as instances of benign periodic activity, are unknown and unlabeled, and which exhibit larger volumes of automated traffic than can be parsed through manually to separate malicious from benign.

Heard et al. [12] compute the Discrete Fourier Transform (DFT) of time series of binary indicators for netflow connections between two IPs with the primary goal of filtering out benign polling activity. AsSadhan et al. [5] use a similar DFT-based approach to flag significant frequencies in the series of packet counts observed over time. Both algorithms only consider the highest magnitude Fourier coefficient, allowing only one frequency to be automatically detected. In contrast, our approach allows for a given signal to exhibit multiple period lengths simultaneously, and our visualization presents this information compactly. The approaches in these two papers are also constrained by the choice of time bin aggregation as to the period lengths that can be detected.

Chen et al. [9] propose an adaptive method to iteratively test for progressively longer periods. This iterative procedure stops when a significant period is detected, and therefore, unlike the method we propose, cannot identify multiple periods present in the same signal. Their intended detection target is slow paced periodic activity that is persistent over time, whereas our algorithm is designed to be able to detect this in addition to short bursts of periodic activity.

Bartlett et al. [6] develop a multi-level resolution methodology to successfully detect periods of widely varying time scales and changes in periodic behavior. This goal is similar to ours, however they accomplish this using discrete wavelet decomposition as opposed to the Fourier transforms used here.

Our work is most similar to Huynh et al. [2]. They use Fourier analysis to identify periodic activity in netflow, and interactive visualization to explore alerts and temporal patterns. In contrast, our work proposes a more robust algorithm to detect periodic network events along a wide spectrum of temporal scales. Whereas Huynh et al. use the share of signal energy contained in the top ten percent of Fourier coefficients, we consider the magnitude of each coefficient individually, allowing us to detect multiple periods present in a given signal, which can be relevant for certain malicious beaconing activity, as will be presented in Section 7. To eliminate the false positives that could arise from considering each frequency coefficient individ-

ually, we also automatically verify candidate periodicities identified in the periodogram with a computed autocorrelation function.

While Huynh et al. focus on periodicity detection in network flows, we show how to generalize our algorithm to other types of network events, such as DNS queries and HTTP requests, and provide evidence that our automated workflow can help point the analyst to the most interesting and unusual instances of periodic activity for a variety of use cases. To reduce the number of alerts of periodic activity presented to analysts, Huynh et al. cluster the alerts based on the alert's periodic signal: its fundamental period, its periodicity measure, and its energy. Although we incorporate analogous information in our tool, we use additional contextual cyber-relevant data as the primary source for grouping periodic activity (as described in detail later in this paper), allowing the analyst to automatically be guided to the most interesting, and potentially malicious, periodic activity present in a noisy dataset with billions of netflow observations.

Temporal Event Sequence Visualization. Research on temporal visualization is broad. Some work focuses on event sequences within a single entity (aka record) [14, 15, 19] or aggregate patterns across entities [21], but they do not focus on recurring temporal patterns or the time differences between events. Calendar viewer [16] and spiral layouts [20] show cyclic patterns, but these are not designed to show multiple period lengths across multiple entities, and they require manual specification the period length. VizTree [13] shows frequencies of patterns from discrete event sequences or discretized time series, but it does not detect periodic event sequences.

Several cybersecurity tools focus on monitoring network traffic for anomaly detection and active network defense [3, 4, 8], but they are not designed to detect or analyze periodic events.

In contrast to the above, our work automatically detects and visualizes periodically occurring events that may be persistent or bursty and that occur along a wide range of period lengths ranging from seconds to days (possibly within the same entity), and can operate on large datasets representing millions of entities. The aggregations of periodic activity presented in the visualization guide the analyst towards the most interesting and potentially malicious cases of automated activity.

3 Problem Background

Detecting periodic behavior in cyber-relevant logs for the purpose of identifying malicious activity requires addressing two distinct problems: 1) designing a scalable, general algorithm able to detect a variety of periodic patterns in discrete time series data, and 2) distinguishing potentially malicious periodic activity from benign.

The use of Fast Fourier Transforms (FFTs) to flag periodic behavior within discrete time series signals provides a scalable method to automatically detect regular temporal patterns more complex than those identifiable using simpler count and entropy measures of interarrival times. However, DFT-based algorithms can be subject to both false negatives and false positives, due to several well studied issues arising in digital signals processing. These include spectral leakage (when energy contained in one frequency bin spills over to adjacent bins, largely due to time series length not being a perfect multiple of period length), time bin resolution (periods corresponding to frequencies that lie between discrete frequency bins), and aliasing (when periodic activity occurs at frequencies higher than half the sampling rate, the "Nyquist frequency").

Independent of the technical issues arising in detecting periodic signals, the application of such detection techniques to cyber security problems requires the ability to distinguish malicious periodic activity from non-malicious. Users need contextual information beyond temporal features of network activity to understand whether it is potentially malicious.

To address both problems, we present a robust, scalable algorithm that uses DFTs to identify periodic activity, a set of automated post processing aggregations that incorporate relevant contextual information, and a visualization tool for surfacing potentially malicious activity and inspecting it in detail. The intended use case for this proposed workflow is to provide an automated tool to identify typical network activity, possibly for the purposes of whitelisting it, and to identify unusual or unexpected patterns of activity, including malware connecting to C2 servers, exfiltrating data, etc.

4 Detecting Periodic Network Events

The periodicity detection algorithm we propose is designed to minimize false negatives and false positives, to be able to detect complex patterns of regular activity at a variety of time scales potentially ranging from seconds to years, and to operate at scale on cyber-relevant logs that may contain billions of records across millions of entities. This algorithm is designed to be applicable to general discrete time series, with the ability to distinguish important periodic patterns in both impulse or rectangular signals as well as smoother functions closer to sinusoids. As such, it can be used to automatically detect spiky periodic signals that may arise from beaconing or periodic data transfers as well as to characterize the periods present in aggregate flows over time (such as bytes over time across a network) that may be cyclical. The specific examples presented in Section 7 apply the algorithm to entities that are defined based on specific use case scenarios, however the entire pipeline, including post-processing to locate unusual periodic activity within the visualization tool aggregations, can be applied to a wider variety of use cases and entities.

4.1 Two Step Periodicity Detection

To avoid false positives resulting from spikes in Fourier coefficient magnitude that can arise from issues including spectral leakage, aliasing, and noise, periodicity detection is performed on discrete time series in two steps: 1) Fourier coefficients are computed and magnitudes compared to a threshold to identify "candidate" periodicities, and 2) the autocorrelation function of the time series is computed and used to verify or reject these "candidate" periodicities. This two-stage method is largely adapted from the "AUTOPERIOD" strategy developed by Vlachos et. al. [18]. In particular, for a discrete time series signal $x[n]$ observed over N discrete time bins of length b seconds, we consider the coefficients of the DFT

$$X[k] = \frac{1}{N}\sum_n x[n]e^{-i2\pi\frac{k}{N}n}$$

and designate frequency k as indicative of a "candidate" periodicity at Nb/k seconds whenever $\frac{1}{N}|X[l]|^2$ exceeds threshold

$$H(p) \equiv \frac{-\ln p}{N}\sum_{n=0}^{N-1}|x[n]|^2 = \frac{-\ln p}{N^2}\sum_{k=0}^{N-1}|X[k]|^2 \qquad (1)$$

where this threshold is parameterized by p, the probability of labeling a frequency as "significant" in the case where the true signal was drawn entirely from Gaussian noise and therefore contains no meaningful periodicity. That is, parameter p represents the user's tolerance level for false positives in identifying "significant" periodicities in this stage. [1] From (1), we see that a frequency will be considered "significant" if its Fourier coefficient magnitude is large relative to the average coefficient magnitude. This relative frequency strength is represented by the ratio

$$FreqStrength(l) = \frac{|X[l]|^2}{\frac{1}{N}\sum_{k=0}^{N-1}|X[k]|^2} \qquad (2)$$

so that combining (1) and (2), we will consider a frequency l to be significant whenever this ratio exceeds negative $\ln p$, such that lower

[1] Note that p governs the tolerance for false positive detections arising in the periodogram due to noise; that is, detections reported for signals which do not exhibit periodic behavior. The parameter has no bearing on what an analyst may deem to be a "false positive" from a security perspective.

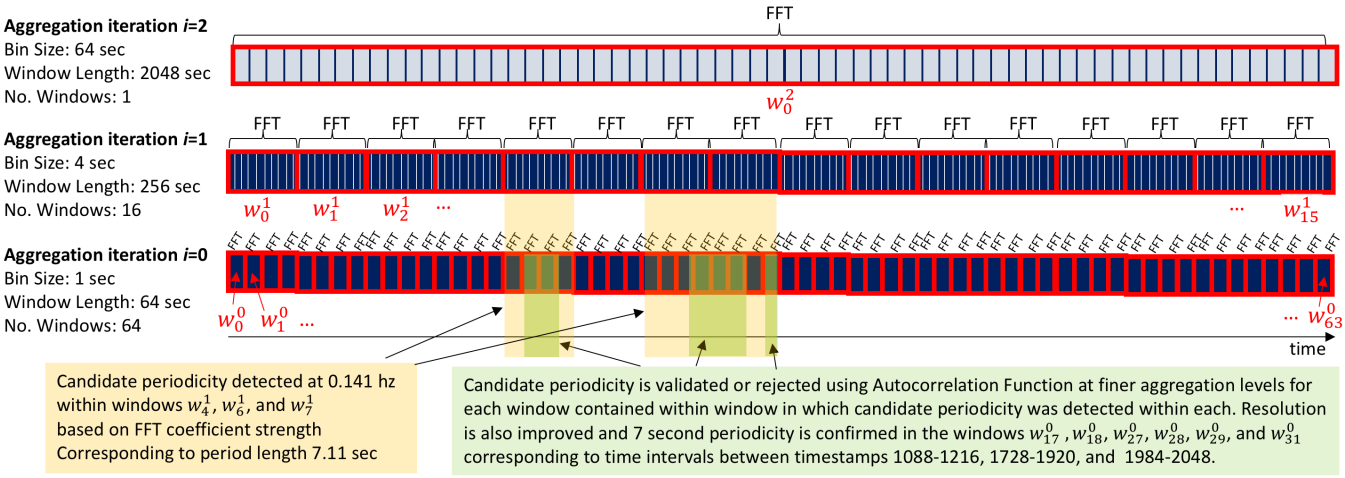

Figure 1: Example of multiple time bin aggregation levels and windows for a time period covering 4096 seconds, starting at second 0. In this example, the number of discrete time bins in each discrete series for which an FFT is computed is N=64. One such FFT is computed (in parallel) for each red window of length 64 bins, where these windows cover longer periods of time each for progressively coarser bin aggregation. Candidate frequencies detected in the first stage are verified and refined using the autocorrelation function both within the window where it was detected, as well as all windows at finer aggregation levels contained therein. In the example shown here, a detection at aggregation level $i = 1$ is discovered within windows $j = 4, 6, 7$ at frequency index 36 equal to $\frac{36}{64*4} = 0.141$ hz corresponding to a period length of $1/0.141 \approx 7.11$ sec. The autocorrelation function value near 7.11 seconds is not high enough to verify the candidate periodicity within the longer time window of detection, but is found to be sufficiently high to flag this period length within smaller windows of time during intervals 1088-1216, 1728-1920, and 1984-2048.

values of p require a higher frequency strength to be present in order for our algorithm to consider it a "candidate" periodicity.

In order to eliminate additional false positives that may arise due to spectral leakage or noise, and to improve detected period length resolution, "candidate" frequencies are verified by examining the autocorrelation function properties in the neighborhood of the corresponding period length. Specifically, we compute the autocorrelation function value for each lag τ,

$$ACF(\tau) = \sum_{n=0}^{N-1} x[n] * x[n-\tau] = \mathscr{F}^{-1}\{|X[k]|^2\}_\tau$$

using another FFT, taking advantage of the fact that this autocorrelation function is equal to the inverse Fourier transform, $\mathscr{F}^{-1}$, of the periodogram for real valued signals. A "candidate" will pass the second stage and be returned as a detected period if both of the following conditions are met: 1) The normalized autocorrelation value, $ACF(\tau)/\sum_{n=0}^{N-1} x[n]^2$, exceeds some user specified cutoff between 0 and 1, and 2) A quadratic least squares approximation of the ACF curve on the interval of period lengths within a half frequency bin of the "candidate" frequency has negative second derivative (in the case where this interval extends across at least two time bins b). That is, we require that the autocorrelation function be consistent with the presence of a relative maximum within this interval.

To detect regular activity ranging from periods of a few seconds to potentially hours or days, the two-stage algorithm described above is implemented in parallel at multiple levels of time bin aggregation. Binning time at multiple levels allows us to identify a wide range of period lengths, and increases algorithm robustness to different levels of jitter. Automatic binning at multiple aggregation levels is guided by explicit expressions governing the range of detectable period lengths under various assumptions on signal characteristics as derived in [11] and is depicted in Figure 1. The algorithm takes advantage of low frequency "candidate" periodicities detected within large time windows at higher aggregation levels to report period lengths at higher resolution obtained using second stage autocorrelation for smaller time windows contained therein. The pseudocode is shown in Algorithm 1, and it is outlined in detail in [10].

4.2 Entity Definition

In the case of applying our periodicity detection algorithm to DNS logs for the purpose of identifying potentially malicious beaconing behavior, we define an entity to be the 2-tuple (Source IP, DNS query string), and the time series of interest to be a simple binary indicator of whether the given query string was observed from the given source IP in each time bin. The aggregation method used here is $agg = max$ so that when aggregating at progressively larger time bins, we retain an indicator for whether the given IP submitted the query anywhere in the time bin. For this entity definition, our periodicity detection algorithm will flag instances where a given Source IP periodically queries the same DNS string at regular intervals. Alternative entity definitions that we have experimented with for this use case include aggregating Source IP up to the level of CIDR block or Autonomous System Number (ASN) company identifier and aggregating query strings to the level of domain or host. Doing so allows for detection of regular beaconing that originates from a pool of IPs within a CIDR block or company or which beacons to changing domains. An alternative entity definition that we have experimented with combines output of a Domain Generation Algorithm (DGA) detection algorithm to replace query string with a binary indicator for DGA domains in order to flag beaconing to dynamic domains, which are commonly associated with malware attempting to avoid detection by cycling through randomly generated domains.

In the case of applying our periodicity detection algorithm to netflow logs for the purpose of identifying unexpected automated behavior, we define an entity to be the 4-tuple (Source IP, Destination IP, Protocol, Destination Port), and the time series of interest to be the total number of bytes transferred during the flow initiated in the given time bin, with these values being aggregated over time bins by taking the sum (that is, $agg = sum$ for this use case). Note that while flows are often analyzed at the level of the 5-tuple including Source Port (as in [2], for example), we have found that aggregating over all source ports is useful in detecting meaningful automated activity, as source ports for automated activity often differ with each connection. Alternative entity definitions that we have experimented with for netflow analysis include replacing Source and Destination IPs with Company ASNs (to account for automated activity to/from a given range of IPs where the individual IP varies over time), replacing Destination port with min(Source Port, Destination Port) to better

Algorithm 1 Multiple resolution periodicity detection algorithm.

Input: Discrete Time Series $x(n)$ spanning a time period of L seconds; minimum bin size b; number of discrete time bins in FFT, N, such that $N = 2^{2p}$ for some $p \in \mathbb{Z}^+$; ACF cutoff A; and aggregation function agg

Output: Array of periodicity detection tuples: ⟨detected period length; window when periodicity discovered; ACF value; frequency strength⟩

procedure PERIODICITYDETECTOR($x(n)$, b, N, A, agg)
  $f \leftarrow$ associative 2D array ▷ f contains the candidate frequencies indexed by i (time bin aggregation index) and j (window index)
  $D \leftarrow$ empty array ▷ D is the array of detections, each a tuple of period length, window, autocorrelation value, & frequency strength
  for $i = 0$ to $\left\lceil \frac{\log_2(\frac{L}{b})-2p}{p-1} \right\rceil$ **do** ▷ Iterate over i while length of window w_j^i is less than the length of the full time series x
    $binSize_i \leftarrow b * 2^{i(p-1)}$
    $y \leftarrow [agg_{n=m*binSize_i}^{n=(m+1)*binSize_i-1}\{x(n)\}]_{m=0}^{m=\lceil \frac{L}{N*binSize_i} \rceil - 1}$ ▷ Bin time series x into bins of $binSize_i$ seconds using aggregation function agg
    for $j = 0$ to $\lceil \frac{L}{N*binSize_i} \rceil - 1$ **do**
      $w_j^i \leftarrow [y(jN), y(jN+1), \ldots, y((j+1)N-1)]$ ▷ Break the sequence of binned values into windows w_j^i of length N bins each
      $X_j^i \leftarrow [\frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-i2\pi\frac{k}{N}n}]_{k=0}^{N-1}$ ▷ Compute array of Fourier coefficients for w_j^i using a FFT
      for $k = 0$ to $N-1$, if $|X_j^i(k)|^2 > H_j^i(p)$ **do** ▷ Compare magnitude at each discrete frequency to threshold
        $f(i,j) \leftarrow \frac{k}{N*binSize_i}$ ▷ Store candidate frequency
      $ACF_j^i(n) \leftarrow iFFT(|X_j^i(k)|^2)$ ▷ Compute autocorrelation function
  for all candidate frequencies $f(i,j) \in f$ **do**
    for all $i' \geq i$ **do** ▷ Consider finer time aggregation windows $w_{j'}^{i'}$ contained within window w_j^i
      for $j' = 2^{(i'-i)(p-1)} j$ to $2^{(i'-i)(p-1)}(j+1) - 1$ **do**
        $I \leftarrow \left[\left\lfloor \frac{N}{f*N*binSize_i+0.5} \right\rfloor * binSize_i, \min\left\{ \left\lceil \frac{N}{f*N*binSize_i-0.5} \right\rceil, N-1 \right\} * binSize_i \right]$ ▷ Interval I within a half bin of $1/f(i,j)$
        $\widehat{ACF_{j'}^{i'}} \leftarrow$ 2nd order least squares approximation of $ACF_{j'}^{i'}$ on interval I
        if $\widehat{ACF_{j'}^{i'}}'' < 0$ and $\max_{\tau \in I}[ACF_{j'}^{i'}(\tau)] > A$ **then** ▷ Accept periods where ACF may exhibit relative max
          D.push($\langle \tau, [j' * binSize_{i'}, (j'+1) * binSize_{i'}], \max_{\tau \in I}[ACF_{j'}^{i'}(\tau)], \frac{|X_j^i(k)|^2}{\frac{1}{N}\sum_{l=0}^{N-1}|X_j^i(l)|^2} \rangle$)
  return D

approximate service port, and aggregating together records where both source and destination ports lie in the ephemeral range in order to flag automated traces left by services set up by users on arbitrary high ports.

5 Design Requirements

The visualization tool's design has many requirements. These are motivated by our conversations with three cyber analysts who were involved with our development process in order to investigate seemingly malicious detections and provide informal feedback on the tool's development.

Finding relevant activity in large logs. The data from medium and large organizations are typically too much for analysts to examine all of the periodic signals—both the raw log data and the output from a periodicity detector. Users need some way to find network events that are worth investigating further.

Simplicity. Many cybersecurity analysts do not have a background in data science. From our conversation with cyber analysts, it is not uncommon for lower tier analysts to follow an analysis script to decide what should be elevated to higher tier analysts for in-depth analysis. The user interface should attempt to simplify the underlying mechanics of the periodicity detector, thereby allowing analysts to focus on whether or not the periodic activity is concerning enough to investigate further.

Support existing methods. A common technique in cybersecurity threat hunting is to perform a top-*n*/bottom-*n* analysis. This involves sorting a table of network log data to find the most and least common things in the logs, such as IP addresses, ports, domains, countries, etc. The cybersecurity analysts we talked to often perform this step as a means to provide situational awareness in an unfamiliar network. This provides a quick understanding of typical network activity (top n) and common types of one-off activity (bottom n). Sometimes malicious activity appears in the bottom n analysis (e.g. a single infected host pinging a command and control server on an unusual port), but it is obscured because there is so much one-off network activity. Furthermore, usually low-tier analysts are the ones examining new alerts, and they often follow a script, or checklist, to make an initial determination of whether or not to escalate the alert for further investigation. Therefore, showing an overview of all alerts may not be helpful, but providing appropriate context about the periodic activity, such as ports and IPs, is necessary for determining whether or not to investigate an entity.

Support existing tool chains. From our conversations with network operation center analysts, they typically have established and preferred tools for investigating network activity; the main challenge is to help them find activity that should be investigated. Therefore, new tools should support analysts in conducting work that is difficult or impossible to perform using their existing tools, but users should be unconstrained to use their existing tools when needed.

6 System Design

Network Logs (e.g. Bro[2]) are ingested and stored in the Hadoop File System. The periodicity detection analytic and post processing aggregations are written in Scala using Spark and operate on the ingested data. Results are stored in Hive and then ingested into Elasticsearch. The visualization system is designed as a client-server web application built using Angular.js[3] and D3.js [7]. The application queries Elasticsearch to retrieve and aggregate data.

The user interface is divided into two main components: Aggregation summary tables to find periodic activity to investigate, and a periodic activity visualization to analyze periodic network events.

[2] https://www.bro.org/

[3] https://angular.io/

Table 1: The aggregation summary tables for two categories of log type. For some aggregation dimensions, relevant metadata is also included in the set of group by fields, if available (such as city, country, and company in the case of Source or Destination IP aggregations). These dimensions are included to provide context for the analyst but would not affect which records are included in the grouping as they do not vary within the other group by fields.

Log type	Aggregation summary table	Group by
DNS & HTTP	Domain	Domain and detected period length
	Query	Full query string, hostname, top-level domain, and detected period length
	Source IP	Source IP, source company, source city, source country, and detected period length
Netflow	Destination Company	Destination company, destination city, destination country, and detected period length
	Destination IP	Destination IP, destination company, destination city, destination country, and detected period length
	Destination Port	Protocol, destination port, and detected period length
	Source Company	Source company, source city, source country, and detected period length
	Source IP	Source IP, source company, source city, source country, and detected period length

A note on terminology: because autocorrelation function (ACF) is unlikely to be a familiar term to cyber analysts, we opted to use the term "regularity" in the user interface instead. "Regular" is probably easier to understand since it is in the Basic English word list[4].

6.1 Aggregation Summary Tables

The periodicity algorithm outputs many detections (on the order of less than 1% of all entities in input data, for the example described in Section 7). For large volumes of input data, 1% will generally still yield too many detections to parse through manually, some of which may be malicious activity, while others are typical, non-malicious activity.[5] To support analysts in finding detections of interest, the visualization system creates several tables that aggregate the data into meaningful groups. The visualization creates different tables depending on the type of log data being used (DNS queries, HTTP requests, or netflow). These are called the *aggregation summary tables*. Each table groups the detections by a different combination of attributes that are relevant to the type of log data. The grouping in these aggregation summary tables is described in Table 1. Figure 2 shows an example of an aggregation summary table.

Users can sort and filter these tables, thereby supporting the top-*n*/bottom-*n* analysis often performed by cyber analysts. Users can sort the aggregation summary tables using multiple columns, for example first sorting by count of IPs and then sorting by regularity score. This gives users a sense of the typical activity on the network. When users see one or more rows that they wish to investigate, they can select the rows by clicking the checkboxes and then clicking the "View" button. This takes them to the periodic activity visualization.

[4]http://ogden.basic-english.org/words.html

[5]However, note that in an operational setting, whitelists would exist that could dramatically reduce input size, e.g. by whitelisting internal traffic and traffic to known services such as Windows Update.

6.1.1 Alternative Designs and Lessons Learned

An initial version of this page did not have the aggregation summary tables. Instead, it had range sliders to filter detections by time, window length, period length, frequency strength, and regularity. After users adjusted the filters, they could click "View" to see all matching detections in the periodic activity visualization. However, we did not find these initial filters to be useful from a cybersecurity perspective. This is because malicious activity usually was not tied to the attributes of the periodic signal. Instead, it is more often the event features themselves, such as ports and IPs, that are interesting to analysts. It is possible that a time filter could be useful in an incident response scenario—for example, if users already know that malicious activity occurred in a given time range and they wish to look for additional activity—but this did not fit our typical usage. Instead, we found that the aggregation summary tables provided a more useful information scent about suspicious activity while also enabling users to rapidly filter down to a narrow set of related data.

6.2 Periodic Activity Visualization

This view shows the entities for the rows selected in the aggregation summary tables on the previous page. The periodic activity view has two visualizations per entity. The top one is a visualization of detections of periodic activity, and the bottom one is a time series visualization of the entities. These are shown in Figures 3 and 4.

The periodic activity is grouped by entity. Each colored bar is a detection of periodic activity. Darker colors indicate the periodic activity is more regular (i.e. it has a higher ACF score). The horizontal position indicates the window of time where the periodic activity was detected. The vertical position of the colored bar indicates the period length, and its height indicates the possible range of the period length (where this is determined by the bin size in the aggregation level at which detection occurred). Users can hover over a colored bar to see a tooltip with more details, such as frequency strength and precise time range. One advantage of this visualization is that it clearly communicates the different period lengths detected in the entity's time series. This visualization also allows users to see if entities exhibit coordinated periodic activity by presenting time series data for multiple entities on a common time scale.

The timeseries visualization shows when the entity's log events were recorded, enabling users to confirm that the periodic activity they see is supported by the data. When users are zoomed out and more than one event has the same x-axis pixel value, users can hover over the data point to see how many events are contained within it.

The view is paginated, showing 10 entities per page, sorting entities either by the maximum regularity per entity, or the maximum frequency strength per entity. Filters on the right allow users to filter out detections, and entities with no detections matching the filters are hidden from view.

6.2.1 Alternative Designs and Lessons Learned

The initial design used a row-based view for the y-axis. This design had one row for each distinct period length detected on an entity. Detections were shown using the same colored bars and x-axis as the current design. We chose this initial design because we were concerned that having two continuous axes related to time would be confusing to users. However, we found the row-based view confusing when there were two distinct, but similar, period lengths detected. In this case, after rounding the period lengths for display, the period lengths appeared the same and it was not obvious why the data was not merged into a single row. To avoid this confusing scenario, we used the continuous y-axis described above.

Because most period lengths are short, but a few are very long, using a linear scale on the y-axis created problems: when adjusted to make detections with small period lengths visible, this made the y-axis too long and detections with long period lengths took up too much space. Using a log scale compressed the height of detections with long period lengths too much, and they were too thin on the screen to be visible. Eventually we settled on using a square root

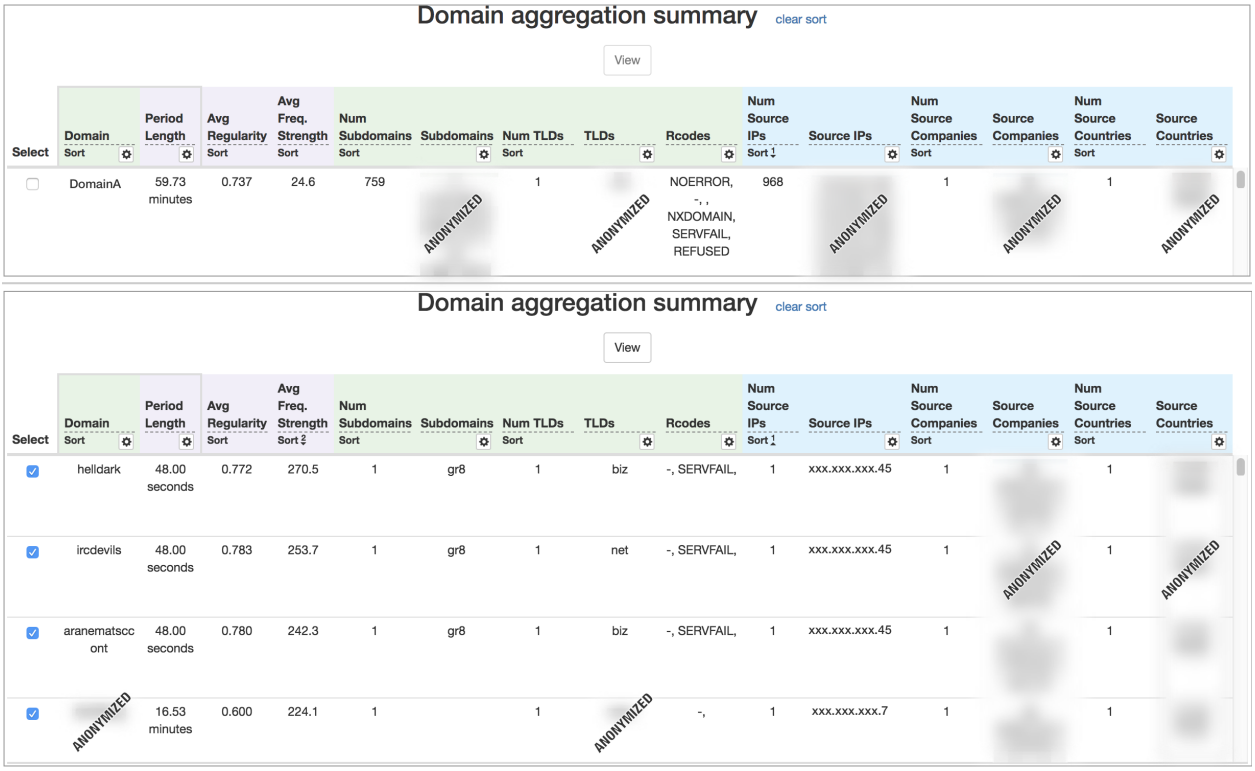

Figure 2: The Domain aggregation summary table for the DNS data discussed in Section 7. The top is a "top-n" analysis, showing the most common domain ("DomainA") periodically queried by source IPs. This (anonymized) domain is non-malicious and can be whitelisted. The bottom is a "bottom-n" analysis, showing uncommon domains with periodic queries. These domains have been associated with Poison Ivy C2 servers.

scale as a compromise that compressed the scale of the y-axis but did not compress it as much as the log scale. This comes with an additional tradeoff of using an unconventional scale that may be misunderstood by users, but we feel this is an acceptable design tradeoff because comparing the height of bars is not a crucial task when analyzing this data.

7 Example Usage Scenario

To demonstrate the ability of our algorithm to detect periodic activity and our visualization to help identify potentially malicious activity, we describe an example usage scenario. This scenario uses Bro DNS log data from a large government organization (referred to as LGO for the rest of this section), comprising 45 days, 59,528,647 records, and over 1.8 million distinct entities. Using detection cutoffs of 0.6 for autocorrelation score and 1E-5 for p, the algorithm described in Section 4 ran on this batch dataset in 66 minutes and yielded detections of periodic activity for 13,711 entities. The screenshots shown here use anonymized IP addresses, domains, and company names.

We begin by examining the Domain Aggregation Summary table. We sort the table in descending order by number of source IP addresses (see Figure 2). We see that DomainA is the most common domain for periodic DNS queries in this dataset. This is a primary domain owned by LGO, and therefore unlikely to be related to malicious activity, and a good candidate for whitelisting.

After sorting the Domain Aggregation Summary table by ascending source IP count and descending frequency strength, the top three rows are all from the same source IP and they have the same period length. The domains appear to contain suspicious names (e.g. helldark), and having the same period length indicates that these automated queries may be related. We select the first 10 rows, including these three rows, and click the "View" button to visualize them. The number 10 is chosen somewhat arbitrarily; we know we are interested in the first three, but we are also interested in looking at additional rows within the "bottom-n" to explore other unusual periodic query activity. The number of rows will depend on the user's time and established procedures for choosing alerts to analyze.

For these entities, we see three distinct types of coordinated activity, one of which is shown in Figure 3 and which exhibits multiple detected period lengths. After consulting open source intelligence, we found that the domains these IPs are periodically querying have previously been associated with known malware families (Poison Ivy and Palevo/Rimecud). We note the three source IP addresses generating that coordinated periodic activity (xxx.xxx.xxx.45, xxx.xxx.xxx.193, xxx.xxx.xxx.194). We now want to know if there are other infections related to these source IPs, so we go to the Source IP aggregation summary table and filter to show only detections for these three source IP addresses. Now we see all of the domains that these IPs are periodically querying. They are either the malicious domains we have already seen, or are known non-malicious domains.

8 Discussion

In the example usage scenario, we found that being able to quickly pivot between types of aggregations (i.e. Domain and Source IP) and sort and filter the tables enabled us to develop new questions about the data and answer them quickly. We also found that showing an overview of all detected periodic activity or a detailed exploration interface to be unnecessary for finding malicious activity; rather, the simple tables performed well for our use case. Being able to see the periodicity visualizations of multiple entities enabled us to identify coordinated activity among IPs and domains.

We found our periodicity visualization to be very effective for identifying periodicity trends within entities. Figure 4 shows entities from our netflow data that exhibit periodic behavior with period length gradually increasing over time—a feature that is potentially difficult to see by looking at a simple output table.

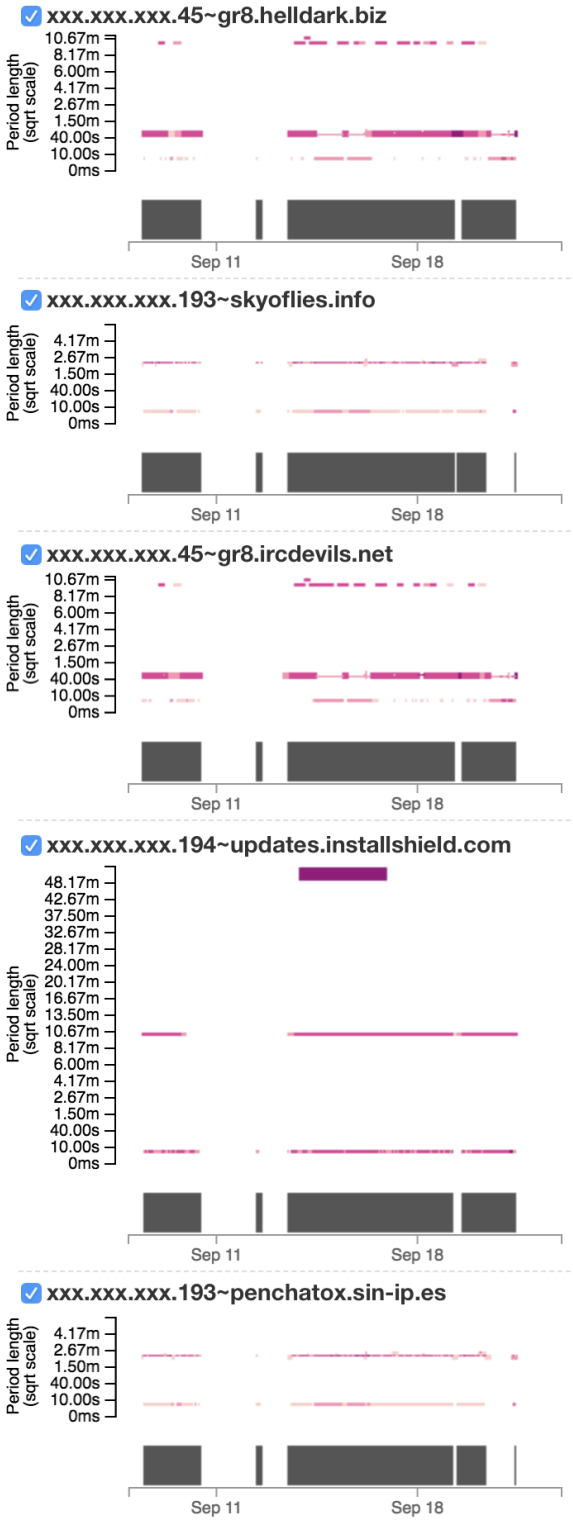

Figure 3: Detected periodic activity in the example usage scenario. Darker colored bars indicate more regular (higher ACF score) detections. The dark gray bars for each entity are the time series of the entity. The majority of the entities appear to follow similar, if not the same, coordinated periodic activity.

This periodicity analytic and accompanying visualization tool were developed in a lab that was set up to perform a mixture of cyber threat hunting and incident response work. Therefore, the tool was designed with those use cases in mind, but it could be modified to fit other use cases. As it is, we believe this would be a useful tool to make available for incident response and cyber threat hunting teams. These teams occasionally need to look for undiscovered malicious activity on a network, and finding suspicious periodic activity can fit well with these use cases.

Our three cyber analysts often commented that this tool appears very useful. One feature request was to see the total traffic for the entire network in conjunction with the periodicity visualizations. This would provide additional context about why there are gaps in the time series or periodicity detections (e.g. knowing whether a gap in periodicity detections was related to missing log data).

Additionally, the paginated display and limited screen space inhibits the ability to see any overview of all selected entities. This is partly due to technical limitations; for large selections, the necessary data to create an overview would require hundreds of thousands of matches from Elasticsearch, which would be time consuming to retrieve and then create the overview visualization. Future work should investigate not only an appropriate overview visualization, but also the appropriate technology to enable dynamically creating an overview of such large data selections.

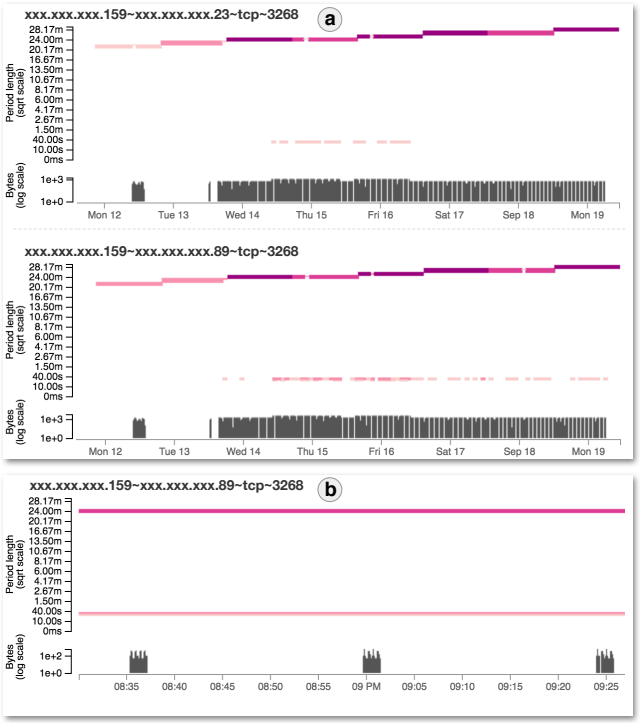

Figure 4: (a) We see two entities (one source IP and two destination IPs) exhibiting coordinated periodic activity. The period length is increasing over time, an insight that could be difficult to see in time series data or a table. The algorithm detects a weaker periodicity of about 30 seconds during certain intervals throughout the period, in addition to the longer period lengths. (b) By zooming in on one of these entities we get a better sense of the relationship between the two periods detected. On September 15, we see bursts of communication between xxx.xxx.xxx.159 and xxx.xxx.xxx.89 on port 3268 every 24 minutes. Within each burst, there are regular spikes occurring every 30 seconds, but with a smaller regularity (autocorrelation) score.

8.1 Future Refinements

Based on our experience using this tool and informal feedback from our three cyber analysts during the development process, we propose several refinements to improve the tool's utility to make it better suited for use by incident response, cyber threat hunting, and continuous monitoring teams.

To better support workflows in continuous monitoring environments, such as security operation centers (SOCs), we suggest modifying the tool to support whitelists and blacklists. Users could import whitelists and blacklists from other sources, and also select entities, IP address, ports, or other attributes within the tool to add them to the whitelist or blacklist. This would drastically reduce the number of detections for known non-malicious activity, thereby making the number of detections more tractable for a Tier 1 analyst to work through and decide whether to ignore a detection or escalate it to a Tier 2 or Tier 3 analyst.

Additionally, while the algorithm described in Section 4 assumes data is ingested in a single batch covering the available time period of logged data, there is a logical extension for streaming data. In this case, the algorithm is repeated at the lower time aggregation levels on minibatches as data streams in, while simultaneously, data aggregated at coarser time bin levels is retained for periodic processing on longer minibatches. Efficient implementation of such a periodic minibatch application is left as a future refinement.

To provide better context for analysts, we suggest adding additional statistics in the visualization for each entity. These statistics would include data such as the number of other IPs on this port, or the number of other IPs with periodic activity to the given destination IP. This information is mostly contained in the aggregation summary tables, but incorporating it in the periodicity visualization would help analysts more quickly pivot to answer questions related to a given entity. This, combined with the existing visualizations, could provide information and context to analysts faster than querying and sorting database tables, while also using analysts' visual system to help find trends, patterns, and outliers.

The above modifications would help analysts from all tiers, but to help complete the analysis cycle, we would also like to include the ability for users to inspect packets, similarly to Huynh et al. [2]. Alternatively, incorporating functionality to automatically generate a SQL query string that retrieves the original data related to a detection could also help analysts more quickly determine whether a given detection is malicious.

9 Conclusion

In this paper we presented a novel algorithm based on Discrete Fourier Transforms capable of detecting multiple distinct period lengths in a single timeseries and time varying periodic activity. We also presented a novel user interface for exploring the detections found by the algorithm. This user interface is composed of two main parts: Aggregation summary tables, that provide a method of grouping and sorting detections by cybersecurity-relevant features, and periodicity visualizations that visualize multiple period lengths of an entity's timeseries simultaneously while also enabling users to compare across entities to check for coordinated periodic activity. We discussed alternative designs and lessons learned from our iterative design and development process.

We presented an example usage scenario, demonstrating the effectiveness of the periodicity detection algorithm and the utility of combining the aggregation summary tables with the periodicity visualization. We have also shown how the periodicity visualization helps users identify patterns in periodic activity that would be more difficult to identify in a typical table view used by analysts.

Acknowledgments

The authors thank Mike Barlow, Kevin Fields, Mike Geide, John Lankau, and Kyle Smith. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

References

[1] N. Anh Huynh, W. Keong Ng, and H. Giang Do. On periodic behavior of malware: experiments, opportunities and challenges. In *MALCON*, pages 1–8, 2016.

[2] N. Anh Huynh, W. Keong Ng, A. Ulmer, and J. Kohlhammer. Uncovering periodic network signals of cyber attacks. In *2016 IEEE VizSec*, 2016.

[3] D. Arendt, D. Best, R. Burtner, and C. Lyn Paul. CyberPetri at CDX 2016: Real-time network situation awareness. In *2016 IEEE VizSec*, pages 0–3, 2016.

[4] D. L. Arendt, R. Burtner, D. M. Best, N. D. Bos, J. R. Gersh, C. D. Piatko, and C. L. Paul. Ocelot: User-centered design of a decision support visualization for network quarantine. In *2015 IEEE VizSec*, pages 0–7, 2015.

[5] B. AsSadhan, J. Moura, and D. E. Lapsley. Periodic behavior in botnet command and control channels traffic. In *GLOBECOM*, pages 1–6, 11 2009.

[6] G. Bartlett, J. Heidemann, and C. Papadopoulos. Using low-rate flow periodicities for anomaly detection: Extended. Technical report, 2009.

[7] M. Bostock, V. Ogievetsky, and J. Heer. Data-Driven Documents. *IEEE TVCG*, 17(12):2301–2309, 2011.

[8] B. C. Cappers and J. J. Van Wijk. SNAPS: Semantic network traffic analysis through projection and selection. In *2015 IEEE VizSec*, 2015.

[9] L. M. Chen, S.-W. Hsiao, M. C. Chen, and W. Liao. Slow-paced persistent network attacks analysis and detection using spectrum analysis. *IEEE Systems Journal*, 10(4):1326–1337, 2016.

[10] L. Deason. Detecting automated and periodic activity of varying time scales (at scale) in cyber relevant log data. Working paper, 2018.

[11] L. Deason. Introduction to periodicity detection in discrete time series using fourier transforms. Working paper, 2018.

[12] N. Heard, P. R. Delanchy, and D. Lawson. Filtering automated polling traffic in computer network flow data. In *Joint Intelligence and Security Informatics Conference*, pages 268–271, Sept 2014.

[13] J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom. Viztree: a tool for visually mining and monitoring massive time series databases. In *VLDB*, pages 1269–1272, 2004.

[14] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: visualizing personal histories. In *CHI*, pages 221–227, 1996.

[15] J. Saxe, D. Mentis, and C. Greamo. Visualization of shared system call sequence relationships in large malware corpora. In *VizSec*, pages 33–40, 2012.

[16] J. J. Van Wijk and E. R. Van Selow. Cluster and calendar based visualization of time series data. In *Information Visualization*, pages 4–9, 1999.

[17] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*, pages 131–142. ACM, 2004.

[18] M. Vlachos, P. Yu, and V. Castelli. On periodicity detection and structural periodic similarity. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 449–460. SIAM, 2005.

[19] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events. In *CHI*, pages 457–466, 2008.

[20] M. Weber, A. Marc, and W. Müller. Visualizing time-series on spirals. In *Information Visualization*, pages 7–12, 2001.

[21] K. Wongsuphasawat, G. John Alexis Guerra, P. Catherine, W. Taowei David, S. Ben, and M. Taieb-Maimon. LifeFlow: visualizing an overview of event sequences. In *CHI*, pages 1747–1756, 2011.