Discovering Significant Co-Occurrences to Characterize Network Behaviors

Kristine Arthur-Durett[1], Thomas E. Carroll[1(✉)], and Satish Chikkagoudar[2]

[1] Pacific Northwest National Laboratory, Richland, WA, USA
{Kristine.Arthur-Durett,Thomas.Carroll}@pnnl.gov
[2] U.S. Naval Research Laboratory, Washington, DC, USA
satish.chikkagoudar@nrl.navy.mil

Abstract. A key aspect of computer network defense and operations is the characterization of network behaviors. Several of these behaviors are a result of indirect interactions between various networked entities and are temporal in nature. Modeling them requires non-trivial and scalable approaches. We introduce a novel approach for characterizing network behaviors using significant co-occurrence discovery. A significant co-occurrence is a robust concurrence or coincidence of events or activities observed over a period of time. We formulate a network problem in the context of co-occurrence detection and propose an approach to detect co-occurrences in network flow information. The problem is a generalization of problems that are encountered in the areas of dependency discovery and related activity identification. Moreover, we define a set of metrics to determine robust characteristics of these co-occurrences. We demonstrate the approach, exercising it first on a simulated network trace, and second on a publicly-available anonymized network trace from CAIDA. We show that co-occurrences can identify interesting relationships and that the proposed algorithm can be an effective tool in network flow analysis.

Keywords: Cyber situation awareness
Significant co-occurrence detection · Temporal relationship discovery
Robust correlation

1 Introduction

Characterizing network behavior can help inform the operation and state of a networked computing environment. Characterization is a process of describing attributes and activities. There are a number of existing characterizations that illuminate a system's relationship to a network [13], role [17], and activities [4,18]. Characterizations improve situation awareness, helping network operators and cyber practitioners understand purpose and intent of the component, along with its relation to the network. In this paper we introduce a novel network behavior characterization method based on discovering significant co-occurrence

S. Yamamoto and H. Mori (Eds.): HIMI 2018, LNCS 10904, pp. 609–623, 2018.
https://doi.org/10.1007/978-3-319-92043-6_49

within network flow information. A *significant co-occurrence* is a robust concurrence or coincidence of events or activities observed over a period of time. As the definition suggests, the analysis occurs in the temporal plane, examining for correlation of events or activities.

The method discussed in this paper came from our research into dependency discovery. Advanced network defenses such as application reconfiguration and network address hopping often need to understand dependencies to maintain network and service function. Dependencies between components can be a surprisingly complex web of relationships. Direct dependencies (e.g., web service requires a database to store and retrieve data) were documented by architects and owners, but indirect relationships (e.g., the database would admit connections slowly if it could not reverse map the network address to a qualified hostname) were unnoticed or overlooked. Automated methods to discover these relationships are needed to overcome this gap in knowledge. Dependencies exhibit both topological and temporal characteristics and we found it difficult to meaningfully combine both aspects into existing graphical representations. We modified our approach to first extract temporal structure, before performing any type of topological analysis. We tried an approach based on Self-Organizing Maps (SOMs), an unsupervised, two-dimensional artificial neural network [14], to extract temporal relationships from sets of time-encoded event vectors. The result was that the SOM would overgeneralize relationships, learning aspects of the data not relatable to the problem. We believe this is a natural consequence of the low information content of network flow. Each record is a 13-tuple, and while a single record has informational merit, a set of records is highly redundant. This results in component analysis focusing on elements that have high entropy but little interest, such as linearly incrementing or randomly chosen port numbers or unmeaningful differences in network addresses. We finally fell to an approach that discovers significant co-occurrences between aggregates of data.

Later research identified that significant co-occurrences are useful for detecting related activity, such as coordinated interactions over remote shell sessions. This is useful because: (1) identified behaviors are coordinated both in time and topology, and (2) it permits identification of relationships in the time domain that cannot be inferred from graph representations.

This paper is organized as follows. In Sect. 2, we describe a characterization problem and formulate it as significant co-occurrence discovery. We introduce a robust algorithm to discover significant co-occurrences in Sect. 3 and we then evaluate our approach in Sect. 4. In Sect. 5, we discuss the outcomes of the evaluation, additional measures that we investigated, and work to enrich the models. We then conclude in Sect. 6.

Related Work. In earlier work, we examined flow information to discover network service and application dependencies. We introduced the idea of encoding network flow information into time series [6]. We then used cross-correlation to identify dependencies as a function of the lag between classes of time series. This showed an improvement in sensitivity and specificity when compared to earlier approaches. We then refined this approach by contextualizing flow behavior [7].

This is done by aggregating flow series based on a set of frequently occurring patterns of elements. We did not guarantee long-term stationarity (that is, mean and variance does not change with time) in the time series, which is an underlying assumption of the cross-correlation methodology. The approach proposed in this paper employs a cross-correlation analog that is not sensitive to stationarity.

Upon further investigation, we generalized our approach as a method to detect significant co-occurrence relationships. Work by Jalali and Jain [12] considers a similar problem formulation, but uses counts and conditional probabilities to extract relationships. NSDMiner [16] and Sherlock [1] use a similar count-based approach to extract dependencies. In an evaluation comparing NSDMiner and our approach [6], we demonstrated that cross-correlation was less sensitive to flow volumes than count-based approaches. CloudScout [21] uses cross correlation to detect dependencies, but did not consider that the cause and effect did not occur concurrently. Tor correlation attacks take advantage of correlating traffic patterns at different locations [15]. The Tor injection and correlation attack described in [8], which deanonymizes users, is performed over flow information and can be thought of as an exercise of a co-occurrence detection of the reference signal (created by the injection).

2 Problem Formulation

The completeness of network flow information is dependent on the positions of the flow probes. It is a simple fact that a probe will only report flows that it observes in its observation domain (not withstanding sampling policies, capture

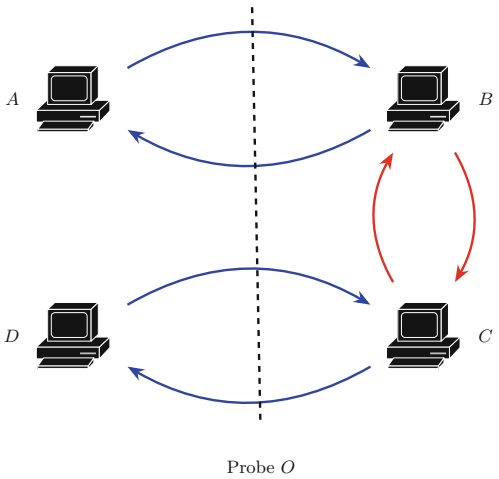

Fig. 1. The communications between systems A, B, C, and D are coordinated. Probe O can observe communications that the dashed line cuts, which are A-B and C-D interactions. Due to probe O's position in the network, it cannot observe $B \leftrightarrow D$ communications.

speed, resource limitations, etc.). Due to the incompleteness of data, constructed graph topologies may be missing significant relationships, complicating the process of computing graph walk statistics, centrality measures, graph distances, and other graph theoretic measures.

To illustrate the problem, consider Fig. 1 where four systems, A, B, C, and D are coordinating communications. Probe O can observe communications that are cut by the vertical dashed line. In this example, communications between $A \leftrightarrow B$ and $C \leftrightarrow D$ are observed, but the probe does not observe $B \leftrightarrow D$. While the relationship between B and D cannot be observed, there may exist temporal structure between $A \leftrightarrow B$ and $C \leftrightarrow D$ to infer a hidden relationship. Specifically, if $A \leftrightarrow B$ are recurrently co-occurring with $C \leftrightarrow D$, we may infer a relationship between $A \leftrightarrow B$ and $C \leftrightarrow D$. Since there are four communication channels between $A \leftrightarrow B$ and $C \leftrightarrow D$ (traffic destined to A, B, C, and D), there are four opportunities for co-occurrences (in practice this often limited as a probe sees on direction only, either due to configuration or asymmetric routing).

The problem as formulated is the correlation in time of observable activities in network flow information and/or packet traces. The conceptual *mechanism* is that if components exhibit some form of functional relationship, that we should observe that network activity at one component will cause an incidence of co-occurring network traffic at another component. In our parlance, we map network activity to a stream of timestamped *events*, and then discover significant co-occurrences between the events. A *co-occurrence* is a concurrence or coincidence of events, discovery of which entails observation for these types of recurrent relationships over a time period. *Significant co-occurrences* are then, given some metric, robust co-occurrences. Significant co-occurrences are of interest in multiple domains as causal relationships often emit co-occurrences and significant co-occurrences are frequently determined on the path to causal identification. In the context of our problem, co-occurrences do not strictly mean causations. Information systems and architectures have not conventionally been built for experimentation, that is, to allow an invention that may reduce network redundancy and may impair system function. Moreover, passive approaches are preferable and feasible to more data sources.

The problem as formulated has evolved with our understanding and work on several related problems that have common threads of time structure and practical causation/correlation (i.e., without intervention) discovery. Returning to our prior work on dependency discovery, our insight was that significant co-occurrences are established among aggregates of systems and components. For instance, a dependency exists between set of clients accessing a web application running on a cluster (a set of servers). The set of clients and set of servers are their own *event class*. The dependency exists between the classes and not necessarily among the individual elements contained therein. We encoded the problem as the counts of connection initializations and terminations, analyzing classes for significant co-occurrences of these events.

The problem introduced at the start has distinguishing attributes that require modified aggregation and encoding. Moreover, aspects of collection influence the

design. In terms of collection, connection initializations and terminations are not reliably observed or inferred. Moreover, to reduce resource requirements, high-speed collectors don't maintain full connection state. Instead, they export flow information based on resource thresholds and timers, and not necessarily at the proper end of flow. Furthermore, we don't expect to observe many connection initializations (or terminations) when looking for activity relations. To provide better quality encoding for the related activity, we aggregate flow information by unique source and destination network address and transport attributes. We then encode the number of bytes transferred during each period. The significant co-occurrences are identified, which correlates variations in the time series.

3 Approach

Our proposed approach is founded on co-occurrence detection, followed by identifying significant co-occurrence relationships. A *co-occurrence* is a repeated observation of (concurrent or delayed) "first this, then that." A *co-occurrence relationship* is revealed through time and is a primitive aspect of causal relationships.

Returning to Fig. 1, our approach is tooled to detect the co-occurrence of *A*-*B* communications with *C*-*D* communications. More specifically, we are looking for co-occurrences of changes in datarate, that is, the changes in *A*-*B* datarate co-occur with changes in *C*-*D* datarate. We conceive of a *mechanism* where the systems are organized such that *A*-*B* interactions result in effects along a series of systems, which eventually transits *C*-*D*. As we observe relative changes in *A*-*B* datarate, we expect to observe relative changes in *C*-*D* datarate. This mechanism fits where information is being passed from one node to the next, such as what occurs in Tor and other low-latency virtual/overlay networks and the chaining of remote terminals.

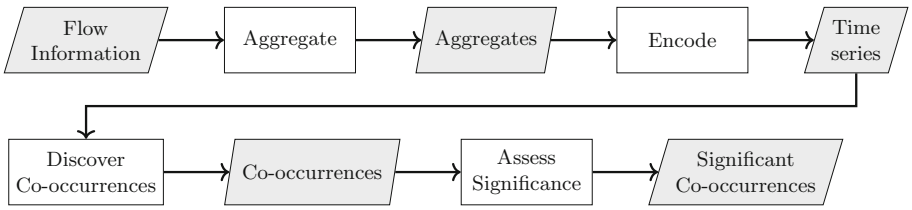

Fig. 2. Diagram of the analytic workflow. A rectangle represents a process, a parallelogram shows an input or output.

We break our approach to detect significant co-occurrence relations in network flow information into four stages:

1. Aggregate flow information,
2. Encode aggregates as time series,
3. Detect co-occurrences within the time series, and
4. Identify *significant co-occurrence relationships* over the time dimension.

The relationships of the stages and the corresponding inputs and outputs are diagrammed in Fig. 2. The combined effect of State 1 and 2 is to represent the flow information as time series. This input is then consumed by Stage 3 to analyze for co-occurrences. Lastly, the significance of the co-occurrences are assessed. We now describe the stages in full.

3.1 Aggregation

The analysis workflow begins by dividing the set of flows into flow aggregates. The flows in an *aggregate* share common features, such as all are sourced from or destined to the same endpoint. The exact details of the procedure are dependent on the causal mechanism and its effects. Once understood, rules can be defined to categorize flows into aggregates. In our previous work [6,7] we investigated dependencies in enterprise computing networks. A *dependency* exists between component a (say, a web server) and b (say, a database), if a's function is impaired when b becomes impaired (i.e., fails, faults, or is degraded). For this case, flows that have destination address, protocol, and destination port in common are merged into a single aggregate. For the question introduced in this paper, the dynamics are not expressed *en masse* but are expressed *singularly* between pairs of endpoints. Consequently, we create aggregates based on common source and destination addresses, protocol, and destination port.

3.2 Encode

Next, we encode aggregates as time series. A feature of interest from the flows in the aggregate is accumulated along an equally-spaced time dimension. We choose equally-spaced time series over other time series representations as it promotes efficient computation of correlation to be performed in the next step. In one encoding, the number of flow starts or completions are accumulated as a count in the respective time bin. In another possible encoding, we use the byte (flow volume) or packet count to model data rates. In this encoding, the count is accumulated in the bins representing the time interval the flow was active. Before accumulating, the count is divided by the number of bins. The primary encoding discussed in this paper is based on byte counts.

Assuming we have b_t bytes in the t time period, we compute the log ratio as follows:

$$r_t = \log\left(\frac{b_t + 1}{b_{t-1} + 1}\right), \tag{1}$$

which gives the normalized datarate changes. Computing the log compresses the dynamic range of the ratio, preventing false positive results that result from high dynamic range.

3.3 Co-Occurrence Detection

We correlate the time series to detect co-occurrence relationships between flow aggregates. To determine the correlation between each pair of time series, we

used an alternative coefficient presented by Erdem et al. [9]. This coefficient effectively captures the cross-dependence of the two time series over time, but does not have the requirement of stationarity of the time series, in contrast to the more commonly applied Pearson's product moment correlation coefficient (see [2] for more information about Pearson's). We quickly appreciated the alternative coefficient as we could not guarantee stationarity over the length of the time series in prior work.

Given two time series, X and Y, the Erdem coefficient ρ_O is defined as:

$$\rho_O = \frac{\alpha_{xy}}{\sqrt{\alpha_{xx}}\sqrt{\alpha_{yy}}},$$

where $\alpha_{xy} = \mathbf{E}\left[(X_t - X_{t-1})(Y_t - Y_{t-1})\right]$, $\alpha_{xx} = \mathbf{E}\left[(X_t - X_{t-1})^2\right]$, and $\alpha_{yy} = \mathbf{E}\left[(Y_t - Y_{t-1})^2\right]$. Expanding the definition, we observe that

$$\alpha_{xy} = \frac{1}{T-1}\sum_{t=2}^{T}(X_t - X_{t-1})(Y_t - Y_{t-1}),$$

is the first-order autocorrelation of X_t and Y_t [3]; the definitions of α_{xx} and α_{yy} directly follow. The definition as specified is for the case when the cause and effect simultaneously occur at the same time t. We can generalize the definition for a delay model, for which the effect *lags* ℓ time periods behind the cause. For $\ell = 0, \pm1, \pm2, \ldots$, the lagged variant of the coefficient is $\rho_O[\ell] = \alpha_{xy}[\ell] / \sqrt{\alpha_{xx}}\sqrt{\alpha_{xy}}$, where:

$$\alpha_{xy}[\ell] = \begin{cases} \frac{1}{T-1}\sum_{t=2}^{T-\ell}(X_t - X_{t-1})(Y_{t+\ell} - Y_{t+\ell-1}) & \ell = 0, 1, 2, \ldots \\ \frac{1}{T-1}\sum_{t=2}^{T+\ell}(Y_t - Y_{t-1})(X_{t-\ell} - X_{t-\ell-1}) & \ell = 0, -1, -2, \ldots \end{cases} \tag{2}$$

Using the log ratio defined in (1), we can rewrite (2) to

$$\alpha_{xy}[\ell] = \begin{cases} \frac{1}{T-1}\sum_{t=2}^{T-\ell} r_{xt} r_{yt+\ell} & \ell = 0, 1, 2, \ldots \\ \frac{1}{T-1}\sum_{t=2}^{T+\ell} r_{yt} r_{xt-\ell} & \ell = 0, -1, -2, \ldots \end{cases} \tag{3}$$

Using the fast Fourier transform (FFT) to compute the discrete frequency transform (DFT) of the time series, convolution in the frequency domain can be used to compute the coefficients $\rho_O[\ell]$ for all lags $\ell = 0, \pm1, \pm2, \ldots, n$ providing significant computational speedup and improved precision over the naïve approach computed in the time domain. Prior to performing the FFT, the time series is doubled in length by padding with zeros. Convolution allows us the opportunity to smooth the results. We continue to have success using Lanczos' σ factors [10, "Lanczos' σ factors" and "The σ Factors in the General Case"] in increasing the magnitude of the coefficient of true correlations:

$$\sigma_i = \begin{cases} 1 & i = 0, \\ \frac{\sin(2\pi i/2(T-1))}{2\pi i/2(T-1)} & i = 1, 2, \ldots, 2(T-1) \end{cases} \tag{4}$$

3.4 Significant Co-Occurrence

A significant co-occurrence is a robust pattern found in the observations. In order to identify these significant co-occurrences, we developed several criteria. Fundamentally, we associate significance with values of ρ_O. As ρ_O approaches one, we assume a greater significance. Unfortunately, our time series encoding combined with the character of the problem produces a sparse representation, where many elements of the time series are zero. We developed additional criteria to measure the amount of information in an encoded time series, which supports the interpretation of ρ_O and significance conclusions. The intuition for the criteria is as follows. Assume that two co-occurrences are measured to have equal ρ_O, we consider the time series with more non-zero elements to be of greater significance. Additionally, we have found the zeroes between the first and last non-zero elements to also support the claim of significance. Given equal quantity of non-zero elements, the longer this length, more significant the co-occurrence. We developed two measures of this criteria: span and occupancy. For an encoded time series, we define *span* as the length of the run of the first to last non-zero elements, while *occupancy* is the quantity of non-zero elements. Formally, we define span as:

$$s = j - i,$$

where i and j are the indices of the first and last non-zero elements, respectively. Occupancy is defined as:

$$o = count\{x_i | x_i \neq 0\}. \tag{5}$$

To speed the computation and increase scalability, encoded time series must satisfy minimum span and occupancy conditions. The time series that satisfy these conditions are then submitted for correlation computation. The distribution of ρ_O is then examined and a threshold determined. Finally, we were interested in novel pairings. To find these, we sorted the IP/port pair labels and counted the number of occurrences. We then kept records that had a count of 2 or less. The last step was necessary as certain time series were characterized by low variance. The correlation coefficient can be considered a measure of covariance between two time series; this has the effect of driving the value of ρ_O higher if one or both time series has a low variance. Therefore, time series with a low variance will tend to falsely present as significant co-occurrences with multiple pairings.

4 Evaluation

We evaluated our proposed approach using two methods. For the first method, we build a simple mechanism in a testbed to coordinate remote shell sessions. The goal is to prove out the analysis for unambiguous, optimal series of coordinated interactions. For the second, we apply the approach to a publicly-available dataset to identity significant co-occurrences. Unlike the first evaluation, there isn't additional available information to support conclusions.

Fig. 3. A diagram of the first evaluation's setup. The setup comprised twelve virtual machines, coordinating a VNC session and ten SSH remote shell sessions in a series circuit.

Table 1. Significant co-occurrence coefficients for the first evaluation. The table has been compressed to save space. Observe that closer the sessions are in the sequence, higher the correlation coefficient.

	VNC	SSH1	SSH2	SSH3	···	SSH8	SSH9	SSH10
VNC	—	0.977	0.977	0.976	···	0.976	0.976	0.977
SSH1	0.977	—	1.000	1.000	···	1.000	0.999	0.999
SSH2	0.977	1.000	—	1.000	···	1.000	0.999	0.999
SSH3	0.976	1.000	1.000	—	···	1.000	0.999	0.999
···	···	···	···	···	···	···	···	···
SSH8	0.976	1.000	1.000	1.000	···	—	1.000	1.000
SSH9	0.976	0.999	0.999	0.999	···	1.000	—	1.000
SSH10	0.977	0.999	0.999	0.999	···	1.000	1.000	—

The purpose of the first synthetic test is to evaluate the analysis on unambiguous, optimal series of coordinated remote shell session interactions. A diagram of the setup is shown in Fig. 3. In a testbed environment, twelve Linux virtual machines were created. A harness was constructed that would begin with a VNC sessions and then login via SSH into a series of ten remote hosts (the first host executes the VNC viewer, connecting to the VNC server). On the downriver host, a command was executed that would mirror input to output. A script would arbitrarily generate input that would traverse the series of hosts and then return to be displayed in the VNC viewer. We captured the packets on the network, recording network traffic for a duration of one hour. The YAF tool suite [11] was used to transform the trace into unidirectional flow information. The active timeout (or hard timeout, where the flow entry is exported once expired) and the idle timeout (the flow entry is exported if no traffic is received before timeout) were each set to 10 s. Flow was aggregated and encoded into 30 s interval time series by payload length. Time series representing inbound and outbound traffic were then merged into a single time series. A summary of results is given in Table 1. All coefficients were near one as, again, this was evaluated under optimal conditions.

For the second evaluation, we applied our approach to the CAIDA Anonymized Internet Traces 2016 Dataset [5]. This publicly-available dataset is an hour-long packet trace that was captured on January 21st, 2016 at the Chicago Equinix Internet Exchange. The trace, comprising network and

Table 2. CAIDA Anonymized Internet Traces 2016 Datasets summary information.

First timestamp	January 21, 2016 12:59:11.415 UTC
Last timestamp	January 21, 2016 14:02:51.405 UTC
Capture length	20 B to 80 B
Unique IPv4 addresses	17,722,741
IPv4 flows	165,476,840
IPv4 byte/flow	34,629 B flow^{-1}± 296 B flow^{-1}
IPv4 datarate/1 min average	410.5 Gbit s^{-1}± 3.36 Gbit s^{-1}
Unique IPv6 addresses	476,500
IPv6 flows	3,323,221
IPv6 byte/flow	68,332.0 B flow^{-1}± 806 B flow^{-}1
IPv6 datarate/1 min average	21.0 Gbit s^{-1}± 0.2 Gbit s^{-1}

Table 3. Notable significant co-occurrences that were identified.

Pair 1	Pair 2	Coefficient
194.84.75.170 → 66.216.158.246:ssh	194.84.75.170 → 66.216.152.59:ssh	1.
209.169.193.11 → 138.38.133.180:telnet	209.169.208.163 → 138.38.133.180:telnet	1.

transport headers of incoming and outgoing packets, was collected with the assistance of a high-speed monitor. Dataset documentation does not mention lost or dropped packets. Summary information about the dataset is presented in Table 2. Our objective was to identify plausibly related remote shell sessions.

We began the analysis by transforming the packet trace to flow information. The packet trace arrives in a series of files, demarcated by direction and minute timestamp. We merged the files by direction in increasing order of timestamp, followed by merging the directions into a common trace. The `YAF` tool suite was used to transform the trace into unidirectional network flow information. The active timeout (or hard timeout, where the flow entry is exported once expired) and the idle timeout (the flow entry is exported if no traffic is received before timeout) were each set to 10 s. We next filtered the flow information by application, keeping records characteristic of remote shell sessions, and discarding the rest. We filtered on the combination of protocol and well-known or common ports, preserving flows sourced or destined to TCP ports 22 (ssh), 23 (telnet), 512–514 (BSD UNIX "r" commands), 3389 (remote desktop protocol), 5900–5963 (VNC), 5938 (TeamViewer), and 6000–6063 (X11); along with UDP 3389 (remote desktop protocol), 5938 (TeamViewer), and 60000–61000 (Mosh). Flow was then encoded and aggregated into 30 s intervals, producing time series of 128 elements in size.

Analysis was performed and 131,161 co-occurrences were identified with a correlation coefficient greater than or equal to 0.80. Upon examination, we noted that 122,001 results involved TeamViewer. Investigating the packet traces, we

observed that the flows exhibited 1 min-periodic traffic pattern. A TeamViewer protocol description [19,20] supported our suspicion that the purpose of this communication was to maintain connectivity (keep a firewall from considering the connection dead, among other reasons) and inform status. While these co-occurrences were significant, they were not notable in the context of our objective. We further winnowed the results by offset (preferring alignments of −1, 0, or 1), then by uniqueness of network and transport pairs. The treatment identified 43 co-occurrences that were deemed significant *and* notable. While all 43 co-occurrences were plausible, two co-occurrences, which are identified in Table 3, comprise two nodes connecting to a service on a third. Time of use and the structure further supports the conclusion that these sessions were possibly coordinating their interactions.

Once again analyzing the CAIDA dataset, we investigated for significant co-occurrences of HTTP communications. HTTP operates using TCP on well-known ports 80 and 443, and alternate ports 4433, 8000, 8008, 8080, 8443, and 8888. A significant co-occurrence opens up the possibility that the channel is used for something other than strictly document transfer, such as anonymizing network traffic. Using the aggregation and encoding approach as described in the treatment of the remote shell session, we identified 1172 significant co-occurrences among the 27,789,076 flow records. For this problem we defined significance as correlations with $\rho_O \geq 0.8$, an offset of zero, occupancy greater than or equal to ten, and equal occupancy and span of the aggregate time series. After reviewing the significant co-occurrences, we argue that they are plausible. Additional measures need to be defined to assess notability, that is, the interest of an analyst to the problem.

The evaluation demonstrates approach finding significant co-occurrences that could be interesting to cyber practitioners and analytics. The datasets themselves have limitations that prohibit further validation. As the packet trace does not provide payloads and the data itself was passively collected from an anonymous population, it is difficult to improve our confidence in the results. Significance is built with repeated observations: given more data observed over a longer duration we would improve our confidence in the results. We are further evaluating the approach using internal datasets, where it continues to show promise.

5 Discussion

We utilized several measures and metrics to identify significant correlations. As is evident from the evaluation and prior work, specific aggregations, encodings and measures are required to model aspects of the temporal behaviors exhibited by complex systems. The span and occupancy measures were found to be useful for filtering the time series during the encoding phase. Additional measures and metrics were used during the significant correlation detection stage. As discussed below, some of those metrics explored did not prove to be useful to the context of the problem in hand and further work is required to explore additional metrics. Additionally, the relationship calls being made do not have any directionality or

order associated to them. Further exploration to address the directionality and "*causal*" nature of relationships is underway. Additional attributes can be added to the generated models to make them more useful to analysts.

5.1 Measures and Metrics

During initial attempts at significant correlation detection, we investigated several metrics to identify "interesting" characteristics of co-occurring time series. We considered a time series to contain more significant information if it has more non-zero entries or a wide span of non-zero entries.

Attempts to identify significant correlations involved scaling the cross correlation peaks using span and occupancy. Peaks between time series with a high occupancy and/or span were weighted more heavily than sparsely populated time series. Two different weighting factors were developed from the ratio of the occupancy and the span of a time series to its total length. The peak cross correlation value is then scaled using the weighting factors of both contributing time series. A pair of time series with full span and occupancy have a weighting of one; thus, the weighting does not reduce the peak. Significant correlations were defined to be outliers based on the *Interquartile Range* (IQR) of all peak values.

However, these weighting metrics significantly compressed the dynamic range of all peak values. This led to difficulty in identifying outliers. Significant co-occurrence relationships were occasionally compressed to the point where they were unidentifiable. This led us to conclude that these measures were not useful in this context. However, the basic concepts of span and occupancy proved useful in initial filtering of time series. Encoded time series are not submitted to correlation calculations unless they meet a minimum threshold of both span and occupancy.

5.2 Model Enrichment

Additional attributes and statistics may be calculated to enrich the generated models and improve their significance and understandability to end users. For example, identification of recurrent patterns in either the encoded time series or the occurrence of significant relationships can provide experts with information on the typical behavior of a system. *Recurrent* patterns are behaviors that have a periodicity to their occurrence; these events may take place hourly, daily, or weekly.

Recurrent Events. Recurrent events are event classes that occur with some periodicity. For example, connections to a time card certification system may peak on the day that certifications are due. Identification of recurrent events informs experts on when they can expect a given event class to typically occur. In turn, this allows them to develop rules for other applications or detect unusual behaviors.

Identification of recurrent events follows naturally from the calculation of cross-correlation coefficients described in Sect. 3.3. Given flow records, we may encode time series based on occurrence count, byte or packet counts. The bin sizes are chosen based on the periodicity being explored. Once the time series is encoded, we may slightly modify the Erdem coefficient ρ_0 to calculate the autocorrelation.

Substituting a time series X into (3) for both the X and Y inputs gives:

$$\alpha_{xx}[\ell] = \frac{1}{T-1} \sum_{t=2}^{T-\ell} r_{xt} r_{xt+\ell} \qquad \ell \in (0, \infty) \tag{6}$$

Analogously to the original cross-correlation approach, this coefficient describe the covariation of a time series with itself at various lag and is contained within the range $[-1, 1]$. A coefficient closer to 1 indicates a strong positive correlation, or that the time series tends to change in a similar manner over time. By its definition, the autocorrelation coefficient will always equal 1 at the zero lag because a time series is always correlated with itself.

When autocorrelation is performed on a time series, a resulting peak indicates the period at which the event class tends to occur. For example, if data is binned by the minute and there is a peak in the autocorrelation at lag 10, this event tends to reoccur every 10 min. The lack of a peak in the autocorrelation implies that the event class does not display any recurrent behavior at the current level of granularity. However, care should be taken to verify that periodicity cannot be identified at a different granularity.

Recurrent Correlations. It may be of further interest to explore *recurrent correlation* present within the dataset. A recurrent correlation is a significant co-occurrence that will occur at regular time intervals. For example, an individual may initiate a remote session connection once a day. Identification of these patterns allow experts to determine typical joint behaviors of event classes.

Determination of a recurrent correlation follows similar steps as those for recurrent events. The difference exists in the encoded time series that is the input to the autocorrelation function. For this case, the time series are encoded based on the presence of significant relationships. The time series bins contain counts of significant co-occurrences during that time interval. A peak in the resulting autocorrelation indicates that there is periodicity in this relationship.

5.3 Further Application

The work presented shows how Erdem's coefficient can be expanded to find lagged correlations between time series. This approach furthers experts' understanding of relationships within a network. An understanding of significant correlations can aid in related domains, such as mission assurance. When contemplating the risks involved in completing an organization's mission, it is necessary to

understand the underlying dependencies in the networks that support the organization. Unexpected relationships may lead to additional considerations when evaluating the criticality of mission support systems.

6 Conclusion

This paper presents a general method to characterize network behaviors that manifest in the temporal domain. The characterized behaviors describe interactions between networked components. By analyzing the data in the temporal domain, indirect relationships (an indirect relationship is one in which no graph path exists between components) may be inferred. The proposed method employs significant co-occurrences detection which is a generalization of our prior work on dependency discovery. Formulations of different mechanisms governing interactions, influence the design of aggregation and encoding stages. We showed the method can be applied on flow information to related problems such as coupled remote shell sessions. Further analysis can be applied to determine the notability of the co-occurrence to network defenders and operators.

Acknowledgment. Portions of the research were funded by PNNL's Asymmetric Resilient Cybersecurity (ARC) Laboratory Research & Development Initiative. This work was performed while Satish Chikkagoudar was at Pacific Northwest National Laboratory. The views expressed in this paper are the opinions of the Authors and do not represent official positions of the Pacific Northwest National Laboratory, the Department of the Navy, or the Department of Energy.

References

1. Bahl, P., Chandra, R., Greenberg, A., Kandula, S., Maltz, D.A., Zhang, M.: Towards highly reliable enterprise network services via inference of multi-level dependences. In: Proceedings of the ACM SIGCOMM Conference on Data Communications (SIGCOMM), pp. 13–24 (2007)
2. Box, G.G.P., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control, 4th edn. Wiley, Hoboken (2008)
3. Brockwell, P.J., Davis, R.A.: Time Series: Theory and Methods, 2nd edn. Springer, Heidelberg (1991). https://doi.org/10.1007/978-1-4899-0004-3
4. Bruillard, P., Nowak, K., Purvine, E.: Anomaly detection using persistent homology. In: Proceedings of the Cybersecurity Symposium (CYBERSEC). IEEE (2016)
5. CAIDA: The CAIDA Anonymized Internet Traces 2016 Dataset (2016). http://data.caida.org/datasets/passive-2016/README-2016.txt. Accessed 29 Mar 2016
6. Carroll, T.E., Chikkagoudar, S., Arthur-Durett, K.: Impact of network activity levels on the performance of passive network service dependency discovery. In: Proceedings of the Military Communications Conference (MILCOM), pp. 1341–1347. IEEE (2015)
7. Carroll, T.E., Chikkagoudar, S., Arthur-Durett, K.M., Thomas, D.G.: Automating network node behavior characterization by mining communication patterns. In: 2017 IEEE International Symposium on Technologies for Homeland Security (HST), pp. 1–7. IEEE (2017)

8. Chakravarty, S., Barbera, M.V., Portokalidis, G., Polychronakis, M., Keromytis, A.D.: On the effectiveness of traffic analysis against anonymity networks using flow records. In: Faloutsos, M., Kuzmanovic, A. (eds.) PAM 2014. LNCS, vol. 8362, pp. 247–257. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04918-2_24
9. Erdem, O., Ceyhan, E., Varli, Y.: A new correlation coefficient for bivariate time-series data. Phys. A: Stat. Mech. Appl. **414**, 274–284 (2014)
10. Hamming, R.W.: Numerical Methods for Scientist and Engineers, 2nd edn. Dover, New York (1986)
11. Inacio, C.M., Trammell, B.: YAF: yet another flowmeter. In: Proceedings of the 24th Large Installation System Administration Conference (LISA 2010). USENIX (2010)
12. Jalali, L., Jain, R.: A framework for event co-occurrence detection in event streams. CoRR abs/1603.09012 (2016)
13. Joslyn, C., Cowley, W., Hogan, E., Olsen, B.: Discrete mathematical approaches to graph-based traffic analysis. In: Proceedings of the International Workshop on Engineering Cyber Security and Resilience (ECSaR) (2014)
14. Kohonen, T.: Self-Organization and Associative Memory, 2nd edn. Springer, Heidelberg (1987). https://doi.org/10.1007/978-3-642-88163-3
15. Murdoch, S.J., Danezis, G.: Low-cost traffic analysis for Tor. In: Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P 2005), pp. 183–195. IEEE (2005)
16. Natarajan, A., Ning, P., Liu, Y., Jajodia, S., Hutchinson, S.E.: NSDMiner: automated discovery of network service dependencies. In: Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOMM 2012). IEEE (2012)
17. Oler, K., Choudhury, S.: Graph based role mining techniques for cyber security. In: Proceedings of the FloCon. CERT (2015)
18. Sayegh, N., Elhajj, I.H., Kayssi, A., Chehab, A.: SCADA intrusion detection system based on temporal behavior of frequent patterns. In: Proceedings of the 17th IEEE Mediterranean Electrotechnical Conference (MELECON). IEEE (2014)
19. Thomas, B.: Teamviewer authentication protocol (part 1 of 3), 31 January 2013. https://www.optiv.com/blog/teamviewer-authentication-protocol-part-1-of-3. Accessed 1 Jan 2018
20. Thomas, B.: Teamviewer authentication protocol (part 2 of 3), 31 January 2013. https://www.optiv.com/blog/teamviewer-authentication-protocol-part-2-of-3. Accessed 1 Jan 2018
21. Yin, J., Zhao, X., Tang, Y., Zhi, C., Chen, Z., Wu, Z.: CloudScout: a non-intrusive approach to service dependency discovery. IEEE Trans. Parallel Distrib. Syst. **28**(5), 1271–1284 (2017)