Identifying Anomalies in Event Time Series Data Using Neural Spike Train Distance Measures

Jonathan Marty[1], Giovanni Di Crescenzo[1], and Stanley Pietrowicz[1]

Perspecta Labs, Basking Ridge, NJ 07920, USA
{jmarty,gdicrescenzo,spietrowicz}@perspectalabs.com

Abstract. Network traffic often exhibits periodic and predictable patterns, as the traffic is often associated with periodic events. The ability to detect deviations from expected patterns of events is a valuable security tool as it enables network operators to accurately detect anomalies. While a number of useful approaches exist for finding statistical deviations in network traffic patterns, we are not aware of a current approach that defines a distance measure between two series of network events.
Measuring the distance between two series of events has been studied extensively in the context of neurology, specifically in measuring the difference between two series of electrical discharges between neurons. These discharges are know as spikes, and a series of spikes between neurons resulting from some stimulus is referred to as a spike train.
In this paper we explore three common spike train distance measures, extend these measures to make them appropriate for network traffic event distance measurement, and evaluate their properties under various conditions. Additionally, we present a real-world example using this method to detect anomalies in control messages from a production power grid.

1 Introduction

In this paper we introduce a method to enhance network security by extending neural spike train distance measures to find anomalies in event time series data. Event time series occur frequently in many areas of networking. In industrial control systems network messages often correspond with actions in the physical world such as reading a meter or opening and closing a valve. Events can also be data-centric such as the times at which a data backup starts and finishes or the completion time of a periodic download of public data and the amount of data retrieved, network-centric such as the number of new and changed routes during a routing table update or a periodic count of network health checks warnings, and application-centric such as the time of a scheduled report generation or the time and duration of Java full garbage collections.

Another example of an event time series is neuronal spiking. Each neuron in the brain has a voltage across its membrane, referred to as a membrane potential. Factors internal or external to the neuron cause changes in the membrane potential. If the potential grows large enough, it can trigger a spike, or the release of an action potential down the neuron's axon. Spikes between neurons are

a natural event time series because once a spike starts it completes in a consistent way, regardless of the amplitude of the stimulus that caused it. As such, spike train data can be represented by the discrete timings of individual spikes. In neurology this is known as the all-or-none law [1].

The function of our brain is defined by these electrical spikes, which encode transmission of data around the nervous system. A series of spikes between neurons is referred to as a spike train. Understanding the nature and properties of these spikes is critical to understanding neuroscience, so there has been significant work done to study these spike trains and to measure the difference, or distance, between two spike trains.

In this paper we propose a method for finding anomalies in network traffic using neural spike train distance measures. We start by considering three common measures: Victor-Purpura [15], Kreuz [8] and van Rossum [11]. We observe that in neuroscience synaptic discharges are assumed to follow the all-or-none law, and thus a direct application of such distances to compare network events may miss significant event information. Accordingly, we propose extensions to these distance measures to incorporate both the magnitude and timing of each event, and evaluate the extended measures across panels of tests to better understand the dynamics of each.

To evaluate this method in real-world conditions, we apply it to production control network data from a commercial power grid. We introduce a novel method for mapping periodic network traffic to events, and propose an algorithm for a moving window distance measurement. We outline a practical approach to handling false positives by creating *whitelist* event signatures, and evaluate the performance under various attack patterns. We show that using the proposed method effectively isolated anomalous patterns in the event data under real-world conditions.

Section 2 of this paper outlines related work in this area. Section 3 examines the details of three common spike train distance measures. Section 4 proposes extensions to each of the spike distance measures to include spike magnitude. Section 5 evaluates these proposed measures under various time and magnitude scenarios. Section 6 evaluates the method in a real world example. Section 7 concludes this paper.

2 Related Work

Timing attributes, such as average packet inter-arrival time, can serve as indicators of cyber attacks. Lin et al. [9] propose a timing-based anomaly detection system for network traffic. The system models the mean and range of inter-packet times, which are used to define boundaries for normal activity. Points outside of these boundaries are deemed anomalous. Sayegh et al. [13] propose an intrusion detection system that uses histograms of inter-packet times to detect anomalies. Based on a selected set of fields, each packet is assigned a unique hash signature. Histograms are generated of the inter-arrival times between packets with different signatures, which are used to determine the anomalous packets.

While these are valuable techniques for aggregate network data flow analysis, they are not constructed to model temporal sequences that characterize event time series data. Additionally, as both measures were built to operate on packet data, they have no facility to measure differences in the magnitude of events.

Sequence-aware intrusion detection algorithms have emerged in recent years. These algorithms model the order of messages. They typically implement state machines, with states modeling different message types. Caselli et al. [3] proposes the modeling of sequences of protocol messages as Discrete Time Markov Chains (DTMC). The authors build a state machine, with states modeling possible messages. DTMCs are generated using both learning and testing data and the difference between the resultant models is measured using a metric. While this is a promising line of research, the practical aspects of developing and calibrating an DTMC are cubersome, and the resulting implementation will be complex and difficult to maintain leading to a strong assumption of stationarity in network traffic [16]. This technique is also insensitive to changes in timing if the order of messages is not changed, and has no facility to measure changes in event magnitude.

In contextual anomaly detection systems, a multivariate statistical model of the network data is generated, and outliers are detected based on observed deviations from this model. Algorithms such as K Nearest Neighbors (kNN) [5], Local Outlier Factor (LOF) [6], and Robust Principal Component Analysis (rPCA) [2] serve as the basis for a number of outlier detection models. The value of these models, and models derived from them, is in detecting deviations in correlations between signals, and does not address the temporal aspect of a single signal or event time series.

Spike train distance measures are a class of metrics designed to judge dissimilarity between time series of neuronal spikes, and have been successfully applied to the analysis of neural coding. Houghton and Victor [7] presents an overview of spike train distance measures, classifying metrics as kernel-based, synapse-based, or edit-distance, and provides mathematical frameworks and specific examples for each. In a similar manner, Victor [14] defines various classes of spike train metrics and explores their applications. Victor and Purpura, Kreuz, and van Rossum [15, 8, 11] propose spike train distance measures, which are described in detail in Section 3. A comparative analysis of various spike train metrics can be found in Satuvuori and Kreuz [12], which compares the sensitivities of measures to rate and time coding and explores their ability to deal with low spike rates.

3 Neural Spike Train Distance Measures

As a baseline for our work we look at three current methods used to measure the difference, or distance, between neural spike trains.

3.1 Victor-Purpura Distance

The Victor-Purpura distance is an edit-distance metric. It measures the distance between two spike trains based on the minimum number and type of edits re-

quired to convert one spike train into the other. It was first introduced in [15], which defines two types of edits and their associated costs: spike insertion or deletion (cost 1) and shifting a spike over an interval Δt (cost q$|\Delta$t$|$). Here q is the cost per time unit and is inversely related to the time scale τ of the analysis. Values of q close to 0 make the metric more sensitive to differences in the rate of spikes while large values of q make the metric more sensitive to differences in the timings of individual spikes.

3.2 Kreuz Distance

The Kreuz distance distance measure is introduced in [8]. A dissimilarity profile is calculated between two spike trains, taking into account differences in spike times. Calculating the profile is accomplished in two steps. First, for each spike, the distance to the nearest spike in the other spike train is calculated. Second, for each time instant, the relevant spike time differences are selected, weighted, and normalized. Each time instant has three quantities assigned to it for each spike train $u \in \{x, y\}$. These are the time of the preceding spike t_P^u, the time of the following spike t_F^u, and the interspike interval z_{ISI}^u. The time of the preceding and following spikes for a spike train u at time t can be determined:

$$t_F^u(t) = \min_i [t_i^u | t_i^u > t] \quad \text{and} \quad t_P^u(t) = \max_i [t_i^u | t_i^u \leq t] \tag{1}$$

and the inter-spike interval can be determined as:

$$z_{ISI}^u(t) = t_F^u - t_P^u \tag{2}$$

Each time instant is surrounded by 4 "corner" spikes, these are the preceding spike in the first spike train t_P^x, the following spike in the first spike train t_F^x, the preceding spike in the second spike train t_P^y, and the following spike in the second spike train t_F^y. Each of these "corner" spikes can be identified with a spike time difference.

$$\Delta t_P^x = \min_i (|t_P^x - t_i^y|) \quad \text{and} \quad \Delta t_F^x = \min_i (|t_F^x - t_i^y|) \tag{3}$$

For each spike train, a locally weighted average is applied so that the difference of the closer spike dominates; the weighted factor depends on the intervals to the preceding and following spikes:

$$z_P^u(t) = t - t_P^u(t) \quad \text{and} \quad z_F^u(t) = t_F^u(t) - t \tag{4}$$

The local weighting of the spike time differences for each spike train reads:

$$S_u(t) = \frac{\Delta t_P^u(t) z_F^u(t) + \Delta t_F^u(t) z_P^u(t)}{z_{ISI}^u(t)}, \quad u \in x, y \tag{5}$$

While this quantity takes into account the relative distances within each spike train, it does not take into account relative distances between spike trains. In order to account for the relative distances between spike trains as well as

differences in firing rates, the contributions from the two spike trains are weighted by their interspike intervals. This leads to the dissimilarity profile:

$$S_{x,y}(t) = \frac{S_x(t)z^y_{ISI}(t) + S_y(t)z^x_{ISI}(t)}{2\langle z^u_{ISI}(t)\rangle^2_{u\in\{x,y\}}} \tag{6}$$

Averaging $S_{x,y}(t)$ over $t \in 0, \ldots, T$ yields the Kreuz distance.

3.3 van Rossum Distance

The van Rossum distance is introduced in [11]. A convolution with an exponential kernel with time constant τ is performed on each spike t_k within the discrete spike trains. The kernel function is defined as:

$$k(t) = H(t)e^{-t/\tau} \quad \text{where} \tag{7}$$

where $H(t)$ is the Heaviside function which equals 1 for $t \geq 0$ and 0 otherwise.

The use of an exponential kernel is based on its resemblance to the shape of postsynaptic currents. The resulting waveforms $\tilde{x}(t)$ and $\tilde{y}(t)$ can be used to calculate the van Rossum distance D based on τ in the following way:

$$\tilde{x}(t) = \int_{-\infty}^{\infty} x(t-T)k(T)dT \tag{8}$$

$$D_{x,y} = \frac{1}{\tau}\int_{0}^{\infty}[\tilde{x}(t) - \tilde{y}(t)]^2 dt \tag{9}$$

The time constant τ acts as the parameter that sets the time scale. Low values of τ make the measure more sensitive to the differences in the timings of individual spikes, while high values of τ make the measure more sensitive to differences in the rates of spikes between the two spike trains.

4 Proposed Distance Measures

While synaptic discharges are assumed to follow the all-or-none law [1], in general the magnitude of network events holds significant information. Based on the framework described above, we propose extensions to the Victor-Purpura, van Rossum, and Kreuz distance measures which incorporate the magnitude of each spike.

4.1 Extended Victor-Purpura Distance

Our proposed extension to the Victor-Purpura Distance is an additional type of edit whereby the magnitude of a spike m_k is be altered. The cost of this transformation would be $a|\Delta m|$, where Δm is the change in spike magnitude

and a is the cost per unit of magnitude. Increasing the value of a relative to q increases the importance of magnitude differences in determining the value of the metric. Inversely, decreasing the value of a relative to q increases the importance of differences in the rate of spikes and spike timings in determining the value of the metric. In accordance with the intention of this change, the cost of spike addition and deletion would be am, where m is the magnitude of the spike being added or deleted and the cost of shifting a spike over some interval Δt and changing its magnitude by Δm would be $q|\Delta t| + a|\Delta m|$.

4.2 Extended Kreuz Distance

Our proposed extension to the Kreuz distance, which uses a vector-space, is to incorporate magnitude into the spike time difference. First we replace equation 3 to reflect the euclidean distance between the time and magnitude of the spikes where γ is a scaling factor used to determine the relative importance of differences of magnitude:

$$\Delta t_u^x = \min_i \sqrt{(t_u^x - t_i^y)^2 + \gamma(m_u^x - m_i^y)^2}, \quad u \in F, P \tag{10}$$

In a similar manner we change the inter-spike interval in Eq. 2 to:

$$z_{ISI}^u(t) = \sqrt{(t_F^u - t_P^u)^2 + \gamma(m_F^u - m_P^u)^2}, \quad u \in x, y \tag{11}$$

where $m_P^u(t)$ and $m_F^u(t)$ are the magnitudes of the preceding and following spikes in spike train u at time-step t.

4.3 Extended van Rossum Distance

The van Rossum distance adapts naturally the to incorporation of magnitude by changing Eq. 8 to:

$$\tilde{x}(t) = \int_{-\infty}^{\infty} x_m(t - T)k(T)dT \tag{12}$$

where $x_m(t)$ includes the magnitude of the spike at time t.

4.4 Equivalent van Rossum Distance

In Section 5 we'll show that both the Victor-Purpura and Kreuz measures exhibit similar properties in our suite of tests, however the van Rossum distance has results that are less similar. We find that replacing the exponential decay kernel in Eq. 7 with a square kernel:

$$k(t) = \begin{cases} 1 & \text{if } 0 < t < T \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

And replacing the calculation of the distance in Eq. 9 with:

$$D_{x,y}^{Equiv} = \frac{1}{\tau} \int_0^\infty |\tilde{x}(t) - \tilde{y}(t)| dt \tag{14}$$

yields a measure that's similar to the Victor-Purpura and Kreuz measures while being simpler to compute and more intuitive to understand. In the remainder of this paper we will refer to this measure as the *vanRossumEquiv* distance.

5 Evaluation of Distance Measures

To better understand the performance of the proposed distance measures we evaluated each extended distance across a panel of deterministic and random tests.

5.1 Incremental Time and Magnitude Response Tests

First, we performed tests of the sensitivity of each algorithm to changes in the magnitude and timing of spikes. In both time and magnitude tests, a base spike train is compared with an incrementally altered spike train. The base spike train consists of spikes every τ seconds with unit magnitude.

For the magnitude test, the altered spike trains had the same timing as the base, but the magnitude of all spikes was increased incrementally from 50% to 150% of the magnitude of the base spikes. The spike distances for each magnitude are plotted on the top half of Figure 1.

Similarly, for time sensitivity, the altered spikes train had the same magnitude and inter-spike interval as the base, but the timing of all spikes was offset by $-\tau/2$. This offset was increased incrementally from $-\tau/2$ to $\tau/2$, where a negative offset is a shift to the left.

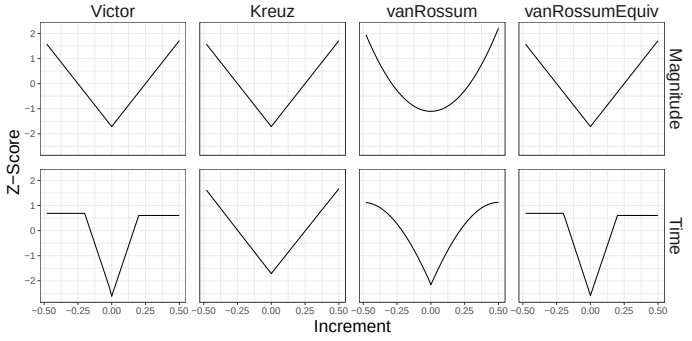

Fig. 1. Incremental Magnitude and Time Spike Train

Figure 1 plots the z-score of the spike train distances as a function of the offset in time and magnitude. The spike distances for each magnitude are plotted in the top half, while the spike distances for each time offset are plotted in the bottom half.

From these tests we can conclude that the Victor and Kreuz algorithms differ from the van Rossum algorithm in that their response to changes in spike timing and magnitude are linear. Note that changing the kernel used in the van Rossum convolution from exponential to square as described in Section 4.4 yields the same linear response.

The Victor measure shows a maximum impact of varying spikes in time because at some point it's cheaper to replace a spike than it is to move it. The vanRossumEquiv measure has a similar property which occurs when the convolutions of the square wave between the two time series no longer overlap.

5.2 Random Time and Magnitude Trials

Random Time Trial: In the time test, the altered spike train had the same magnitudes as the base, but timing for each spike was offset individually by τX where X is randomly selected from a uniform distribution between $[-0.5, 0.5]$. The results from this test are shown in Figure 2. The scatter plots above the diagonal in Figure 2 compare the z-scores of the respective outputs, while values below the diagonal note the correlation and confidence intervals for the pairs of algorithms. Charts along the diagonal show the empirical probability density function of the respective algorithms.

From Figure 2 we see that the Victor, Kreuz, and vanRossum algorithms all have a similar response to variations in spike timings.

The difference between the Victor and vanRossum algorithms is primarily a result of the exponential decay kernel and the $\mathcal{L}2$ summation function used in the vanRossum algorithm. When we change the vanRossum kernel from exponential to square as defined in Equation 13 and change the summation function from $\mathcal{L}2$ to $\mathcal{L}1$ as defined in Equation 14, the resulting metric (vanRossumEquiv in Figure 2) is very similar to the Victor metric with respect to time

Variation between the Victor and Kreuz algorithms is driven by the maximum time cost exhibited in the Victor algorithm and shown in the bottom left panel of Figure 1. This maximum time cost in the Victor algorithm is the point where it's less costly to replace a spike than to move it. We find that as we increase the relative cost of replacing a spike in the Victor algorithm, the response of the Victor and Kreuz algorithms with respect to spike timing converges.

Based on the above observations, we can make the transitive argument that the Kreuz and vanRossumEquiv algorithms differ primarily because of the maximum cost associated with a difference in spike timing. In the vanRossumEquiv algorithm, this maximum occurs when square waves in the convolved time series do not overlap. We can increase the correlation between Kreuz and vanRossumEquiv by increasing the length of the square kernel defined in Equation 13.

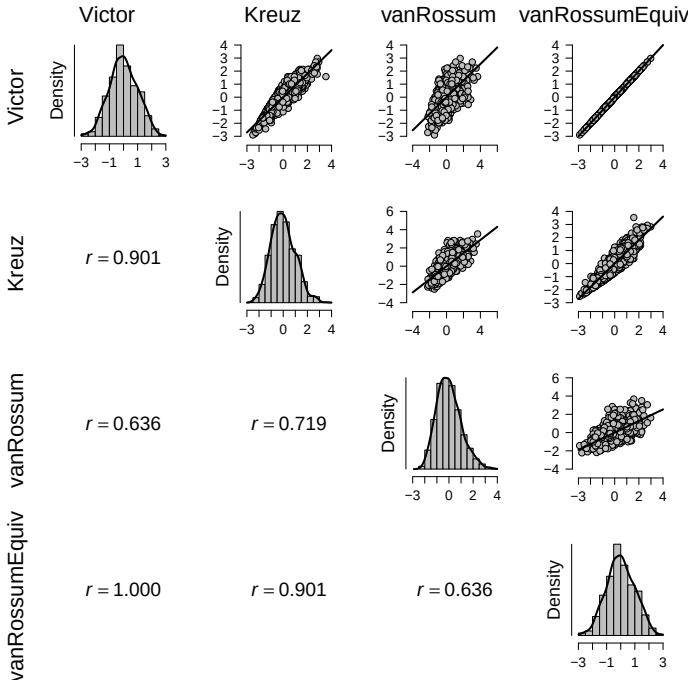

Fig. 2. Random Time Noise Spike Train Distribution

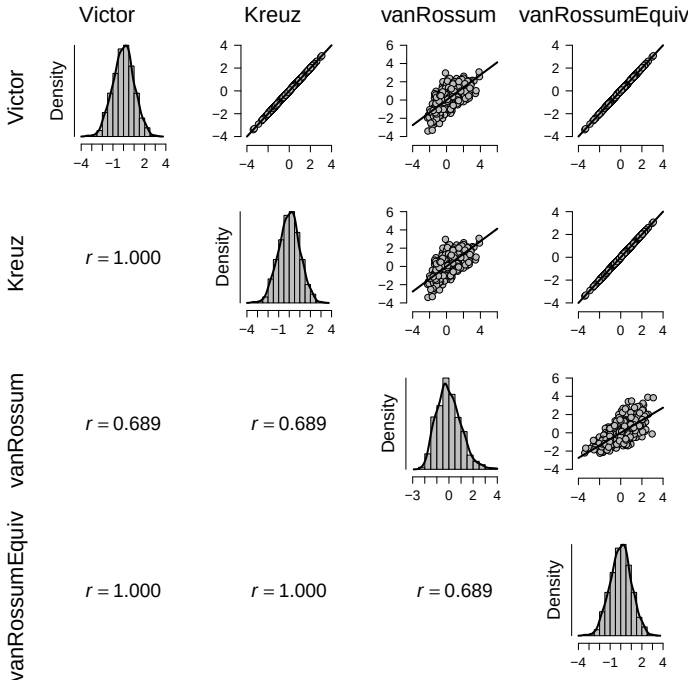

Fig. 3. Random Magnitude Noise Spike Train Distribution

Random Magnitude Trial: In the magnitude test, the altered spike trains have the same timing as the base, but the magnitude of each spike is multiplied by X, where X is randomly selected from a uniform distribution between $[0.5, 1.5]$. The results of this test are shown in Figure 3 which shows the correlations and distributions of each algorithm in a format similar to Figure 2.

From Figure 3 we see that the response to changes in spike magnitude between the Victor, Kreuz, and vanRossumEquiv algorithms are all similar. This is a result of extending each algorithm in a consistent way in Section 4.

The difference between the vanRossum algorithm and the other algorithms is fully described by differences between the vanRossum and vanRossumEquiv algorithms defined in Section 4.4.

Summary of Results: While all three measures are similar, each has properties that may make it appealing in a particular situation. The Kreuz measure has the property that there are no configurable parameters other than the relative importance of magnitude γ introduced in our changes. While this property is interesting, it comes at a cost of algorithmic complexity and heavy computational load. The Victor measure, with its costs for adding/removing/moving spikes is certainly the most intuitive and is easy to extend for additional distance metrics. The algorithm to compute this cost, while not trivial, is computationally efficient. For our use we prefer the van Rossum algorithm, as it's the most computationally efficient, and it's mathematical foundation in basic convolution lends itself to flexibility. Note that we could use any of the methods with equal effect.

6 Practical Application

In this section we apply the spike distance measure to an actual event time series in a production power grid control system. In Section 6.1 we describe the time series studied, in Section 6.2 we develop a mapping of continuous time data to temporal event data, in Section 6.3 we create a spike distance time series from the event time series, in Section 6.4 we introduce a method for handling false-positive results, in Section 6.5 we examine the performance of the metric under various attack patterns, and in Section 6.6 we discuss results and limitations of the method.

6.1 Event Time Series Studied

The time series studied is the count of power grid smart *meter reads*, collected from a commercial power grid using the Perspecta Labs SecureSmart$^{\text{TM}}$ power grid security system [10]. A sample day of this time series is shown in the top panel of Figure 4.

This time series exhibits a bursty traffic pattern with periods of high traffic separated by periods of roughly no traffic. An uncompromized, properly functioning system will exhibit a series of meter read counts that follows a consistent

schedule across days. In our analysis, the binary on-off nature of this traffic flow made it hard to analyze using traditional intrusion detection techniques.

6.2 Mapping Network Traffic to Spikes

Like many network events, the *meter reads* data exhibits bursty periods of network traffic with relative quiet between bursts as shown in the top panel of Figure 4. While we could coarsely classify each of these bursts of activity as an event, we chose to further decompose this traffic into a series of spikes that demarcate the beginning and end of the periods of traffic, and capture the magnitude of the bursts. Note that many time series exhibit discrete event patterns and will not require such a transformation.

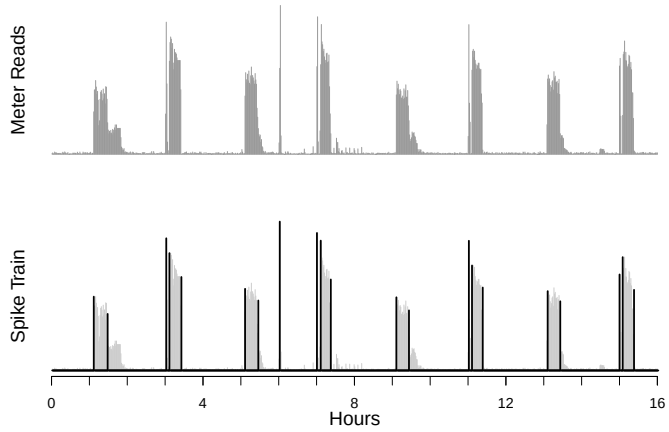

Fig. 4. Western Electric Rules Spike Train

The Western Electric Rules (WER) [4] were developed in 1956 as a method for finding non-random events in statistical charts. A modified version of the Western Electric Rules is used to detect singular events in continuous time series data. Here, we used 3 of the Western Electric Rules for determining the location and magnitude of spikes.

The original WER method qualified a point as a spike if that point had a z-score exceeding 3, if both the point and the following point had a minimum z-score exceeding 2, or if the point and the following 3 points had a minimum z-score exceeding 1. However, given that network data contains periods of sustained high activity (referred to here as extended spikes), using base WER would lead to too many spikes.

To remedy this, a stateful system based on the WER was developed, which varied between spiking and non-spiking states. Starting from a non-spiking state, the system switches to a spiking state if a point satisfies any of the WER, leaving

a spike at that point with a magnitude equal to the max value of the current point and the next 3 points. The system switches back to a non-spiking state if a point has a negative z-score or if the next 3 points have a z-score less than 1. When the system switches to a spiking state, it increments a counter for each time-step until it switches back to a non-spiking state. If the counter exceeds 5, an additional spike is placed where the system transitions back to a non-spiking state, marking the spike as an extended spike. This spike has a magnitude which is the mean value of the points within the extended spike.

The output of this system for a sample of the *meter reads* event data used is shown in Figure 4 where the top panel shows the original time series and the bottom panel shows the spikes generated from this algorithm superimposed on the original *meter reads* time series.

6.3 Creating Spike Distance Time Series

Our baseline expectation is that the event time series is repeated daily. As such, we use previous days as our benchmark of a proper series of events. This approach could easily be adapted to measure distance against a known-good series of events.

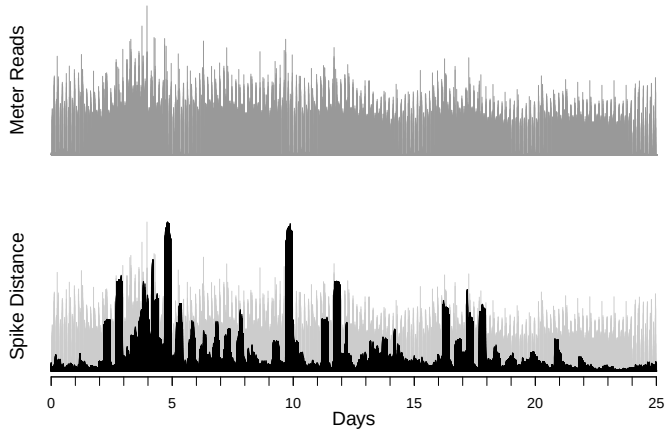

Fig. 5. Spike Distance Time Series

Using the original time series of event data transformed into spikes using our modified WER approach, we ran the van Rossum algorithm to calculate the distance between a 6 hour window and same 6 hour window for each of the previous 5 days. We used the median of the 5 distances as the spike distance for that point in time.

The time series of event data, along with the derived spike distance measure, is shown in Figure 5. The top panel is a 25 day sample of the original event time

series, and the bottom panel shows our spike distance measure superimposed on the original time series.

6.4 Handling False Positives

The top panel of Figure 6 shows the spike distance time series of our data. The horizontal dashed line marks 2 standard deviations above the mean. Examining this time series, we find that there are patterns of double spikes on days 3,5,9,12,17, and 18. In examining the raw data we see that these days include a pair of spikes at 4am and 4pm that do not exist in other days.

While these days may appear anomalous when compared to adjacent days, we know that these additional spikes are normal behavior and the spikes those days are best classified as false positive. Visual inspection of day 3 confirms that the events observed are normal and should not trigger an alarm.

We make the statement that "if a day looks like day 3 it's not anomalous," by adding day 3 to a *whitelist* database of known good event signatures. We enforce this in code by changing the spike distance time series algorithm to be the minimum of the event pattern with each of the whitelist series, and the median of the past 5 days. This is more clearly stated in Equation 15 where $sd_{t-1}, \ldots, sd_{t-5}$ are the spike distances between the current period and the same period over the past 5 days, and $sd_{w,1}, \ldots, sd_{w,N}$ are the spike distances between the current period and each of the N whitelisted event signatures:

$$min(median(sd_{t-1}, \ldots, sd_{t-5}), sd_{w,1}, \ldots, sd_{w,N}) \tag{15}$$

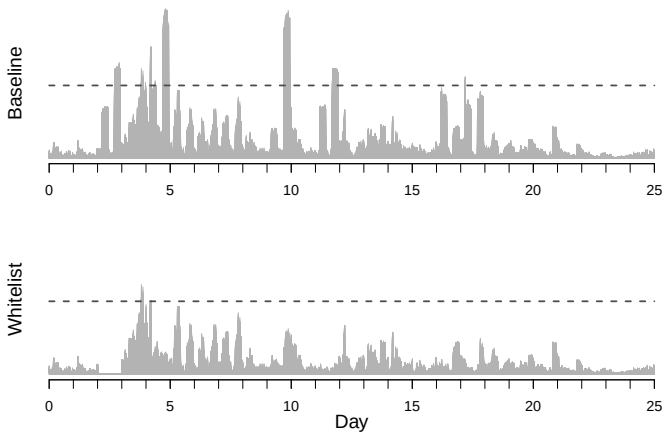

Fig. 6. Spike Distance Whitelist

The bottom panel of Figure 6 shows the time series of spike distances with day 3 whitelisted. Note that the anomalies previously observed on 5,9,12,17,

and 18 are effectively removed. Also note that the event data at the end of day 4 did show a marked increase in the number of meter reads, thus we see a spike distance level that exceeds our threshold. We consider this true anomalous behavior and worthy of inspection.

While managing the *whitelist* signature database is work for systems administrators, it's a practical and transparent method for addressing repeated false positive results. One could also track the number of times each whitelist signature is matched as it lends further transparency. This could also be extended to include a *blacklist* signature database that contains bad event patterns that require immediate escalation.

6.5 Detecting Anomalies

While the proposed method is well suited for detecting complex differences in event time series data, the *principal components* of these differences are adding a spike or removing a spike (or moving a spike which is removing plus adding), and changing the magnitude of a spike.

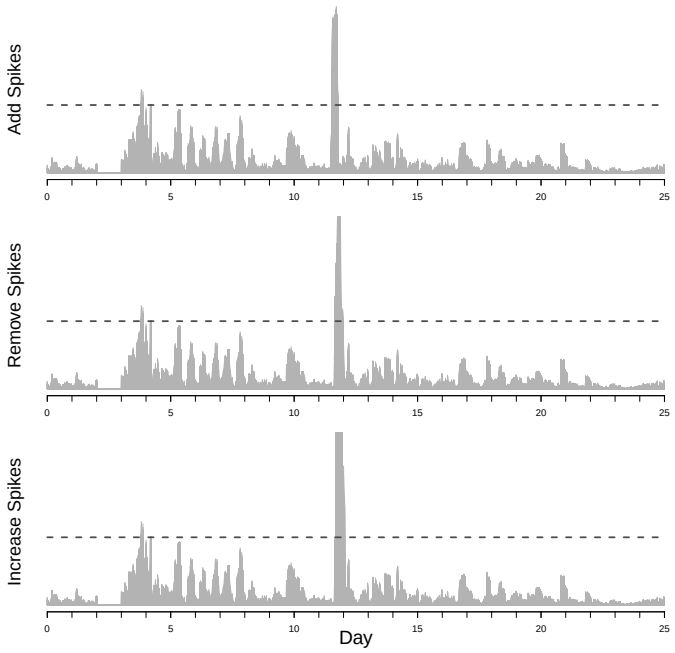

Fig. 7. Spike Distance Anomalies

Using intuition from Caselli et al. [3] we create a histogram of inter-event times and find that our modes are intervals of roughly 5, 20, and 100 minutes.

In the top panel of Figure 7 we take a 100 minute inter-event period and inject events spaced at 20 minutes. While this has a minimal effect on the relative histogram levels, it shows a pronounced spike in the spike distance measure. It would be trivial to create daily samples that have identical histograms but show very large spike distance values.

In a similar manner, the middle panel of Figure 7 shows the effect of removing events spaced at 5 and 20 minutes to create an inter-event time of 100 minutes. The result is clearly shown as a spike in day 12. The bottom panel of Figure 7 show the effect of increasing the magnitude of spikes in day 12 by 25%. This, again, is clearly shown in the corresponding spike distance measure.

6.6 Summary of Implementation Results

In practice we find that this method is effective in adding another important facet to a multi faceted security approach. It fills an important gap in a way that's both theoretically elegant and practical to implement. This method is well suited for application to event time series data that follow predictable patterns, though these patterns can be determined either explicitly or empirically. Event time series that do not exhibit predictable patterns, but rather have a predictable distribution of inter-arrival times are better handled by Lin et al. [9] or Sayegh et al. [13], and we suggest that these methods are complementary to our proposed method.

While no anomaly detection scheme is free from the possibility of false positive results, we find that the method proposed to handle false positive results significantly increases the fidelity and manageability of the system. We find that the simplicity and transparency of this method lends itself to being practical to operate when compared to more complex methods such as Caselli et al. [3].

In a separate line of research we find that replacing an event time series with a spike distance time series is a valuable method for integrating event data into traditional intrusion detection frameworks.

7 Conclusion

This paper explores the use of neural spike train distance measures to assist in detecting anomalies in network event time series data. We explore three common spike train distance measures and propose extensions to each to include the magnitude of spikes. We compare properties of these measures to variations of time and magnitude, and we examine their similarity over random trials. We explore details of a production implementation of the technique and find that using spike distance measures adds a valuable facet to a comprehensive security infrastructure.

References

1. Bishop, G.H.: Natural history of the nerve impulse. Physiological Reviews **36**(3), 376–399 (1956). https://doi.org/10.1152/physrev.1956.36.3.376, https://doi.org/10.1152/physrev.1956.36.3.376, pMID: 13359129
2. Candès, E.J., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? Journal of the ACM (JACM) **58**(3), 11 (2011)
3. Caselli, M., Zambon, E., Kargl, F.: Sequence-aware intrusion detection in industrial control systems. In: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security. pp. 13–24. ACM, 2 Pennsylvania Plaza 701, New York, NY 10121 (2015)
4. Company, W.E.: Statistical quality control handbook. Western Electric Company (1958), https://books.google.gr/books?id=gj8IAQAAIAAJ
5. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE transactions on information theory **13**(1), 21–27 (1967)
6. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern Recognition Letters **24**(9-10), 1641–1650 (2003)
7. Houghton, C., Victor, J.: Measuring representational distances–the spike-train metrics approach. Visual Population Codes–Toward a Common Multivariate Framework for Cell Recording and Functional Imaging pp. 391–416 (2010)
8. Kreuz, T., Chicharro, D., Greschner, M., Andrzejak, R.G.: Time-resolved and time-scale adaptive measures of spike train synchrony. Journal of neuroscience methods **195**(1), 92–106 (2011)
9. Lin, C.Y., Nadjm-Tehrani, S., Asplund, M.: Timing-based anomaly detection in scada networks. In: Proceedings of the 12th International Conference on Critical Information Infrastructures Security. Telecom Italia, The Fraunhofer Institute for Intelligent Analysis and Information Systems, Schloss Birlinghoven, 53757 Sankt Augustin, Germany (2017)
10. Pietrowicz, S.: Cybersecurity intrusion detection and monitoring for field area network. Tech. rep., Applied Communication Sciences (Vencore Labs) (2016)
11. van Rossum, M.: A novel spike distance. Neural computation **13**(4), 751–763 (2001)
12. Satuvuori, E., Kreuz, T.: Which spike train distance is most suitable for distinguishing rate and temporal coding? Journal of Neuroscience Methods **299**, 22 – 33 (2018). https://doi.org/https://doi.org/10.1016/j.jneumeth.2018.02.009, http://www.sciencedirect.com/science/article/pii/S0165027018300372
13. Sayegh, N., Elhajj, I.H., Kayssi, A., Chehab, A.: Scada intrusion detection system based on temporal behavior of frequent patterns. In: Electrotechnical Conference (MELECON), 2014 17th IEEE Mediterranean. pp. 432–438. IEEE, 445 Hoes Ln, Piscataway Township, NJ 08854 (2014)
14. Victor, J.D.: Spike train metrics. Current Opinion in Neurobiology **15**(5), 585 – 592 (2005). https://doi.org/https://doi.org/10.1016/j.conb.2005.08.002, http://www.sciencedirect.com/science/article/pii/S0959438805001236, neuronal and glial cell biology / New technologies
15. Victor, J.D., Purpura, K.P.: Metric-space analysis of spike trains: theory, algorithms and application. Network: computation in neural systems **8**(2), 127–164 (1997)
16. Ye, N., Zhang, Y., Borror, C.M.: Robustness of the markov-chain model for cyber-attack detection. IEEE Transactions on Reliability **53**(1), 116–123 (2004)