Timing Patterns and Correlations in Spontaneous SCADA Traffic for Anomaly Detection

Chih-Yuan Lin
Linköping University, Sweden
chih-yuan.lin@liu.se

Simin Nadjm-Tehrani
Linköping University, Sweden
simin.nadjm-tehrani@liu.se

Abstract

Supervisory Control and Data Acquisition (SCADA) systems operate critical infrastructures in our modern society despite their vulnerability to attacks and misuse. There are several anomaly detection systems based on the cycles of polling mechanisms used in SCADA systems, but the feasibility of anomaly detection systems based on non-polling traffic, so called spontaneous events, is not well-studied. This paper presents a novel approach to modeling the timing characteristics of spontaneous events in an IEC-60870-5-104 network and exploits the model for anomaly detection. The system is tested with a dataset from a real power utility with injected timing effects from two attack scenarios. One attack causes timing anomalies due to persistent malfunctioning in the field devices, and the other generates intermittent anomalies caused by malware on the field devices, which is considered as stealthy. The detection accuracy and timing performance are promising for all the experiments with persistent anomalies. With intermittent anomalies, we found that our approach is effective for anomalies in low-volume traffic or attacks lasting over 1 hour.

1 Introduction

Supervisory Control and Data Acquisition (SCADA) systems are used for monitoring and controlling critical infrastructures such as waste water distribution facilities, gas production systems, and power stations. In the past decades, SCADA systems have increasingly adopted open protocols and connected to the Internet for improved flexibility and ease of use. These changes make SCADA systems vulnerable to cyber attacks, in particular with Advanced Persistent Threats (APT). Such attacks are usually slow and stealthy but ultimately lead to sever damage on physical devices.

One building block for protection of these systems is the implementation of anomaly detection. Most earlier works on the network-based anomaly detection exploit the cyclic traffic patterns found in the request-response communication mode [3, 14, 9]. That is, the SCADA master cyclically sends a request to field devices such as a Remote Terminate Unit (RTU) and a Programmable Logic Controller (PLC) and the field devices return monitored values after receiving a request. However, some SCADA protocols also allow non-requested communication whereby the field devices can report monitored values in spontaneous events without receiving any request. In a previously proposed anomaly detector [18], the authors observed low detection rates on an IEC-60870-5-104 (from now referred to as IEC-104) dataset because most of the communications were issued in non-requested mode. This motivates the search for modeling approaches that deal with spontaneous event sequences used for anomaly detection.

Anomaly detection is a technique that captures stable characteristics of spontaneous traffic and identifies unusual behaviours. In related research [17], Lin and Nadjm-Tehrani characterized the timing attributes of two emulated IEC-104 datasets. The datasets show a diverse set of possible timing behaviours and many of them exhibit event sequences varying in an irregular manner. However, process dynamics might imply two stable attributes over a longer period: (1) The probability distribution of event inter-arrival times shows clear peaks. That is, some inter-arrival times are more likely to be present than others, and (2) The probability distribution of inter-arrival times changes in groups of different data flows in the same system. The flows in the same group tend to change at approximately the same time. That is, there could exist positive correlation between the occurrence of spontaneous events in different flows.

In this work, we model the timing attributes of spontaneous traffic from a real power facility based on the above two hypotheses. We also propose an anomaly detector to detect anomalies within such traffic. The proposed detector is tested with two attack scenarios at the field device level. Our threat model is as follows. A field device such as RTU or PLC is an interface between the field sensors, actuators and SCADA master. Instead of directly breaking a field device, attackers may target disrupting the controlled process and/or

communication between the field device and SCADA master in a gradual manner. Our goal is to detect anomalies caused by the attacks before they cause irreparable damages on critical infrastructures. The contributions of this work are:

- Provide an empirical study on the timing attributes of spontaneous traffic from a real power facility. The study shows that inter-arrival times and correlation between flows are stable and there are persistent timing characteristics in spontaneous traffic.
- Present two attack scenarios producing timing anomalies in the spontaneous traffic and provide the corresponding anomaly generation approaches.
- Explore the potential of anomaly detection for IEC-104 spontaneous traffic by combining two methods for modeling event inter-arrival times and correlation between flows in the same network.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 provides the background knowledge needed for this work. Section 4 discusses the threat models and caused traffic anomalies. Section 5 describes the components of the proposed anomaly detection and evaluation system. Section 6 provides an overview of the datasets and the parameter settings. Section 7 reports the experimental methods and results. Section 8 concludes this study.

2 Related Work

Various anomaly detection models have been proposed for SCADA systems with different research methodologies. To better position our work, this section reviews related work in three orthogonal research domains: (1) network-based anomaly detection, (2) physic-based anomaly detection, and (3) content-based anomaly detection for SCADA systems.

2.1 Network-based Anomaly Detection

Network-based anomaly detection models target how the components in a SCADA system communicate to each other instead of looking into the payload. Many of the network-level detectors make use of the cycles of SCADA traffic with a focus on their cyclic sequences.

Timing-based anomaly detection usually employs statistical models and raises alarms when the monitored statistical parameters exceed certain thresholds. Valdes and Cheung [27] capture the attributes *bytes per packet* and *packet inter-arrival time* and compare the testing traffic against the historical records by T-test. Sayegh et al. [25] model the historical time intervals between signatures (i.e., a fixed sequence of packets) and Barbosa et al. [3] model the historical period of repeated messages in an orderless group. Lin et al. [18] focus on the inter-arrival time for periodic events and propose a modeling approach based on the sampling distribution of mean and range.

Sequence-aware intrusion detection systems usually employ automata-based models. Goldenberg and Wool [9], Kleinmann et al. [15, 14], and Yang et al. [30] use Deterministic Finite Automata to model the message sequences of Modbus[1], Siemens S7[2], and IEC-104 traffic respectively. Yoon et al. [31] model the Modbus command sequences as Dynamic Bayesian Network.

Recently, more non-cyclic traffic has been found in studies with production data. Casselli et al. [5] propose a methodology to model the message sequences of Modbus, MMS[3], and IEC-104 in Discrete-time Markov Chain. The authors observe no clear cyclic message patterns in the IEC-104 datasets. Faisal et al. [7] apply the Goldenberg and Wool's approach on a large dataset collected from a water facility in the U.S. and show that the model performs poorly because 72% of the flows did not exhibit clear cyclic patterns. Later, Markman et al. [21] propose a variant of the Goldenberg and Wool's model by considering the traffic on each flow as a series of bursts and create a burst-deterministic finite automata for each flow. This methodology is tested with the same data used in Faisal et al.'s study and 95% to 99% of the packets are successfully captured by this model.

Our work differentiates itself from the previous work by providing dedicated anomaly detection methodologies for sopntaneous SCADA traffic while others focus on the cyclic attributes. In an earlier work by Lin and Nadjm-Tehrani [17], where they analyze the IEC-104 spontaneous traffic and show that the emulated traffic contains some stable timing attributes other than cyclic sequences and periodicity, no anomaly detection is proposed. In this work, we propose an anomaly detection system for IEC-104 spontaneous traffic and test the system with data collected from a real power facility.

2.2 Physics-based Anomaly Detection

Physics-based anomaly detection models the physical process managed by SCADA systems. Physics-based approaches model the process and detect the anomalies with high accuracy while network-based approaches model the network behaviour without knowing process semantics but aim to alarm the users before the attack has major impacts on the process. We consider these two approaches complementary not competing with each other. Giraldo et al. [8] presents a literature review on physics-based anomaly detection.

System identification techniques can model behaviours of a physical system by its inputs and outputs. Two models that

[1] Modbus. http://www.modbus.org/

[2] A proprietary protocol used Siemens S7 series PLCs.

[3] Manufacturing Message Specification (MMS). https://www.iso.org/standard/37079.html

are widely used are the *Linear Dynamic State-space (LDS)* [26, 19] and the *Auto-Regressive (AR)* [11] models. These models may accurately predict the system behaviour, but they require a detailed description of the process and deep understanding of system that are not always available.

Machine learning approaches require less or no prior knowledge of the underlying process. Krotofil et al. [16] use the correlation entropy in clusters of related sensors to detect sensor signal manipulations. Kiss et al. [13] adopt the Gaussian mixture model to form sensor clusters. Aoudi et al. [2] propose a departure-based detection system that measures the distance between the normal signals and the signals under attack projected to a subspace. These works show that sensors in SCADA systems are intricately correlated. Since the sensor values and spontaneous events have a cause-effect relationship, these works explain and support our hypothesis that spontaneous traffic from different flows can be correlated.

2.3 Content-based Anomaly Detection

Content-based anomaly detection based on in-depth analysis of packet contents is an important research topic for general-purpose networks. Due to the use of proprietary protocols and lack of availability of specifications, content-based anomaly detection for SCADA systems is still in its infancy. Düssel et al. [6] present an anomaly detection system based on *n*-grams using distance metrics. Hadžiosmanović et al. [10] investigate four anomaly detection algorithms, POSEIDON, Anagram, PAYL, and McPAD, all using *n*-gram analysis for message payloads. The authors conclude that there is no absolute best algorithm among the tested methods in terms of detection rates and false positive rates. Wressnegger et al. [29] propose an anomaly detection system based on *n*-grams with a special focus on proprietary binary protocols. The authors show that the content-based approach is applicable to binary protocols with high-entropy data.

3 Preliminaries

This section briefly presents how an IEC-104-compatible RTU works and generates spontaneous events, together with important fields of IEC-104 packets. The section also provides a short review on spontaneous traffic attributes found in a previous study.

3.1 IEC-60870-5-104

The IEC-104 protocol applies to modern SCADA systems for monitoring and controlling geographically widespread processes. It enables communication between a master (control station) and one or more slaves (substations) via a standard TCP/IP network. IEC-104-compatible RTUs or PLCs store inputs from the controlled process in its own storage which is indexed by Information Object Addresses (IOA). In order to improve the communication efficiency, IEC-104-compatible RTUs scan monitored data in certain IOAs with a fixed rate and generate spontaneous events when the monitored data have changed.

Every spontaneous packet contains two important fields in addition to IOA. First, Cause Of Transmission (COT) specifies whether it is a spontaneous packet or other type of packet such as a reply on interrogation or a cyclic data report. Second, Type IDentification (TID) specifies the type of the monitored data. The most common data type is Monitored MEasured point, normalized value (M_ME_NA) or its variant (M_ME_TA and M_ME_TD). This datatype is 16-bit long and contains a measured value from a certain IOA. The system administrator needs to define an acceptable range for each measured point. The RTU sends out a spontaneous event when the measured value falls outside the range. Monitored Single Point (M_SP_NA) is 1-bit long and Monitored Double Point (M_DP_NA) is 2-bit long representing the state of a switch or circuit breakers [4]. A RTU sends out spontaneous events if the value of these two datatypes changes.

3.2 Known Spontaneous Traffic Attributes

This work is motivated by earlier results [17] which show that the IEC-104 spontaneous traffic contains some timing attributes that could last for a long time. Our notion of flow is based on a sequence of timed events with a given IEC-104 type, from a given RTU, based on values stored in a given IOA. There are two attributes explored this study. (1) Inter-arrival time attribute: assuming that the spontaneous traffic has limited groups of event inter-arrival times, and (2) correlation attribute: assuming that positive correlation exists between the changes on inter-arrival times in different flows.

To illustrate these attributes, we present the partial characterization results of two flows, named IOA_10091 and IOA_10092, which are initiated from the same RTU but different IOAs in a virtual SCADA testbed, RICS-el, developed in a national research project [1]. The flows are from a 12 day collection and separated into 2-hour long segments for characterization. We choose these flows because they show strong characteristics, but the other flows in the same collection also contain similar attributes.

Inter-arrival time attribute. Figure 1 (a) presents the histogram of event inter-arrival times within a segment of two hours from IOA_10091. There exists clear peaks with high frequency and several groups of inter-arrival times with low frequency in the collected data over two hours. The analysis of the collected data shows that few of inter-arrival intervals fall outside these groups in the subsequent 12 days.

[4] IEC 60870-5-104 summary. https://spinengenharia.com.br/en/biblioteca/protocolo-iec-60870-5-104-client/download/

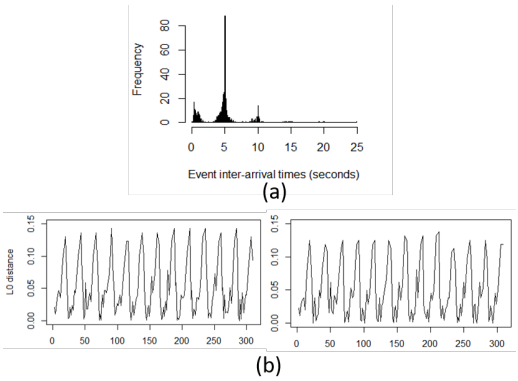


Figure 1: Traffic characteristics found in the collected data. (a) Histogram of spontaneous event inter-arrival times from IOA_ 10091. (b) L0 distances over time (hours) for two spontaneous traffic flows from the same RTU but different IOAs [17].

Correlation attribute. Continuing with the above example, we get several groups (i.e., peaks) of inter-arrival times in the first segment of IOA_10091 and IOA_10092. The empirical probability of each group can be easily calculated as number of observations in the group divided by number of observations in the segment. Figure 1 (b) shows the changes in the probability distribution of each segment over 12 days. The x axis gives the relative time in every two hours (a segment). The y axis gives the distance of probability distribution between the first segment and the later ones. The distance is defined as

$$D = \sum_{\sigma_i^1 = \sigma_i^2} (p(\sigma_i^1) - p(\sigma_i^2))^2 + \sum_{\sigma_i^1 \neq \sigma_i^2} p(\sigma_i^k)^2 \qquad (1)$$

where σ_i is the i^{th} group of inter-arrival times in the set of identified groups, and σ_i^k means the i^{th} group in distribution k, $k = 1$ or 2. $k = 1$ means the distribution of the first two hours and $k = 2$ means another distribution in comparison. $p(\sigma_i^k)$ thus means the empirical probability of the group. Previous work does not calculate any type of correlation coefficients for the datasets. It only categorizes the traffic *for each flow* by observing changes of L0 distances over time.

In Section 5.1 we will enrich the modeling of the spontaneous traffic by exploiting both attributes. For the sake of simplicity, we refer to the model based on the first attribute as *inter-arrival time model* and the second one as *correlation model* in the rest of the paper.

4 Threat Model

This section presents two types of attacks on the field device level and the possible traffic anomalies caused by the attacks. This provides an intuitive view on the need for anomaly detection.

In order to disrupt the controlled process and communication, an attacker can (1) take control of other components and launch attacks such as Denial-of-Service (DoS) to the targeted field devices or (2) exploit code-integrity vulnerabilities in the targeted field devices and run malware on them.

Attack against field devices. Legacy SCADA field devices have limited computing capabilities so that they are fragile and sensitive to network traffic increment [4, 28]. Niedermaier et al. [24] showed that most of the PLCs from major vendors are vulnerable to packet flooding in different network levels and even scanning for benign purposes. The attacker can influence the PLC cycle times and the controlled processes. With regards to the network anomalies, this type of attacks causes network performance degradation due to competition on resources such as CPU and I/O port of the devices. Long et al. [20] model the network performance under DoS attack with two queuing models but no real attacks and devices are involved. Other authors [23, 12] simulated the impact of different DoS attacks on SCADA networks. Both simulations showed performance impacts including network delays, packet drops, and unavailability of targeted devices.

Malware on field devices. Many SCADA field devices were designed without security considerations, and potential vulnerabilities allowing remote code execution on RTUs have been documented officially (e.g., CVE-2017-12738, CVE-2018-10605). Therefore, an attacker would gain remote access to field devices and modify their software to launch an attack against the critical infrastructure. Malicious changes on the controlled process can possibly be observed by an operator from abnormal number of alarms, values of measurements, amount of traffic, etc. In order to hide the malicious activity, the malware usually suppresses the real outbound packets and sends altered packets to the SCADA master. Stuxnet[5] and Irongate[6] capture normal outbound values and replay it to mask anomalies produced while launching attack to the controlled process. In this work we propose a more stealthy scenario wherein the replayed traffic not only contains the historical measurements but also follows historical timings to send. With regards to the anomalies, the value of the payload, size of messages, and timings are totally legitimate. However, SCADA traffic usually contains phase transitions or seasonal changes. This makes it detectable (after a period of time) by comparing the similarities between field devices. A slow detection is acceptable because this kind of malware is usually a part of an APT, which is designed to be effective for an extended period of time.

The two proposed attack scenarios will be used to generate relevant test cases for evaluation in Section 7.

[5] https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

[6] https://www.fireeye.com/blog/threat-research/2016/06/irongate_ics_malware.html

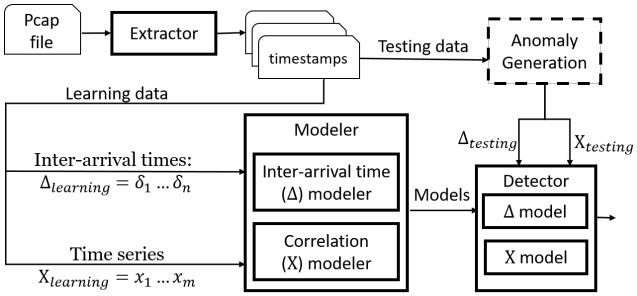


Figure 2: System components and workflow

5 Proposed Anomaly Detection System

The proposed system contains three main modules as illustrated in Figure 2. First, an *Extractor* software that can be run in a operation site, e.g., a utility company, for collecting the input traces to the anomaly detection process. The *Extractor* records the timestamps of packets containing spontaneous events (COT=spont) and separates the timestamps in flows which are defined by a three-tuple {RTU, TID, IOA}. The timestamp sequences in each flow are then used as learning data and later sequences collected in a similar way can be used as testing data.

Second, for each flow from the first step the learning data is transformed into two data types as the inputs for the *Modeler* to build models. For the *inter-arrival time model*, we form a sequence of inter-arrival times $\Delta_{learning}$, and denote each inter-arrival time appearing in the sequence by δ_i. For the *correlation model*, we form a time series $X_{learning}$, and x_i denotes the number of spontaneous events in i^{th} bin. The bin size is a configurable parameter, and will be 1 minute in this work.

Third, the models are sent to the *Detector*. The testing data in the same format as the learning data will be transformed into $\Delta_{testing}$ and $X_{testing}$ and used to test our *Detector*.

In our evaluation of the method, as we are creating synthetic attacks to test the detector, we also have a separate Anomaly Generation function (dashed box) which will inject suitable anomalies in the collected test data to create test sets which are a combination of real utility data and attack-induced variations of them.

The *Extractor* is written in Python and the other modules are developed and run in R.

5.1 Modeling Spontaneous Events

This section presents how we construct the *inter-arrival time model* and the *correlation model*.

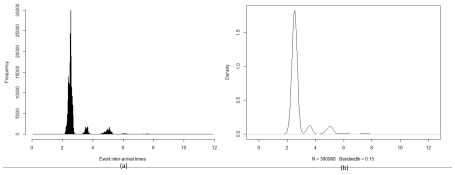


Figure 3: Distribution of inter-arrival times from a sequence within an event set in our data: (**a**) Histogram of $\delta_i \leq 12$ seconds. (**b**) The smoothed version of the sequence, bandwidth=0.15.

5.1.1 Inter-arrival time model

The process of building the *inter-arrival time model* contains three steps: smoothing, grouping, and finding boundaries. The first two steps are developed in an earlier work [17] with modifications on choice of parameters (bandwidth, size of *GroupList*). This section includes these two steps for the sake of completeness.

Smoothing. This step uses kernel density function to smooth out the discontinuities in the distribution of $\Delta_{learning}$. The smoothed distribution of inter-arrival times is called $\Delta_{smoothed}$. Figure 3 shows part of the frequency distribution of inter-arrival times which is less than 12 seconds and its corresponding smoothing results. The bandwidth parameter of the kernel decides the smoothness level of $\Delta_{smoothed}$. The parameter can be an arbitrary large number as long as the peaks are still higher than the bottoms in the resulting $\Delta_{smoothed}$.

Grouping. The next step starts with finding the relative low points in $\Delta_{smoothed}$ with Algorithm 1. The algorithm outputs a *GroupList* which contains a list of low points in pairs. The inter-arrival times within a pair of low points are considered as in the same group.

Finding boundaries. The last step finds the best fitting boundaries of each group. These boundaries are used for detection later. There are two methodologies used for finding the best fitting boundaries. (Section 6 compares the two methodologies.) (1) The best-fitting boundaries can be set as simply percentile of 99.99% and 0.01% of observations in each group without any assumptions on their distribution. (2) With a Gaussian distribution assumption, the module calculates the mean and standard deviation of the observations in each group and sets the boundaries as mean plus/minus three standard deviations.

In both of the methodologies, the module needs to estimate the boundaries of groups having too few observations in a specialized way. Without a specific assumption of distribution, the module takes the range between boundaries in the largest group of the same flow and centers the range with the median of the current group. With a Gaussian distribution assumption, the module calculates only the mean and reuses the standard deviations in the largest group of the same flow.

Algorithm 1: Finding low points

Input : $\Delta_{smoothed}$
Output: GroupList

```
GroupList ← empty // list for output
while true do
    peak ← HighestPoint(Δ_smoothed);
    L = R = peak;
    while Δ_smoothed[R+1] < Δ_smoothed[R] do
        R ← R+1;
    end
    while Δ_smoothed[L−1] < Δ_smoothed[L] do
        L ← L−1;
    end
    if (L == R) then
        break;
    end
    GroupList[i] ← (L,R);
    Δ_smoothed ← Δ_smoothed − PointsBetween(L,R);
end
```

The criteria for adopting this specialized estimation needs to be decided experimentally (See Appendix A).

5.1.2 Correlation model

The process of building the *correlation model* contains three steps: correlation pairing, phase separation, and finding boundaries. We aim to find a monotonic relation between the time-series and pair them. The two paired time-series change (increase or decrease) with the same direction but not always at the same rate. As observed in the earlier work [17, 22, 9], SCADA traffic exhibits phases, and the timing characteristics of the traffic may be different in each phase. We assume correlated time-series are in a monotonic relation instead of linear relation because of phase transitions.

Correlation pairing. This step starts with calculating the Spearman rank correlation matrix between event volume time-series of different flows. Spearman correlation measures the strength and direction of two variables belonging to a monotonic relation. For two time-series $X^p = x_1^p, \ldots, x_m^p$ and $X^q = x_1^q, \ldots, x_m^q$, the Spearman rank correlation is [7]:

$$\rho_{pq} = \frac{COV(R(X^p), R(X^q))}{\sigma_{R(X^p)}\sigma_{R(X^q)}} \tag{2}$$

where $R(X^k)$ denotes the ranked time-series $R(x_1^k), \ldots, R(x_m^k)$. In a ranked time-series the numerical value in each bin x_i^k is replaced by their rank in the sorting. In addition, $COV(R(X^p), R(X^q))$ is the covariance of ranked time series and $\sigma_{R(X^p)}$ and $\sigma_{R(X^q)}$ are the standard deviations.

[7] Spearman, C. The Proof and Measurement of Association Between Two Things. American Journal of Psychology (1904).

In the calculated correlation matrix, for any row representing a flow in the system, the column with the highest coefficient thereby gives the most correlated flow and pairs with it. Note that the pairs are not always symmetric. It can be the case that time-series X^p is most correlated with X^q and X^q is most correlated with another time-series.

Phase separation. This step separates the time-series into phases by assuming the relation between two paired time-series X^p and X^q at time point i and phase k follows:

$$x_{ki}^p = x_{ki}^q + \varepsilon_k^{pq},\ \varepsilon_k^{pq} \sim N(\mu_{pqk}, \sigma_{pqk}^2) \tag{3}$$

where ε_k^{pq} is a constant with added noise. ε_k^{pq} is independent from x_{ki}^p and follows Gaussian distribution. That is, the difference between x_{ki}^p and x_{ki}^q should fall within the range of ε_k^{pq}.

The process of separation starts by observation of the histogram of D^{pq}, the arithmetic difference between two paired time-series X^p and X^q. If the histogram is not a bell curve, we consider it as a Gaussian Mixture Model (GMM). Then, we use an Expectation−Maximization (EM) algorithm [8] implemented in the R library to separate the GMM into a few Gaussian models. Each Gaussian model is called a component and represents a phase. This algorithm gives each d_i^{pq} a posterior probability of belonging to one of these components. We can assign d_i^{pq}, x_i^p and x_i^q into the most likely component at every time point i. The set of datapoints in component k is called X_k^p and X_k^q.

Finding boundaries. This step removes the outliers in X_k^p and X_k^q, sets entering conditions for each phase, and finds the estimated boundaries of ε_k^{pq} for detection. Fluctuations in the volume of each bin from the same phase causes outliers in the probability distribution of X_k^p and X_k^q to be generated in the previous step (See Figure 4). Our model excludes the outliers by adjusting entering conditions (cut-off lines) of phase k (e.g., $x_i^p \in X_k'^p$, $k = 1$, if $7 > x_i^p > 3$).

The module calculates the boundaries of ε_k^{pq} with the adjusted results $X_k'^p$ and $X_k'^q$. Let D_k^{pq} as the arithmetic difference between $X_k'^p$ and $X_k'^q$ for each k. The estimated boundaries of ε_k^{pq} is set as the mean plus/minus three standard deviations of D_k^{pq}. The example calculation of boundary settings with the used datasets are illustrated in detail in section 6.

5.2 Detection Mechanism

We use a two-layer approach to generate alarms for unusual network behaviours. In the first layer, the input traffic is preprocessed into two data types $\Delta_{testing}$ for the inter-arrival time model and $X_{testing}$ for the correlation model. The detector generates an inter-arrival time warning if an inter-arrival time falls outside the boundaries of the learned groups and generates a correlation warning if the difference of two paired

[8] NormalmixEM in library "mixtools". https://cran.r-project.org/web/packages/mixtools/mixtools.pdf

Dataset	TID	# Flows	# Used Flows	AEF
RTU A	M_ME_NA	21	19	228780
	M_DP_TB	8	0	13
	M_SP_TB	3	0	2
RTU B	M_ME_NA	16	14	7837

Table 1: Overview of datasets during 30 days. Here AEF stands for Average Number of events per Flow.

time-series falls outside the boundaries of the identified current phase in the pair.

In the second layer, the detector calculates the number of inter-arrival time warnings for each RTU and correlation warnings in every minute to form a multivariate time-series. The system uses sliding windows W_{Δ_RTUi} and W_X on the time-series to filter out noise. The detector generates alarms when the number of warnings exceeds the predefined second-layer thresholds T_{Δ_RTUi} or T_X. Every RTU in the network has its own detection parameters W_{Δ_RTUi} and T_{Δ_RTUi}, and we refer to it as a $RTUi$ component of the inter-arrival time model. As usual, the configurable parameters of a detection mechanism are a means to set a desired level of accuracy or false positive rate, as will be evident in Section 7.3. The alarms will be sent to the operator monitoring the system but the warnings are internal elements of the analysis used to set thresholds.

6 Datasets and Parameter settings

This section presents an overview of the datasets and the parameter settings for model construction. It elaborates which parts of our data are used for the further experiments and how they are used.

The experiments employ network traces collected from a real-world power facility. The data collection was performed by the utility personnel embedding the *Extractor* component in Figure 2 in their facility and providing the event inter-arrival times for spontaneous events for our evaluation. The network uses multiple protocols and we collect only the timings of communications between the SCADA master and 2 IEC-104 specific RTUs. The datasets have a duration of 30 days and we choose learning data to be 300-hour long (i.e., about 12 days). Table 1 presents a brief summary of the datasets we used in our experiments.

Our datasets contain different types of measurements (TID) and flows. The flows with small event quantities (less than 200 events) are outside our scope because these flows, such as M_DP_TB, M_SP_TB and unused M_ME_NA traffic, apparently have very different timing characteristics and should be monitored using alternative methods. Consequently, all of the used 33 flows are from M_ME_NA measurement traffic. Our datasets also show different event rates in each RTU. RTU A has a much higher event rate than RTU B.

6.1 Parameter and design choices for the inter-arrival time model

The top 5 flows with the highest event frequencies from RTU A and the top 3 flows from RTU B are used for buillding the models. This is because the slow flows contain less observations for each group. Therefore, there are more groups in the slow flows requiring specialized estimation for the boundaries as mentioned in Section 5.1.1. We exclude these flows to focus the feasibility study of our approach on the more frequently populated flows.

Table 2 shows the learning results in terms of warning rates for the selected flows. It explains the methodology to find the best-fitting boundaries for detection with limited length of learning data. Warning rate is defined as the number of inter-arrival times which can not be categorized in any group divided by the amount of inter-arrival times in the whole learning data in percentage (%). Warning rate I is calculated directly after the *Grouping step* (the second step of the inter-arrival time model construction). The results show that the initial grouping covers most of the data points by the chosen bandwidth parameter for algorithm 1.

There are two alternatives methodologies to find the best fitting boundaries for detection. Warning II and warning III are calculated after completing the *Finding boundaries step*. In warning II cases we find best fitting boundaries without any specific assumption on their distribution (i.e., the first methodology for finding boundaries in Section 5.1.1). We can observe that this methodology leads to high warning rates for the flows from the low-event-rate RTU (RTU B). In warning rate III cases we find boundaries with Gaussian distribution assumption (i.e., the second methodology). We note that the two methods are suitable in different cases. The warning II method works better when there are sufficient number of observations, and the second method (Gaussian, warning III) performs best with learning data with fewer observations. Therefore, the boundaries estimated with the first methodology are used for RTU A component and the boundaries estimated with the second methodology are used for RTU B. The detailed warning rates for the whole dataset will be presented in Table 4 of Appendix A.

6.2 Parameter and design choices for the correlation model

For the correlation model, we form 33 pairs from the used 33 flows through the correlation pairing process described in Section 5.1.2. 3 out of 33 pairs are a mixture of two distributions and require a phase separation process. These pairs contain phase transitions which is the desired feature as mentioned in Section 4. They are included for further experiments no matter how many warnings they have produced in

Flow	Warning rate I	Warning rate II	Warning rate III
A_3019	≈ 0	0.03	0.46
A_3014	0	0.02	0.59
A_3013	≈ 0	0.02	0.52
A_3012	≈ 0	0.02	0.46
A_3020	≈ 0	0.02	0.49
B_3019	0.10	14.88	0.54
B_3016	0.10	14.86	0.46
B_3006	0.08	0.22	1.48

Table 2: Overview of selected datasets and learning results. Warning rates are presented in %.

the learning period. 20 out of the remaining 30 pairs have warning rates below 1 % for the learning data and are included for further experiments as well. It is apparent that the pairs having high warning rates are not tightly correlated. These flows are thus useless to include in our correlation-based learning. Table 4 of Appendix A lists all the pairs and their warning rates for the learning data.

Figure 4 (a) presents an example of two paired time-series containing phases. The two time-series show clear high and low values with somewhat visible correlation. Figure 4 (c) is the histogram of difference between the two time-series and indicates it may be a mixture model of two components. The EM-algorithm results suggest that it is a mixture of component 1 $\sim N(-6.25, 8.20)$ and component 2 $\sim N(-0.31, 1.08)$, where $\sim N(\mu, \sigma)$ denotes a Gaussian distribution with mean μ and standard deviation σ. The proportion of these two components are 0.62 and 0.38 respectively. We separate the original time-series into two groups according to the posterior probabilities. Figure 4 (d) is the histogram of RTUA_3018 dataset that are in the component 2 (high-value phase). Since we model the distribution with two components, the boundary of the high-value phase is simply set at one-side as presented by the red line at 18. Figure 4 (e) is the histogram of the difference between RTUA_3018 and RTUB_3015 when the value of RTUA_3018 is higher than 18. The learned boundaries of differences for this pair in the high-value phase are -7 and 10 and the boundaries in the low-value phase are -27 and 6.

This example shows that the replay attacks may break the correlation between some paired time-series if one flow replays the historical data but another one has changed to another phase. It indicates our hypothesis for the approach being suitable for detecting replay attacks is likely to work.

7 Evaluation

We evaluate the proposed anomaly detector in two steps. First we study whether the testing data (with no attacks injected) exhibits signs of pre-existing anomalies. Then we create synthetic anomalies according to the threat models presented in Section 4 and show the performance of the detector against the synthetic anomalies. The performance is evaluated with detection accuracy and time-to-detection.

7.1 Warnings and Actual Anomalies

Figure 5 shows the warnings generated by each model on the collected data from the utility for testing over time. The inter-arrival time model generates warnings at a relatively stable rate compared to the correlation model. The correlation model shows an anomalous burst of warnings around 40000 minutes and it lasts for thousand minutes till the end of the data. These anomalies are generated mostly from a correlation pair as shown in figure 6. As a result of discussions with the utility company we found that the correlation break is caused due to a manual intervention by the operator at the company, where a line breaker was used to turn off the current on a line at 8:53, perform a redirection, including connecting the ground, and restarted the transmission on the original line at 9:44. The normally correlated line was not touched, so the anomaly detected was grounded in reality. The detailed warning rates for each flow can be found in Table 5 of Appendix B.

Due to the existence of actual anomalies, we exclude the data after 39750 minutes for the following evaluation of the anomaly detector in the step where injected anomalies are inserted in the dataset. The authors have noticed that there are still some short bursts of warnings in the used segment of the dataset and these bursts may be caused by manual operations on the system or traffic noise. Since the log of operations is not available for our study, we decide to treat them as normality.

7.2 Synthetic Anomaly Generation

We create two types of synthetic anomalies according to the threat models presented in Section 4. We refer to the anomalies caused by the attack against field devices as performance downgrade anomalies and the one caused by the malware on field devices as replay anomalies. For each type of synthetic anomaly, we conduct several experiments on each RTU with different severity/scale. An experiment will be repeated 25 times with randomly generated anomalies as explained below.

7.2.1 Performance Downgrade Anomalies

In this scenario we emulate cases where attacks lead to performance downgrade. This is done through an anomaly generator that adds delays to and drops packets from the RTU traffic. The anomaly generator is implemented to emulate the impacts of such attacks with a queuing model as $g(L, \mu, \phi, \psi)$, where g represents the queue model with queue length L, μ is the mean service rate (events/second),

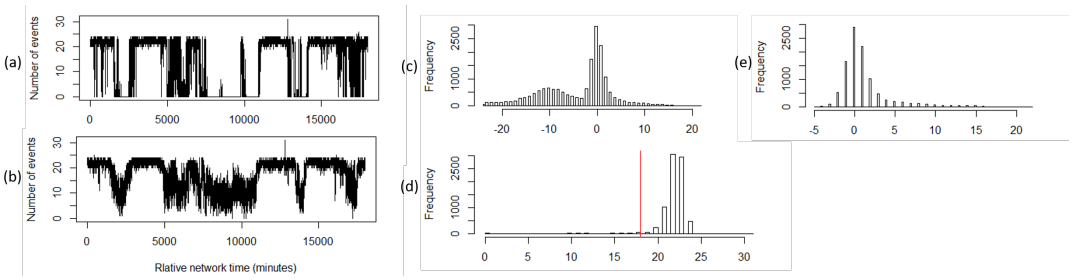


Figure 4: An example pair of time-series. (**a**) Time-series from flow {RTUA, M_ME_NA, 3018}, called RTUA_3018. (**b**) Time-series from flow {RTUA, M_ME_NA, 3015}, called RTUA_3015. (**c**) The histogram of difference between RTUA_3018 and RTUA_3015. (**d**) The result of running EM-algorithm: the histogram of RTUA_3018 dataset that are in the high-value phase. (**e**) The histogram of difference between RTUA_3018 and RTUB_3015 when the value of RTUA_3018 is higher than 18.

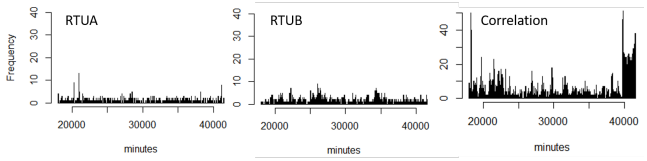


Figure 5: Number of warnings generated by component RTU A of the inter-arrival time model, component RTU B of the inter-arrival time model, and the correlation model in every 24 minutes

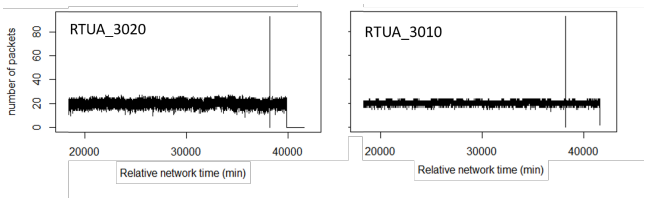


Figure 6: A pair of time-series with correlation breaks.

ϕ is the malicious traffic rate, and ψ is the original traffic randomly selected in our testing data.

To create the queuing model we first abstract the original network packets with their arrival times and number of events (a packet may contain multiple number of events). An event occupies one unit of queue length L. Then, we select a malicious packet rate ϕ which represents the severity of attacks. We can escalate the severity of attacks by increasing ϕ. The original and malicious traffic compete for some shared resources such as CPU time, network card bandwidth, etc. The queue and the service module model the shared resources. The synthetic arrival time of an event is the sum of its original time and waiting time. Service time is only used to model the normal congestion in the queue but not included in the calculation of synthetic arrival times. This is set to prevent generating anomalies when the malicious traffic rate is 0.

There are three configurable parameters in the model, ϕ, μ, L. The parameters for experiments are tuned to match the results of simulation of DoS attacks on real RTUs [12] where an RTU shows little delay when ϕ is 100 packets per second and can barely send traffic at the rate of 580 packets per second. Therefore, μ is set as 82.64 to create congestion in the queue when ϕ is 100. We assume the queue becomes full of malicious packets after a period of time $t = 180$ seconds (corresponding to the attack rate $\phi = 580$). The length of the queue is thus set as $L = (\phi - \mu) * t \cong 89524$.

We conduct an experiment on each RTU for ϕ equals to 100 and 200 respectively. This leads to four experiments. The experiments with ϕ equals to 100 are intended to model the case where resource claims only create delays. They will be used to show the effectiveness of delay detection. The experiments with ϕ equals to 200 model a case where the resource exhaustion creates both delays and packet loss. They will be used to show the timing efficiency of the delay detection (i.e., the detector can find delays before packet losses.).

The total data set was used as follows: The 18 days remaining after removing the 12 days learning time was further reduced to remove the 3 days that included the "benign" anomalies as described in Section 7.1 that appeared at the end of the period. This leaves us with a 15 day interval as testing data. From these 15 days we extract 40-minute long samples at random as the inputs (ψ) of the anomaly generator and then replace the 40-minute long samples in the testing data with the resulting traffic of the queuing model.

7.2.2 Replay Anomalies

Next, we create a threat scenario that the malware repeats certain daily activities and gradually makes damages on the controlled process. It records a 20-second[9] traffic sequence before it starts the malicious activity and replaces the out-

[9] Stuxnet records process values for 21 seconds. Irongate records 5 seconds.

bound traffic with forged traffic while running the activities. The forged traffic replays the values in the recorded and follows the recorded inter-arrival times. We expect that an evasive attacker's forged traffic can bypass the monitor of inter-arrival time component since it replays the regular timings. However, the traffic may cause correlation breaks for the correlated flows from different RTUs.

We conduct one experiment for each activity duration equal to 10 hours, 1 hour, 30 minutes and 5 minutes respectively. This will be done once for RTU A and once for RTU B, leading to 8 experiments. From the 15-day testing data we use the first day to extract the 20-second intervals for the replay action at random, and selected the 10 hour, 1 hour, 30 minutes, and 5 minutes intervals for the 8 replay experiments from the remaining 14 days at random.

7.3 Detection accuracy

This section summarizes the resulting detection accuracy for the experiments presented in Section 7.2.1 (performance downgrade anomalies) and 7.2.2 (replay anomalies). The section first discusses the impact of detection parameters using AUC (Area Under the Curve) and ROC (Receiver Operating Characteristics) curve. During the calculation of ROC, True Positive Rate (TPR) is defined as the number of true alarms divided by the number of windows with inserted events, and False Positive Rate (FPR) is defined as the number of false alarms divided by the number of windows without inserted events. Detection rate is defined as the number of detected event insertions divided by the number of inserted events. Our evaluation is based on the assumption that one type of attack is active at any interval of time. Thereby we evaluate reactions to each type of attack separately with a set of experiments.

There are two types of tunable detection parameters: window size (W_{Δ_RTUi}, W_X) and detection threshold (T_{Δ_RTUi}, T_X). Figure 7 presents the detection ability for each model and the corresponding experiments in terms of AUC ranging from 0.5 to 1, where 1 is the best value with highest detection and lowest false positives. We execute our experiments 25 times with a given window size from 10 to 100 minutes (shown on the x axis) and present the median of the 25 AUCs. The results show that the inter-arrival time model has AUCs close to 1 for detection on all types of downgrade anomalies and with all the window sizes. In contrast, the correlation model has a wide range of AUCs from slightly higher than 0.5 to 1, where the duration of malicious activity is varies from 5 to 600 (minutes).

For the correlation model and the replay experiments, there are three groups of results. The leading group includes the experiments on RTU B with activity duration of 600, 60, and 30 minutes and the experiment on RTU A with activity duration of 600 minutes and 60 minutes. The middle group contains only the experiment on RTU B with activity duration of 5 minutes. The remaining experiments are in the bottom group. The leading and middle group show higher AUCs with small window size, and the AUCs decrease as the window sizes grow. The bottom group has AUCs just above 0.5 with small window size. Though the AUCs grow as the window sizes increase, we speculate this is just because some false positives become true positives in an increased window size (i.e., some inserted events are included in the window). Since this would give an overly positive picture, we suggest a small window size for anomaly detection.

Anomaly Type	Scale	Location	Detection Rate	FPR
Performance Downgrade	100	RTU A	100	0.1
	100	RTU B	100	0.5
	200	RTU A	100	0.1
	200	RTU B	100	0.5
Replay	600	RTU A	61.3	1.2
	600	RTU B	69.4	1.2
	60	RTU A	76.5	1.1
	60	RTU B	80.0	1.0
	30	RTU A	61.4	1.1
	30	RTU B	69.3	1.0
	5	RTU A	44.5	1.1
	5	RTU B	52.9	1.0

Table 3: Detection accuracy (%) with FPR around 1 %

Figure 8 shows an example ROC curve of the inter-arrival time model detecting downgrade anomalies on RTU A with a fixed window size 100 (minutes). The detection thresholds decide the cut-off line. The configuration of detection thresholds depends on the acceptable FPR. If considering an acceptable FPR as 1%, which means 1 false alarm per hundred minutes, we can set the detection parameters as $W_{\Delta_RTUA} = 30$, $W_{\Delta_RTUB} = 30$, $W_X = 10$ and $T_{\Delta_RTUA} = 11$, $T_{\Delta_RTUB} = 7$, $T_X = 10$. Table 3 presents the average detection accuracy (%) with these settings.

The inter-arrival time model providing 100% detection rates and false positive rates below 0.5%. The correlation model has detection rates between 61% to 80% for the leading group of experiments. This is satisfactory because these attacks are expected to be hard to detect as mentioned in section 4. On the other hand, the correlation model shows lower detection rates for the remaining experiments which have low AUCs. The detection rates are between 44% to 61%. This implies that the correlation technique applied here would not be an effective technique for finding anomalies that occur in high volume traffic with short period of activities.

7.4 Timing Performance

This section presents the timing performance for the experiments presented in Section 7.2.1 (performance downgrade anomalies) and 7.2.2 (replay anomalies) in TTD (Time To Detection). The TTD measurement is based on time-series bins (minutes). It is defined as the time on which the IDS

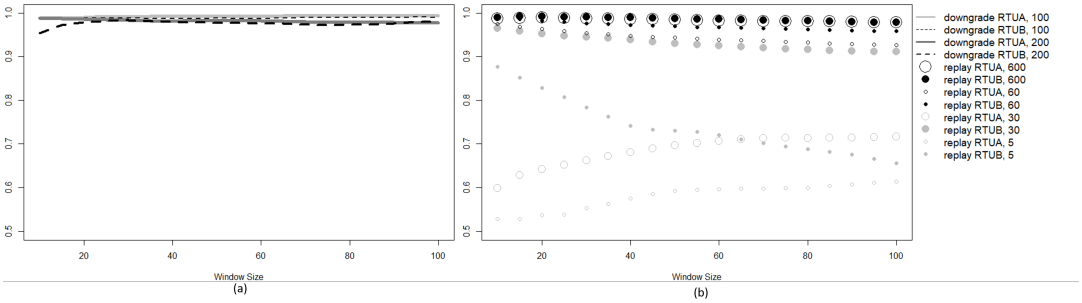


Figure 7: The median of AUCs for: (**a**) performance downgrade experiments, (**b**) replay experiments.

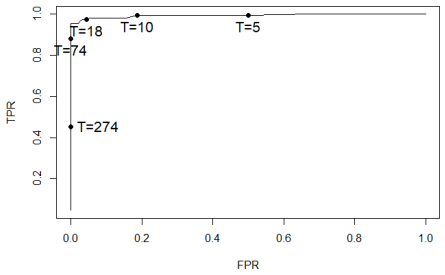


Figure 8: ROC curve of the inter-arrival time model on its first run of experiment performance downgrade RTU A, ϕ=100. T means threshold T_{RTUA}

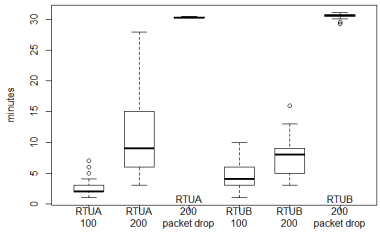


Figure 9: Comparison between time to detection and time to first drop

raises its first alarm minus the time of the first event insertion. The results of the measurements are shown in a boxplot with median, quantile 25% and 75 %.

7.4.1 Performance Downgrade Experiments

The TTD in each downgrade experiment is compared with the time to the first packet drop in the experiment with ϕ=200. If our approach can detect anomalies before the first packet drop, it outperforms the general-purpose network monitoring tools which identifying and monitoring packet loss. As mentioned in Section 7.2.1, we abstract the original network packets with their arrival times and number of events and emulate the performance degradation with a queuing model. The experiments with ϕ equals to 200 model a case where the resource exhaustion creates both delays and packet loss. If the i^{th} packet in the original traffic Pkt^i is the first packet being dropped in the emulation results, the time to first packet drop is calculated as $T(Pkt^{i-1}) + \delta_{i,i-1}$, where $T(Pkt^{i-1})$ denotes the delayed time of Pkt^{i-1} and $\delta_{i,i-1}$ denotes the inter-arrival time between the Pkt^{i-1} and Pkt^i in the original traffic.

Figure 9 presents the timing performance for performance downgrade experiments. The time to the first packet drop measurements are centered around 30 minutes for both of the RTUs. The median of TTD measurements for each experiment from left to right is 2, 9, 4, and 8 minutes. Every iteration of all the experiments has TDD less than 30 minutes. It is worth noting that TTD measurements become longer with higher severity of attack. This is because the attacks with a higher input rate (ϕ) to the queuing model cause more delays between two events such that the number of events and alarms in one minute (bin) decreases.

7.4.2 Replay Experiments

Figure 10 presents the timing performance for replay experiments. In some of the iterations for some experiments there is no detection. Figure 10 (a) presents the TTD measurements for the iterations with detection. The median of TDD for each experiment from left to right is respectively 4, 5.5, 7, 9, 3.5, 4, 4 and 4 minutes. This shows, for most of the cases, our approach detects the anomalies in a short time if they are detectable.

Figure 10 (b) shows the number of iterations without detection for all the replay experiments. The number for each experiment from left to right is 13, 7, 4, 1, 9, 5, 0, and 0. This shows, in a given dataset, the recorded traffic characteristics, the traffic characteristics during the replay, and the length of replay period influence the probability of occurrences of detectable anomalies. Among them, the length of replay period

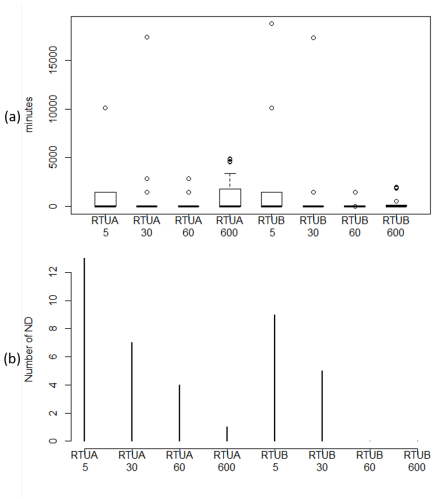


Figure 10: Timing performance for all replay experiments. (**a**) Time to detection measurements. (**b**) Number of iteration without detection, where ND denotes no detection.

plays a decisive role in our experiments. When the length of replay period is short, it could be pretty hard to detect replayed events if both of the recorded traffic and the traffic during replay do not deviate from the learned normality.

7.5 Discussions

This section discusses the proposed approach in terms of its generality and robustness to APT according to the analysis. We then go on to discuss the challenges regarding FPR and lack of semantics for network-based anomaly detection systems and how can them be addressed in practice.

This research inherited the findings of previous work which used a 12-day dataset generated from a virtualized testbed combining a SCADA network formed of virtual machines and a power grid simulator [17]. Though the previous work did not conclude the existence of the used attributes, inter-arrival time and correlation attributes, we can infer from its observations that these attributes may exist as shown in the section 3.2. Considering the results of empirical study in this work, we can expect the used attributes to appear in other power grid network traffic as well, but more work on new sites would be useful before generalizing the result.

Our method was shown promising for flows that have a pattern which is possible to model. It is still an open question whether smart grids with frequent re-configurations in the face of market changes or demand-response imbalances may exhibit new patterns of timing for spontaneous events.

The proposed approach is a composition of two components. The inter-arrival time component examines the pcap timestamp of the packets, which means the time when the packet arrives at the data collection workstation, instead of the time tag in the payload. In practice, this kind of IDS is usually located in the kernel of the workstation and extracts the arrival time directly from the network driver. This information can not be simply altered for evasion or poisoning as long as the adversaries do not take control of the network driver. When the network driver has been compromised, it's too late for any network-based IDS. However, the adversaries may take control of a field device and send forged data that mimic the regular timings as shown in section 7.2.2. This kind of evasion can bypass our inter-arrival time component but it is still detectable by the correlation component as shown in Table 3. This is because the correlated traffic flows are located across the RTUs in the network. The only way to bypass the monitoring of the correlation component is to take control of all the monitored field devices and send the forged data simultaneously. Hence, our approach fits the distributed nature of SCADA systems.

Two main challenges for network-based anomaly detection are avoidance of high false positive rates and reaction to alarms lacking semantics. Our approach reported around 1% FPR, which means 1 alarm per 100 minutes in the configured setting and less than 15 alarms a day. The evaluation on whether the FPR is manageable depends on the settings of the facilities. Solely receiving the alarm with no connection to the system information puts the burden on the operator for further investigations. However, a growing number of critical infrastructure facilities are supported by a security operation center housing security expertise as well as domain knowledge about the operational setting. Combining capabilities of different technologies such as digital forensics, an anomaly detector with 15 alarms a day can be helpful.

8 Conclusions and Future Work

This work explores the potential of modeling IEC-104 spontaneous events using two timing attributes. One is based on the inter-arrival times, and the other is based on the correlation between flows. These two attributes have been shown very stable in the whole experiment period of this work on a dataset collected from a real power station. Our experiments generate consistent false alarm rates in the learning and testing period if the learning period contains enough number of observations.

We also propose methods for modeling two attack scenarios at the field device level and analyze their impact on timings. The performance downgrade anomalies are caused by attacks against a field device and their impact is persistent. The replay anomalies are caused by malware on a field device and the impact is intermittent.

The proposed anomaly detector successfully detects real anomalies in the dataset from a real power facility as well as some of the synthetic anomalies. Our approach shows different level of detection accuracy for different type of anoma-

lies. For the performance downgrades (persistent) anomalies, our approach exhibits a 100% detection rate with at most 0.5% false positive rate. For the replay (intermittent) anomalies, the detection rates and timing performances are satisfactory for experiments under the following conditions: (1) the anomalies last for a longer period (over 1 hour), or (2) the original traffic has relatively low event rates.

The preliminary results show that the idea of modeling the timing characteristics of spontaneous events for anomaly detection is fruitful. The spontaneous traffic exhibits persistent timing characteristics with regards to its inter-arrival times and correlation between flows. This study can be used as a foundation of future research on anomaly detection in spontaneous traffic and our dataset is available for further trials. The obvious future work includes studying a more advanced machine learning model to enhance the detection ability of replayed events. The fact that the timing performance of the inter-arrival time model decreases when the severity of attacks increases also indicates a potential need of a more complicated model to monitor the change to alarm distributions, such as a model using range value or moving average.

To our knowledge this is the first study of anomaly detection focusing on IEC-104 spontaneous traffic. We do not compare our approach with methodologies designed for other types of traffic. Nevertheless, we plan to generate more attack scenarios and study their impact on timing in the context of a national emulated testbed for SCADA systems (RICS-el). Emulated testbeds allow comparison of multiple methods in the same repeatable scenario, but a functioning shared research infrastructure for this purpose is still missing in the research community.

Acknowledgments

This research is supported by the Swedish Civil Contingencies Agency (MSB) through the RICS (www.rics.se) project. The authors would also like to thank the industrial partners who participated in the data collection as well as the anonymous reviewers who helped improving our paper.

Availability

The used dataset, which is formed of CSV files containing timings extracted from a real power station, is available via the following URL.

https://gitlab.liu.se/ida-rtslab/public-code/2019_iec-104_ad

References

[1] ALMGREN, M., ANDERSSON, P., BJÖRKMAN, G., EKSTEDT, M., HALLBERG, J., NADJM-TEHRANI, S., AND WESTRING, E. RICS-el: Building a national testbed for research and training on SCADA security. In *2018 Critical Information Infrastructures Security (CRITIS)* (2019), LNCS, Springer.

[2] AOUDI, W., ITURBE, M., AND ALMGREN, M. Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In *Proc. of the Conference on Computer and Communications Security (CCS)* (2018), ACM.

[3] BARBOSA, R. R. R., SADRE, R., AND PRAS, A. Exploiting traffic periodicity in industrial control networks. *Internation Journal of Critical Infrastructure Protection 13*, C (June 2016).

[4] CÁRDENAS, A. A., AMIN, S., AND SASTRY, S. Research challenges for the security of control systems. In *Proc. of the 3rd conference on Hot topics in security (HotSec)* (2008), ACM.

[5] CASELLI, M., ZAMBON, E., AND KARGL, F. Sequence-aware intrusion detection in industrial control systems. In *1st Workshop on Cyber-Physical System Security (CPSS)* (2015), ACM.

[6] DÜSSEL, P., GEHL, C., LASKOV, P., BUSSER, J.-U., STÖRMANN, C., AND KÄSTNER, J. Cyber-critical infrastructure protection using real-time payload-based anomaly detection. In *2009 Critical Information Infrastructures Security (CRITIS)* (2010), LNCS, Springer.

[7] FAISAL, M., CARDENAS, A. A., AND WOOL, A. Modeling modbus TCP for intrusion detection. In *2016 Conference on Communications and Network Security (CNS)* (2016), IEEE.

[8] GIRALDO, J., URBINA, D., CARDENAS, A., VALENTE, J., FAISAL, M., RUTHS, J., TIPPENHAUER, N. O., SANDBERG, H., AND CANDELL, R. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys 51*, 4 (September 2018).

[9] GOLDENBERG, N., AND WOOL, A. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection 6*, 2 (June 2013).

[10] HADŽIOSMANOVIĆ, D., SIMIONATO, L., BOLZONI, D., ZAMBON, E., , AND ETALLE, S. N-gram against the machine: On the feasibility of the n-gram network analysis for binary protocols. In *Research in Attacks, Intrusions, and Defenses (RAID)* (2012), vol. 7462 of *RAID 2012*, LNCS, Springer.

[11] HADŽIOSMANOVIĆ, D., SOMMER, R., ZAMBON, E., AND H.HARTEL, P. Through the eye of the plc: Semantic security monitoring for industrial processes. In *Proc. of the 30th Annual Computer Security Applications Conference (ACSAC)* (2014).

[12] KALLURI, R., MAHENDRA, L., KUMAR, R. S., AND PRASAD, G. G. Simulation and impact analysis of denial-of-service attacks on power SCADA. In *2016 National Power Systems Conference (NPSC)* (2016), IEEE.

[13] KISS, I., GENGE, B., AND HALLER, P. A clustering-based approach to detect cyber attacks in process control systems. In *13th International Conference on Industrial Informatics (INDIN)* (2015), IEEE.

[14] KLEINMANN, A., AND WOOL, A. Automatic construction of statechart-based anomaly detection models for multi-threaded SCADA via spectral analysis. In *2nd Workshop on Cyber-Physical Systems Security and Privacy (CPSS)* (2016), ACM.

[15] KLEINMANN, A., AND WOOLKW, A. Accurate modeling of the Siemens S7 SCADA protocol for intrusion detection and digital forensic. *The Journal of Digital Forensics, Security and Law 9*, 2 (2014).

[16] KROTOFIL, M., LARSEN, J., AND GOLLMANN, D. The process matters: Ensuring data veracity in cyber-physical systems. In *Proc. of the 10th Symposium on Information, Computer and Communications Security (AsiaCCS)* (2015), ACM.

[17] LIN, C.-Y., AND NADJM-TEHRANI, S. Understanding IEC-60870-5-104 traffic patterns in SCADA networks. In *Proc. of the 4th Cyber-Physical System Security Workshop (CPSS).* (2018), ACM.

[18] LIN, C.-Y., NADJM-TEHRANI, S., AND ASPLUND, M. Timing-based anomaly detection in SCADA networks. In *2017 Critical Information Infrastructures Security (CRITIS)* (2018), LNCS, Springer.

[19] LIU, Y., NING, P., AND REITER, M. K. False data injection attacks against state estimation in electric power grids. In *Proc. of the 16th conference on Computer and Communications Security (CCS)* (2009), ACM.

[20] LONG, M., WU, C.-H., AND HUNG, J. Y. Denial of service attacks on network-based control systems: Impact and mitigation. *IEEE Transactions on Industrial Informatics 1*, 2 (MAY 2005 2005).

[21] MARKMAN, C., WOOL, A., AND CARDENAS, A. A. A new burst-DFA model for SCADA anomaly detection. In *Proc. of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC)* (2017), ACM.

[22] MARKMAN, C., WOOL, A., AND CARDENAS, A. A. Temporal phase shifts in SCADA networks. In *Proc. of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC)* (2018), ACM.

[23] MARKOVIC-PETROVIC, J. D., AND STOJANOVIC, M. D. Analysis of SCADA system vulnerabilities to DDoS attacks. In *2013 11th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS)* (2014), IEEE.

[24] NIEDERMAIER, M., MALCHOW, J.-O., FISCHER, F., MARZIN, D., MERLI, D., ROTH, V., AND VON BODISCO, A. You snooze, you lose: Measuring PLC cycle times under attacks. In *12th Workshop on Offensive Technologies (WOOT)* (2018), USENIX Association.

[25] SAYEGH, N., ELHAJJ, I. H., KAYSSI, A., AND CHEHAB, A. SCADA intrusion detection system based on temporal behavior of frequent patterns. In *17th Mediterranean Electrotechnical Conference (MELECON)* (2014), IEEE.

[26] SHOUKRY, Y., MARTIN, P., YONA, Y., DIGGAVI, S., AND SRIVASTAVA, M. Pycra: Physical challenge-response authentication for active sensors under spoofing attacks. In *Proc. of the 22nd Conference on Computer and Communications Security (CCS)* (2015), ACM.

[27] VALDES, A., AND CHEUNG, S. Communication pattern anomaly detection in process control systems. In *Conference on Technologies for Homeland Security (HST)* (2009), IEEE.

[28] WEDGBURY, A., AND JONES, K. Automated asset discovery in industrial control systems - exploring the problem. In *Proc. of the 3rd International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR)* (2015), ACM.

[29] WRESSNEGGER, C., KELLNER, A., AND RIECK, K. Zoe: Content-based anomaly detection for industrial control systems. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2018), IEEE.

[30] YANG, Y., MCLAUGHLIN, K., SEZER, S., YUAN, Y., AND HUANG, W. Stateful intrusion detection for IEC-60870-5-104 SCADA security. In *PES General Meeting (PES GM)* (2014), IEEE.

[31] YOON, M.-K., AND CIOCARLIE, G. Communication pattern monitoring: Improving the utility of anomaly detection for industrial control systems. In *Workshop on Security of Emerging Networking Technologies (SENT)* (2014), Internet Society.

Appendix

This appendix section lists additional measurement results for the proposed modeling approach. The notion of warning means the first-level anomalies generated by the proposed detection methodology as described in 5.2. They form the basis of further analysis to derive the alarms that will be sent to the operators.

A warning rates over the learning period

Table 4 shows the overview of the datasets and the warning rates generated by the inter-arrival time model and the correlation model for the learning data. For the inter-arrival time model, warning rate I is calculated directly after the *Grouping step* (the second step of the inter-arrival time model construction). Warning II is calculated after completing the *Finding boundaries step*. In these cases we find best fitting boundaries without any specific assumption on their distribution. Warning rate III is calculated after completing the *Finding boundaries step* with Gaussian distribution assumption. For the correlation model, warning rate (Corr.) is calculated after finishing all steps of (correlation) modeling and used to select useful pairs.

The flows are listed in a descending order of its size (number of events). With this specific dataset, we choose to adopt the specialized estimation of inter-arrival time group boundaries when the number of observations is less 50 in the first methodology (percentile) and when the number of observations is less 5 in the second methodology (Gaussian).

It is worth noticing that the inter-arrival time model produces high amount of (false) warnings in either methodologies when the number of events is too small (less than 8330). The specialized estimation of boundaries for groups with too few observations is no longer effective when most of the groups in the same flow are too small.

B warning rates over the testing period

Table 5 presents the warning rates generated by the inter-arrival time model and the correlation model for the testing period from the original traffic. Corr. Warning Rate I is calculated in the original testing period till 41591 minutes. Corr. Warning Rate II is calculated in the testing period before 39500 minutes. Δ Warning Rate is calculated in the original testing period as well. Most of the flows have warning rates that are consistent with the results in Table 4. This shows the stability of the used timing characteristics.

Table 4: Overview of datasets and learning results.

RTU	IOA	# Events	Used in Δ	Warning Rate I	Warning Rate II	Warning Rate III	Paired IOA	Used in Corr.	Warning Rate (Corr.)
RTU A	3019	381038	Y	0.0003	0.026	0.46	A3010	Y	0.422
	3014	347000	Y	0	0.020	0.59	A3013	N	2.578
	3013	346959	Y	0.0003	0.022	0.52	A3014	N	2.578
	3012	341235	Y	0.0003	0.023	0.46	A3013	N	2.439
	3020	328329	Y	0.0003	0.021	0.49	A3010	Y	0.003
	3011	311823	N	0.0003	0.025	0.52	A3015	N	1.761
	3015	311087	N	0	0.024	0.55	A3011	N	1.761
	3017	252486	N	0.0003	0.025	0.41	A3016	Y	0.211
	3018	243580	N	0.0057	0.026	0.85	A3015	Y	1.772
	3010	204059	N	0.0044	0.026	0.28	A3019	Y	0.422
	3021	202539	N	0	0.027	0.50	B3006	Y	0.156
	3016	198924	N	0	0.025	0.40	A3017	Y	0.211
	3005	192750	N	0.0026	0.025	0.54	A3002	N	1.728
	3004	128184	N	0.0062	0.033	0.38	A3009	Y	0.350
	3007	128166	N	0	0.020	0.25	A3010	Y	0.617
	3002	122920	N	0.0016	0.024	0.57	A3005	N	1.728
	3008	111891	N	0	0.017	0.23	A3003	Y	0.594
	3003	111562	N	0.0009	0.024	0.25	A3008	Y	0.594
	3009	82339	N	0.0073	0.035	0.38	A3004	Y	0.350
RTU B	3019	16448	Y	0.10	14.88	0.54	B3016	Y	0.850
	3016	16276	Y	0.10	14.86	0.46	B3019	Y	0.850
	3006	15706	Y	0.08	0.22	1.48	B3019	N	1.144
	3009	8330	N	0.16	0.16	0.23	B3002	N	1.622
	3002	8278	N	0.13	1.96	4.42	B3009	N	1.622
	3004	6421	N	0.20	2.85	6.17	B3011	Y	0.067
	3012	6011	N	0.13	2.23	7.32	B3005	Y	0.228
	3011	5858	N	0.15	2.36	8.04	B3005	Y	0.356
	3018	5657	N	0.16	11.47	0.69	B3009	Y	0.156
	3014	4936	N	0.12	2.90	9.70	B3013	Y	0.111
	3008	4923	N	0.08	3.64	10.97	B3014	Y	0.172
	3005	4157	N	0.07	7.22	11.47	B3011	Y	0.356
	3013	4138	N	0.05	4.18	13.03	B3014	Y	0.111
	3015	2576	N	0.70	1.86	5.09	A3021	Y	0.033

Table 5: The plain warning rates for the inter-arrival time model and the correlation model in the test phase.

RTU	IOA	Used in Δ	Δ Warning Rate	Paired IOA	Used in Corr.	Corr. Warning Rate I	Corr. Warning Rate II
RTU A	3019	Y	0.002	A3010	Y	0.725	0.363
	3014	Y	0.012	A3013	N	4.849	4.910
	3013	Y	0.029	A3014	N	4.849	4.910
	3012	Y	0.043	A3013	N	4.565	4.519
	3020	Y	0.001	A3010	Y	7.503	0.345
	3011	N	0.025	A3015	N	0.886	0.924
	3015	N	0.006	A3011	N	0.886	0.924
	3017	N	0.007	A3016	Y	0.682	0.657
	3018	N	0.154	A3015	Y	0.026	3.434
	3010	N	0.003	A3019	Y	0.752	0.363
	3021	N	0.001	B3006	Y	0.144	0.156
	3016	N	0.000	A3017	Y	0.682	0.657
	3005	N	0.015	A3002	N	3.090	3.117
	3004	N	0.021	A3009	Y	0.271	0.262
	3007	N	0.003	A3010	Y	0.674	0.708
	3002	N	0.009	A3005	N	3.090	3.117
	3008	N	0.004	A3003	Y	0.606	0.625
	3003	N	0.002	A3008	Y	0.606	0.625
	3009	N	0.027	A3004	Y	0.271	0.262
RTU B	3019	Y	0.682	B3016	Y	1.441	1.324
	3016	Y	0.597	B3019	Y	1.441	1.324
	3006	Y	1.252	B3019	N	2.039	1.779
	3009	N	1.816	B3002	N	1.246	1.256
	3002	N	7.473	B3009	N	1.246	1.256
	3004	N	11.398	B3011	Y	0.013	0.014
	3012	N	11.909	B3005	Y	0.322	0.271
	3011	N	12.716	B3005	Y	0.568	0.520
	3018	N	5.463	B3009	Y	0.102	0.106
	3014	N	16.947	B3013	Y	0.424	0.400
	3008	N	16.188	B3014	Y	0.411	0.400
	3005	N	23.503	B3011	Y	0.568	0.520
	3013	N	22.965	B3014	Y	0.424	0.400
	3015	N	21.467	A3021	Y	0.008	0.009