Towards Periodicity Based Anomaly Detection in SCADA Networks

Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras
Design and Analysis of Communications Systems (DACS)
University of Twente
Enschede, The Netherlands
Email: {r.barbosa, r.sadre, a.pras}@utwente.nl

Abstract

Supervisory Control and Data Acquisition (SCADA) networks are commonly deployed to aid the operation of large industrial facilities. The polling mechanism used to retrieve data from field devices causes the data transmission to be highly periodic. In this paper, we propose an approach that exploits traffic periodicity to detect traffic anomalies, which represent potential intrusion attempts. We present a proof of concept to show the feasibility of our approach.

1. Introduction

Supervisory Control And Data Acquisition (SCADA) networks are commonly deployed to aid the operation of large industrial infrastructures, including some considered essential for our society, such as water treatment and power generation facilities. Given their critical nature, security plays a very important role, as the impact of attacks could be catastrophic.

In this context, the use of Intrusion Detection Systems (IDS) is of paramount importance to track down malicious activities. While misuse (or signature) based approaches are essential to deal with known threats, anomaly based methods are necessary to identify novel attacks. Anomaly detection methods characterize the “normal” behavior of the network and identify deviations, that is, anomalies. Anomaly detection systems have been proposed for different aspects of SCADA systems, from communication protocol level [3] to process level log mining [5].

SCADA networks are deployed to monitor and control devices in the factory floor. To accomplish this goal, data is continuously retrieved from these devices, so that a real time view of the infrastructure’s processes can be established. Typically, data is retrieved through an automated polling process, in which requests are sent to field devices every predetermined interval. A side effect of this behavior is that traffic patterns tend to be highly periodic.

Previous work [1] shows that SCADA traffic is indeed highly periodic, and in this paper we propose an approach that leverages this observation to perform anomaly detec-

tion. Our goal is to protect the network services that are accessed in a periodic fashion, by reporting variations in this behavior. Note that although, many attacks can disrupt the traffic periodicity, changes in the traffic periodicity are not necessarily malicious. In this work, we focus on find such disruptions, however identifying their cause is out of the scope of this paper.

The remainder of this paper is organized as follows. In Section 2, we explain in more details how periodicity manifests in the network traffic and discuss the effects of attacks on the periodic behavior. In Section 3, we discuss our anomaly detection approach. A proof of concept is described in Section 4. Finally, in Section 5 we present our conclusions and propose future work.

2. Periodicity and Attacks

2.1 Network Traffic Periodicity

SCADA networks typically exhibit network connections with *periodic bursts* of packets, that is, a fixed number of packets being transmitted at fixed intervals. These bursts are formed by the periodic requests for data sent by clients and by the replies sent back by servers. Network traffic with periodic behavior has two important characteristics that determine its normal appearance: the period (or frequency) and size (i.e., number of packets) of the periodic bursts it contains. It is important to note here that some non-periodic activity, or *noise*, is expected when observing periodic traffic. Noise can be caused by various factors, such as network delay, packet loss and retransmission, protocol specific exchanges (e.g., TCP’s 3-way handshake), etc.

Not all network connections in a SCADA network necessarily show periodic characteristics. For example, if a PLC is commonly accessed manually, non-periodic behavior can be expected. In this paper, we are interested in network traffic that, in its normal behavior, consists only of periodic bursts of packets.

2.2 The Effects of Attacks

Our assumption is that many intrusion attempts disturb the traffic periodicity. To motivate this assumption, we

describe attacks from some of the different categories proposed in [6] and discuss how they would impact the traffic periodicity. Our examples are taken from an open access list of real-world SCADA attacks signatures used in the Quickdraw Intrusion Detection System [4]. The list includes signatures to protect Modbus TCP¹ and DNP3², two well-know standards for SCADA communication. For the sake of simplicity, we consider in the following examples that traffic exchanged between client and server is periodic.

Information Gathering attacks may precede other attacks and are an attempt by the attacker to gather as much knowledge as possible of the target system. A typical way to acquire this information is through *scans*, like the *Modbus TCP Points List Scan*. Scans require a large set of possible addresses or ports to be tested and, hence, are usually performed as fast as possible by the attacker. Such scans could be detected as the generated traffic is clearly not periodic. Note that, if the attacker performs the attack in a slow but periodic way, we might still be able to detect it, as it would either be seen as a new frequency or a change in the amplitude of the service’s normal frequency.

Denial of Service attacks prevent a legitimate user to access a service or reduce its performance. For example, the *DNP3 - Unsolicited Response Storm* attempts to overload a DNP3 server by sending a number of unsolicited response packets, normally used to report alarms. In case the attacker sends a large amount in a short time, this attack could be seen as a large spike in the amount of non-periodic traffic. Similarly to a scan (see above), the attack could be performed slowly, however with reduced effectiveness.

Network attacks manipulate the network protocols. For instance, the *Modbus TCP - Clear Counters and Diagnostic Registers* attack uses a single packet with a specific code function to clear counters and diagnostics registers in a SCADA server, in an attempt to avoid detection. We cannot detect most attacks of this type, as they are executed through just a few packets, and therefore do not disturb the traffic periodicity. We could, however, detect some of the effects of such attacks. For example, the *Modbus TCP - Slave Device Busy Exception Code Delay* attack consists in answering every request with the “device is busy” message preventing a reply timeout. While we do not expect a change in traffic periodicity in this case, the typical answer could consist of multiple packets instead of just one, as in the attack, causing a change in the amplitude.

Buffer Overflow attacks try to gain control over a process or crash it by overflowing its buffer. For example, the *Modbus TCP - Illegal Packet Size, Possible DOS* attack sends a single packet with an illegal packet size, exploiting a bug in the implementation of the protocol stack. Again,

Figure 1. Diagram of our approach

our method is not designed to work for attacks consisting of few packets, but in case the attack is successful and the targeted system crashes, the normal traffic patterns will be clearly disrupted.

In summary, many attacks cause one of three changes in the frequency domain: (1) new or missing periodic burst frequency; (2) change in periodic burst size; and (3) increase in the amount of noise.

It is important to stress that our goal is the detection of anomalies, i.e., deviations from the normal periodic behavior of the traffic. Such deviations are not necessarily malicious. For example, a manual access to a PLC for testing purposes would cause a spike in the non-periodic traffic and, consequently, trigger an alarm.

3. Our Approach

In this section we provide a high-level description of our approach. A proof-of-concept implementation is discussed in Section 4.

Our approach for anomaly detection consists of four modules, depicted in Figure 1. In the first module, traffic from the SCADA network is passively monitored in a central point, where it is analyzed. In this module, packets that are not relevant to SCADA processes, such as DNS and DHCP, are filtered.

The second module task consists in creating network flows, i.e., aggregating packets in a meaningful way. In this work, we propose to aggregated traffic using the server-side transport port, as it identifies the network services we intend to protect. Although the server transport port is sufficient to isolate the periodic traffic in the traces analysed here, additional aggregation keys might be needed in other scenarios. For instance, if a service is accessed by two different clients, one using polling mechanism and the other not, it would be necessary to add client address as an aggregation key in order to isolate the periodic behavior. Another alternative is to isolate the *periodic bursts* using application level information. In practice, the best aggregation method is dependent on the applications and protocols in use.

Flows are stored as time series, i.e., for every fixed interval P , the number of packets belonging to a specific flow is stored. We define sampling frequency SF as $SF = 1/P$. The choice of SF is a trade-off between accuracy and performance. The higher the frequency, the

¹<http://www.modbus.org/>

²<http://www.dnp.org/>

more detailed information about the flows is stored, and, in consequence, more data needs to be processed.

Before performing the detection, we must learn the normal behavior of the system. This is the objective of the *periodicity learning* module. In this step we extract the two characteristics discussed in Section 2.1: the period (or frequency) of the periodic bursts and their size. These characteristics can be extracted, for instance, by the *AUTOPERIOD* method proposed in [7]. Our assumption is the *periodic bursts* characteristics of a service do not change over time, so this analysis can be performed offline, and its results validated by network operators.

Once a flow can be *matched* to a frequency fingerprint, i.e., a flow for which the "normal behavior" is known, it is monitored for anomalies. In Section 2.2, we identified different types of anomalies that can be detected. If an anomaly is identified, an alarm is raised. Ideally, alarms should provide sufficient information so that a corrective action can be executed. For instance, if a new periodic burst is identified in a flow, the alarm needs to provide detailed information on the source of this burst. Alarms are also fed back to the periodicity learning module to support adaptive learning if a fingerprint needs to be updated.

4. Proof of Concept

We show the feasibility of our approach by implementing a spectrogram based anomaly detection module, as a proof of concept of the core function of our approach. The *traffic capture* is emulated by using data collected in a water treatment facility, also used in previous works [1, 2]. Only TCP packets to a few commonly utilized ports were considered. The main reason for this choice is to reduce the amount of data that needs to be analyzed, as most of the process is done manually.

Using TCP connections enables us to identify the server ports that clients connect to, observing the 3-way handshake used to establish the connection. The client always initiates the connection, sending a *SYN* packet to the port to which the server is listening. Although there is no fixed direct mapping from ports to network services, it is fair to assume highly used ports at servers represent a service. In practice, this analysis allows us to build a list of services (ports) available at each server and its respective clients, which we use as basis for the *flow creation*.

In our proof of concept, the *periodicity learning* consists in selecting services that present periodicity with intervals above 1 second. From our interview with the operators, we know that applications should not perform polling with a smaller interval. To perform this task we generate a *periodogram* for each of the flows. The periodogram is a power spectral density estimator defined by the squared length of each Fourier coefficient, which can be calculated with a Fast Fourier Transform (FFT). Through manual inspection of the periodograms, we select the ones with high energy frequencies below 1Hz for the anomaly detection phase, and discard the others.

The *anomaly detection* method consists in tracking changes in the periodic behavior. As the periodogram does not provide time localization, it is not suitable for the task. Instead, we use the discrete-time Short-Time Fourier Transform (STFT), i.e., we apply a sliding window to the data and perform a FFT on each slice. A *spectrogram* can be constructed by plotting time in the x-axis, frequency in the y-axis and the squared magnitude of the STFT as a color. The spectrogram provides a visualisation of how the energy content of the different frequency bands vary over time, making it possible to observe: (1) changes in the set of high energy frequency bands, indicating changes in periodic burst intervals; (2) changes in the amount of energy in the same set, indicating changes in the periodic burst size; and (3) increase in the amount of noise, in this way covering all anomalies we are interested in.

The sliding window method has two parameters: window size and step size. The window size determines the number of samples within a window. Its value is a trade-off between frequency and time localization. In a FFT, the number of samples given as input in the time domain is the same as the number of the discernible frequency bands in the output. Note also that the spectrogram only has 1 data point in the time axis per FFT. Therefore the use of large windows causes a good frequency localization (high number of discernible bands), but bad time localization (small number of data points in the time domain).

The step size parameter determines the number of samples the window moves per FFT. By using small steps one can cope with large windows, as it increases the number of data points in the time domain. However, small steps also mean that the time series for two consecutive FFTs are very similar, as they will have a large overlap.

We selected two flows to illustrate the use of spectrograms to visually detect anomalies. An arbitrary 5-hour slice of the trace was used in the analysis. The flows are generated with a 10Hz sampling rate, and packets are aggregated based on server port and direction (to server or from server). Several combinations of window size and step size were tested. Figure 2 shows the time series and spectrogram plots for the selected flows. For the sake of visualization, we set them to 300s and 18s, respectively, in the results presented here. In addition, the frequency axis range is selected to clearly display the anomalies.

Figure 2a shows the time series for all traffic sent to *service 1*. A few peaks are present in the time series. As any other non-periodic activity, peaks in the time domain cause an increase in noise in the frequency domain, as the energy is spread over the spectrum. The effects of these peaks are clearly visible in the spectrogram for this same flow (Figure 2b). Another interesting behavior is the intermittent activity at around 0.95Hz. The spectrogram suggests a periodic burst of packets at this frequency, but not for the whole duration of the sample. We are not certain of the causes of the behavior.

The second row of Figure 2 shows the more well-behaved traffic sent for a different *service 2*. The spectro-

(a) Timeseries - service 1

(b) Spectrogram - service 1

(c) Timeseries - service 2

(d) Spectrogram - service 2

Figure 2. Detecting Anomalies

gram shows two main frequencies at 0.2Hz (and its integer multiples, or harmonics) and 0.5Hz. However, an increase in the noise level can be identified after 15:00. A closer inspection of the data reveals that this increase is due to small variances in the periodic burst interval. While before 15:00 the bursts are constantly 5s apart, after 15:00 the interval slightly varies in the range 4.8–5.1s, an expected variation when considering the fact that small network delays are common.

The results shows that the spectrogram can be used to localize frequency anomalies in the time domain. However, to serve as a deployable anomaly detection system the analysis has to be automated. One of the possibilities we currently explore is to use the *power distance* [7] metric to compare two consecutive bins of a spectrogram. This metric is shown to be useful when comparing the *periodic structure* of two sequences. A large distance between consecutive bins would indicate the presence of anomalies. The main drawback of this approach is that it does not allow the differentiation between the 3 possible anomalies identified in this work. Furthermore, initial tests indicate that the *power distance* might be too sensitive to period variations like the ones shown in Figure 2d.

5. Conclusions and Future Work

We propose an anomaly detection approach based on the observation that SCADA traffic is highly periodic. The implemented proof of concept shows the feasibility of the approach, however further research efforts towards automation of the approach are necessary. The major challenges are finding an optimal traffic aggregation strategy

to characterize the periodic behavior and dealing with normal variations in the periodic bursts period caused by, among others, network delays.

Furthermore, we will validate the approach with more datasets and test its effectiveness in the presence of realistic intrusion attempt scenarios.

References

- [1] R. R. R. Barbosa, R. Sadre, and A. Pras. A First Look into SCADA Network Traffic. In *IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*, volume 17, page 6. Springer, 2012.
- [2] R. R. R. Barbosa, R. Sadre, and A. Pras. Difficulties in Modeling SCADA Traffic: A Comparative Analysis. *Passive and Active Measurement: 13th International Conference, PAM 2012, Vienna, Austria, March 12-14, 2012, Proceedings*, 7192:126, 2012.
- [3] S. Cheung, K. Skinner, B. Dutertre, M. Fong, U. Lindqvist, and A. Valdes. Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA Security Scientific Symposium*, pages 1–12. Citeseer, 2007.
- [4] Digital Bond. Quickdraw SCADA IDS.
- [5] D. Hadosmanović, D. Bolzoni, and P. H. Hartel. A Log Mining Approach for Process Monitoring in SCADA. *International Journal of Information Security*, 11, 2012.
- [6] S. Hansman and R. Hunt. A Taxonomy of Network and Computer Attacks. *Computers & Security*, 24(1):31–43, Feb. 2005.
- [7] M. Vlachos, P. S. Yu, V. Castelli, and C. Meek. Structural Periodic Measures for Time-Series Data. *Data Mining and Knowledge Discovery*, 12(1):1–28, Feb. 2006.