Network forensic analysis of electrical substation automation traffic

Megan Leierzapf, Julian Rrushi

▶ **To cite this version:**

Megan Leierzapf, Julian Rrushi. Network forensic analysis of electrical substation automation traffic. 11th International Conference on Critical Infrastructure Protection (ICCIP), Mar 2017, Arlington, VA, United States. pp.63-78, 10.1007/978-3-319-70395-4_4 . hal-01819141

HAL Id: hal-01819141
https://inria.hal.science/hal-01819141v1

Submitted on 20 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Chapter 4

NETWORK FORENSIC ANALYSIS OF ELECTRICAL SUBSTATION AUTOMATION TRAFFIC

Megan Leierzapf and Julian Rrushi

Abstract The computations and input/output values of intelligent electronic devices that monitor and operate an electrical substation depend strongly on the state of the power system. This chapter presents an approach that correlates the physical parameters of an electrical substation with the network traffic that intelligent electronic devices send over a substation automation network. Normal network traffic in a substation automation network is modeled as a directed, weighted graph, yielding what is referred to as a model graph. Similar graph modeling is performed on unknown network traffic. The research problem of determining whether or not unknown network traffic is normal involves a subgraph isomorphism search algorithm. Normal network packets in unknown network traffic form a graph that is a subgraph of the model graph. In contrast, malware-generated network packets present in unknown network traffic produce a graph that is not a subgraph of the model graph. Time series analysis of network traffic is performed to estimate the weights of the edges in the graphs. This analysis enables the subgraph isomorphism search algorithm to find structural matches with portions of the model graph as well matches with the timing characteristics of normal network traffic. The approach is validated using samples drawn from recent industrial control system malware campaigns.

Keywords: Industrial control systems, malware, digital forensics

1. Introduction

As the recent BlackEnergy and Dragonfly malware campaigns have demonstrated, industrial control systems (ICSs) are continually targeted by malware that spies on or disrupts industrial processes, including those in the electric power grid. Digital forensic investigations of incidents involving industrial control systems are, therefore, of considerable importance [1, 8]. This chapter

focuses on investigations of intrusions in intelligent electronic devices (IEDs), which are control devices deployed in electrical substations.

The principal contribution is a network forensic approach that leverages graph theory and time series analysis to identify malicious packets in substation automation network traffic. A model graph is used to comprehensively model intelligent electronic devices, their monitoring and protection functions, state of the electrical substation and network traffic that the intelligent electronic devices exchange with each other while performing their functions. The time series analysis helps compute the weights of the edges in the model graph, which characterize the relationships of network traffic with time. The same graph modeling approach is employed to construct graphs that model real network traffic in a substation automation network. Custom Python scripts are used to analyze packet capture (`pcap`) files and help construct adjacency matrices – specifically, concrete graphs – based on captured network packets.

Concrete graphs developed from normal network traffic are subgraphs of the model graph, assuming, of course, that the vertices, edges and weights in all the graphs are computed accurately. Experiments with emulated exploits and malware, including samples used in the Dragonfly industrial control system malware campaign (Havex), demonstrate that concrete graphs developed from their network traffic do not correspond to subgraphs of the model graph. This feature is leveraged to discern malicious packets in large traffic captures from substation automation networks. The novel contribution is a network forensic approach that considers the topological and timing characteristics of intelligent electronic devices induced by the power system.

2. Problem Statement

The research problem involves the dissection and forensic analysis of network traffic from an electrical substation network in order to reliably separate network packets generated by malware from packets generated by intelligent electronic devices during their normal monitoring and protection functions.

The inputs comprise the following data:

- A log file containing the physical parameter values of an electrical substation of interest over time as measured by various sensors. Examples of physical parameters include voltages, currents, temperatures, phases, frequencies and amplitudes.

- A network capture file in `pcap` format that contains traffic captured from the electrical substation network. Each packet is timestamped, as it is commonly the case when network traffic is captured with a tool such as Wireshark.

The outputs comprise the following data:

- A file in `pcap` format that contains all the network packets that the forensic analysis has determined to have been generated by malware.

- A second `pcap` file that contains only the network packets that the forensic analysis has determined to have been generated by intelligent electronic devices as part of their normal operation.

The proposed network forensic approach leverages the logical coupling between intelligent electronic device functions and the state of the power system. This coupling is inherent in an electrical substation. The analysis is conducted based on the IEC 61850 network communications standard.

Intelligent electronic devices do not exercise a directional comparison protection function based on so-called PDIR logical nodes if the forward and reverse-looking instantaneous directional overcurrent or distance elements at each line terminal do not activate the directional comparison scheme in use. If the directional comparison protection function is not exercised, then no network packets are exchanged, except for packets that carry periodic parameter measurements. Otherwise, if the parameters cause the activation of the directional comparison scheme, then the intelligent electronic devices generate network traffic that include commands that trip one or more circuit breakers.

A similar observation applies to the distance protection function for so-called PDIS logical nodes, directional overpower for so-called PDOP logical nodes and, in fact, all the other protection functions performed by intelligent electronic devices in an electrical substation.

Srivastav et al. [6] have discussed the regularities existing in industrial control network traffic. Unlike conventional information technology network traffic that fluctuates at high values, industrial control network traffic is relatively stable and somewhat predictable. The proposed forensic approach deciphers the relationships between the power system state and the network traffic generated by intelligent electronic devices. It engages these relationships to distinguish between normal and malicious traffic.

3. Graph Construction

This section describes the construction of the model and concrete graphs.

3.1 Model Graph Construction

The model graph formally describes normal network traffic in a substation automation network. Each vertex in the graph represents a pair comprising an intelligent electronic device and its monitoring or protection function. The model graph is directed and weighted. An edge from vertex A to vertex B represents network traffic between the intelligent electronic devices represented by vertices A and B, respectively. An intelligent electronic device can be the source or destination of network packets. However, the intelligent electronic device represented by vertex A is the initiator of the network communications; specifically, this device should have initiated the TCP three-way handshake by transmitting a packet with the TCP SYN flag set. In another scenario, the intelligent electronic device represented by vertex B could initiate network communications with the intelligent electronic device represented by vertex A.

However, all the network traffic in the associated TCP session is represented by an edge from vertex B to vertex A, regardless of which devices are the sources and destinations of specific packets in the session.

When the device represented by vertex A sends a UDP packet to the device represented by vertex B as it begins to exercise a function and, thus, possibly trigger the exchange of other UDP packets, then the edge goes from vertex A to vertex B. Each edge in the model graph has a weight that is determined via time series analysis. Every intelligent electronic device in an electrical substation has a set of monitoring and protection functions that it can perform; this increases the number of vertices present in the model graph. For example, if an electrical substation has just three intelligent electronic devices, each with the ability to execute 200 functions, there would be a total of 600 vertices in the model graph corresponding to this rather small electrical substation. As discussed later, knowledge of the specific monitoring and protection functions performed by an intelligent electronic device is of utmost importance.

Exploits and malware that affect an intelligent electronic device can only send or receive network traffic associated with the monitoring and control functions performed by the device. Otherwise, their malicious network packets would stand out and be trivial to spot.

Consider a situation where an intelligent electronic device is supposed to exercise a transformer protection function (and no other function) at time t_x for 4 ms. To conceal malicious network traffic, malware running on or targeting the intelligent electronic device could only send or receive traffic at time t_x, and typically not longer than 4 ms. If the intelligent electronic device in question is not supposed to exercise any functions at time t_x, then it would not send or receive any traffic at time t_x. As a result, the malicious network traffic would stand out.

An intelligent electronic device is a real-time machine, meaning that its reaction time is almost instantaneous. However, there is a minuscule delay in the transmission of its response to an anomalous event. The time taken for the intelligent electronic device to exercise its monitoring and protection functions is known; the expected time may be expressed as a range (e.g., 1 to 4 ms). However, if the network traffic continues to flow for 5 ms or longer, there is a high likelihood that some of the traffic was generated by malware. The malware traffic may not exhibit the typical patterns of electrical substation traffic. Thus, small differences in network activity could be very useful in identifying malicious network traffic.

Monitoring and protection functions are exercised by intelligent electronic devices under a specific set of physical parameter values in order to protect electrical substation equipment. The reporting of the parameter values by sensors is time critical because the intelligent electronic devices must perform their tasks in a timely manner [2]. An example protection function is the redirection of power. If a tree were to fall on a power line and cause a power outage, the protection function would redirect the power around the fallen line to deliver power to the affected customers. The power transformer protection function

switches off a power transformer in an electrical substation when an internal fault is detected. Switching off the transformer has the effect of protecting it; if the power transformer were to be unprotected, it would explode in the worst-case scenario.

An internal fault is detected by comparing the primary and secondary currents. If the currents exceed the computed thresholds, then a fault is assumed and the transformer is switched off. Rahman and Jeyasurya [5] have experimented with several power transformer protection function algorithms; the curve fitting, rectangular transform, Fourier and finite impulse response algorithms were determined to be feasible for implementation in microprocessor-based transformer differential relays. For the power redirection and power transformer protection functions mentioned above to be exercised by intelligent electronic devices, it is necessary for the relevant sensors to report anomalous power system states, namely certain ranges of physical parameter values that characterize the physical processes of the substation.

The two protection functions are exercised by intelligent electronic devices only when the sensor values are in the stipulated ranges. This consideration applies to all monitoring and protection functions. Clearly the physical parameter values are not important to malware; as a result, malware does not adhere to the same structural and time restrictions that the substation protection functions must satisfy. Knowledge of some essential characteristics of the electrical substation of interest helps identify information that is used to create the model graph. Two pieces of critical information are the intelligent electronic devices and the specific functions they can exercise. These enable the determination of the vertices and edges in the model graph.

Two methods may be used to determine the model graph vertices and edges. The first method is to examine the configuration files of the intelligent electronic devices. The configurations indicate the functions that are exercised by the devices and the other intelligent electronic devices with which the devices exchange network traffic while exercising their functions. The drawback of this method is that significant manual effort is required to develop the vertices and edges corresponding to all possible functions exercised by all the intelligent electronic devices in a substation.

The second method is to write code that observes normal substation automation network traffic and learns the vertices and edges. The drawback of this method is that some functions are exercised very infrequently; thus, observing normal network traffic even for long periods of time may not be enough to discern the rarely-implemented functions. The proposed approach combines the two methods; it applies machine learning to network traffic only to compute the weights of the edges in the model graph.

Fiigure 1 shows a small portion of a model graph associated with an electrical substation. The edge weights are integer values to simplify the presentation. The substation has three intelligent electronic devices, 1, 2 and 3, each with two functions, A and B. Decisions about when to exercise the functions are made based on the reported values of ten physical parameters. Six vertices are

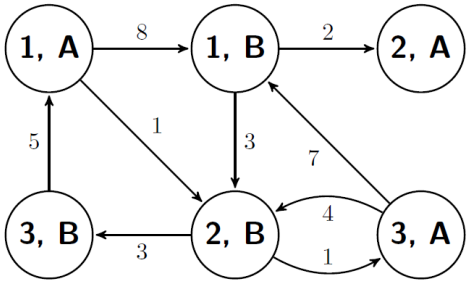

Figure 1. Sample model graph.

required to develop this portion of the model graph, one for each (intelligent electronic device, function) pair.

It was decided to construct the model graph in this manner because the intelligent electronic devices and functions had to be expressed explicitly. Using a single vertex for each intelligent electronic device was not an option because there can only be one directed edge from an origin vertex to a destination vertex. If this approach had been employed, the time series analysis would have represented all the network packets from an origin device to a destination device relative to all the monitoring and protection functions implemented by the two devices. As a result, it would not be possible to discern which functions were exercised and when they were exercised, making it exceedingly difficult to identify malicious network traffic.

In the model graph, physical parameters are directly used when defining each vertex; in fact, the monitoring and protection function modeled by a vertex depends entirely on the physical parameters. The reasoning is as follows: if the physical parameter values of a power system warrant a specific monitoring and control function being exercised by an intelligent electronic device, then a vertex exists and this vertex has an edge with a weight that connects it with another vertex. The model graph covers all the possible states of a power system where the states are expressed in terms of physical parameter value ranges. This is one more reason why the model graph can be extremely large.

3.2 Concrete Graph Construction

A concrete graph is constructed from the network packets contained in a `pcap` file. It is not known *a priori* whether the network packets are normal or malicious. Furthermore, if the `pcap` file contains malicious network packets, it is not known which packets are malicious and which packets are normal. The forensic analysis involves the construction of the concrete graph and the use of subgraph isomorphism search to determine the malicious and normal packets.

The vertices of the concrete graph are determined by examining the headers of the packets for the source and destination IP addresses. The IP addresses help identify the intelligent electronic devices that were senders or receivers

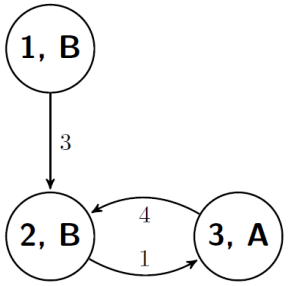

Figure 2. Sample concrete graph of normal activity.

of network traffic. The `pcap` files are timestamped, which enables the determination of the times when network traffic began to flow in the substation automation network.

The concrete graph is constructed by referring to a log file that contains the physical parameter values over time as measured by the sensors. The `pcap` file containing network traffic is then examined. The physical parameter values are used to determine the monitoring and protection functions that the addressed intelligent electronic devices exercised at specific times that the network traffic was recorded. The vertices in the concrete graph are marked based on the identified monitoring and protection functions and the IP addresses observed in the network traffic. The source and destination IP addresses identify the two intelligent electronic devices involved in a TCP session and the specific device that started the TCP session or initiated UDP communications. This information is sufficient to define all the edges of the concrete graph. The weights of the edges are computed via time series analysis as in the case of the model graph.

A concrete graph is compared against the model graph using subgraph isomorphism search. Checking for isomorphism between the concrete graph and a subgraph of the model graph relies on a graph theoretic technique (see, e.g., [7]). Graph isomorphism is a topic that has been researched for decades; a number of algorithms and library implementations for subgraph isomorphism search have been developed.

Figures 2 and 3 show two concrete graphs constructed from traffic in the same network as the partial model graph shown in Figure 1. The edge weights were again chosen to be integer values to simplify the presentation. The concrete graph in Figure 2 was determined to be a subgraph of the model graph; therefore, the captured network traffic represented by the concrete graph was not malicious.

On the other hand, in the case of the concrete graph in Figure 3, the weight of the edge from the origin vertex (1, A) to the destination vertex (2, B) is 5, which is not the same as in the partial model graph in Figure 1. This implies that the concrete graph is not a subgraph of the model graph and the network traffic is, therefore, determined to be malicious. The physical parameter values when

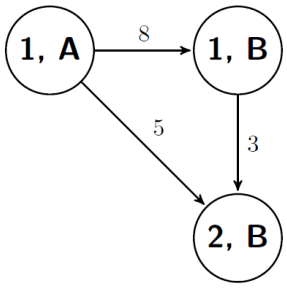

Figure 3. Sample concrete graph of malicious activity.

intelligent electronic devices exercise their monitoring and control functions provide additional clues to whether or not traffic is malicious. Specifically, if the physical parameter values are within the normal ranges and monitoring and control functions are still exercised (i.e., unnecessarily), then malicious network activity has occurred.

4. Time Series Analysis

This section demonstrates how time series analysis is used to compute the weights of the edges in the model and concrete graphs.

Intelligent electronic devices in an electrical substation perform computations and send/receive network traffic in a manner that exhibits strong dependencies on time and the power system state. The logical nodes in intelligent electronic devices do not act randomly. Each possible state of the power system demands that specific intelligent electronic devices exercise specific functions (i.e., manifest specific logical nodes) and these functions are carried out within specified time thresholds. The time constraint (threshold) imposed on a function demands that a logical node in an intelligent electronic device distributes all its function computations and I/O processing (and the traffic it transmits to other intelligent electronic devices) over time such that it satisfies the time threshold.

Experimental research in a laboratory environment has determined that the distribution of computations and I/O processing of a logical node in an intelligent electronic device over time is predictable. Certain fluctuations are observed in the times that network traffic sent by an intelligent electronic device on behalf of a logical node or logical device; however, the fluctuations are within predictable ranges. If this were not the case, intelligent electronic devices would send network traffic to each other too late, failing to meet the time thresholds of the overall monitoring and protection functions represented by the logical nodes. The relationship between network traffic and time for each monitoring and protection function exercised by an intelligent electronic device helps increase the granularity of the graph representation of substation automation network traffic. Thus, the crucial knowledge includes not only the direction

and structure of the communications between one intelligent electronic device and another, but also the time characteristics of the network communications between the devices involved in a monitoring and protection function.

The weight of an edge from an intelligent electronic device A to another intelligent electronic device B is computed by developing a time series of all the network traffic that A sends to B and all the network traffic that B sends to A in response when a monitoring and protection function is exercised. If B is the initiator of the traffic that it sends to A, then this traffic and the traffic that A sends B in response are associated with a different time series. This is because, there is an edge from vertex A to vertex B and another edge from vertex B to vertex A.

The time series comprises the observations of network payloads sent at time t and excludes the packet headers. In other words, the observed data corresponds to a set of data of size x_t sent at time t. Note that these observations are made of network traffic sent by intelligent electronic devices over a substation automation network that is not under attack.

Thus, the proposed network forensic approach has anomaly detection characteristics in that there is a learning phase involving normal network communications between intelligent electronic devices. This learning phase requires a network capture file in the `pcap` format that is obtained by passively – and thus safely – sniffing substation automation network traffic (e.g., using Wireshark). The choice of timestamp precision is critical to the construction of the time series. Since the time window of most monitoring and protection functions ranges from zero to the time thresholds set for the functions (e.g., 4 ms), the timestamp precision in the proposed approach was set in the order of microseconds and the time unit was set to 100 ms. Thus, the time series of a monitoring and protection function would typically go from $t = 0$ to at most $t = 40$.

Time series with longer time windows may be constructed if there is enough network traffic. This is rarely the case for normal substation automation network traffic, but it is more likely for malware traffic. The time series constructed are discrete in that the sets of times t at which observations are made of packet payloads x_t are discrete.

The observations of network traffic over time are cumulative. If device A sends 500 bytes of data to device B at time a and device B responds with 300 bytes of data at time b, then $x_a = 500$ and $x_b = 800$. The time series is viewed according to the classical decomposition model:

$$x_t = w_t + s_t + y_t \tag{1}$$

where the w_t component of the observed x_t is the trend in the time series terms and is the factor that is leveraged by the proposed approach; s_t corresponds to the seasonality, which was noticeably absent in the time series encountered in the experiments; and y_t is the residual component in the time series terms.

The trend w_t is deterministic and is characteristic of a specific time series. This makes the trend a desirable differentiator. Given that the trend is specific to the time series representing normal network traffic for a monitoring and

Table 1. Testbed device functions and communications protocols.

Machine	Function	Protocol
SEL-487E-3	Transformer protection relay	IEC 61850
SEL-421-4	Automation, protection	IEC 61850
SEL-3555	Automation, data, HMI	IEC 61850
Windows Server	OPC server	OPC, IEC 61850
Windows Client	OPC client	OPC
Windows Machine	Development, testing	OPC, IEC 61850

protection function, its value over time would differ from the trend in a time series representing malicious network traffic. Note that the trend w_t of the time series representing normal network traffic is computed during the learning phase. The weight of the edge that goes from device A to device B is set to be a vector of trend values w_t, namely $[w_1, w_2, ..., w_n]$ where n is the end of the time window. During actual network forensic analysis, the time series of the unknown traffic to be analyzed is computed, the trends of each of the time series are calculated and the weight of the corresponding edge is then determined.

The trend of a time series is computed using the least-squares estimation method. The model trend is first expressed as a quadratic function of time:

$$w_t = \beta_0 + \beta_1 t + \beta_2 t^2 \tag{2}$$

Next, the quadratic function of time is fit to the network traffic and the values of the parameters β_0, β_1 and β_2 are determined by minimizing the following equation:

$$\sum_{t=1}^{n} (x_t - (\beta_0 + \beta_1 t + \beta_2 t^2))^2 \tag{3}$$

The optimal values of β_0, β_1 and β_2 are used to compute the trend w_t at each t according to Equation (2), yielding the final edge weight.

Note that perfect matches between edge weights in the model and concrete graphs are neither sought nor desired. This is because fluctuations are normal and are, indeed, expected. In fact, the subgraph isomorphism search algorithm looks for weights that deviate substantially from those in the model graph.

5. Experimental Evaluation

The experiments leveraged a testbed with protective relays that emulated a real-world electrical substation. Table 1 summarizes the main components of the testbed along with their functions and communications protocols. The SEL-487E-3 device is a protective relay designed to monitor and protect a power transformer from electrical faults; it runs intelligent algorithms that detect various types of faults and take actions in a timely manner by operating electrical

circuit breakers and disconnect switches. The SEL-421-4 device performs industrial automation functions; it comprises 32 programmable elements for local control, remote control, automation latching and protection latching. It also performs various functions that protect overhead electrical transmission lines and underground cables. The SEL-3355 device performs multiple substation functions; it has an integrated human-machine interface (HMI) with a local display port. In the experiments, the SEL-3355 device was used as a real-time automation controller. The SEL-3355 polled the SEL-487E-3 and SEL-421-4 for substation data. The network communications between these industrial control devices employed the IEC 61850 protocol.

The experimental testbed incorporated a Windows platform hosting an Object Linking and Embedding (OLE) for Process Control (OPC) server [3], which executed an IEC 61850 protocol driver to obtain substation data (including decoy data) from the protective relays. The IEC 61850 protocol driver subsequently stored the substation data in the OPC server. The testbed also incorporated an OPC client running under Windows; the OPC client application provided a graphical user interface for a human operator to enter OPC commands. Another Windows machine was used for development and testing. All the machines and devices in the experimental testbed were connected via a local area network.

The experiments validated the proposed approach using malware samples involved in the Dragonfly cyber espionage campaign. Several versions of the Dragonfly malware exist; the malware samples used in the experiments were obtained from public malware research repositories. Versions of the Havex industrial control system malware were also used; these were of particular interest because they use the OPC protocol.

The experiments emulated malware propagation and execution in a substation automation network. The experiments used the `nmap` tool to scan for the IP addresses of intelligent electronic devices and their network services. The vulnerability exploitation phase involved code that targeted several memory vulnerabilities using shellcode injection and heap spraying on intelligent electronic devices. Two exploitation scenarios were implemented, one where the malware had no prior knowledge of the power system state and the other where the malware was able to obtain complete knowledge about the power system state before launching its attacks.

The experiments also emulated the malware installation phase, which involved the injection and execution of a dropper on a compromised intelligent electronic device. As with traditional malware, the dropper was responsible for installing malware modules on the compromised intelligent electronic device. Single-stage and dual-stage dropper were emulated. The single-stage dropper incorporated the emulated malware modules that were installed on the compromised intelligent electronic device whereas the dual-stage dropper downloaded the modules from another compromised machine over the network.

Traffic associated with the network scans and test exploit that operated without any prior knowledge of the power system state produced visible struc-

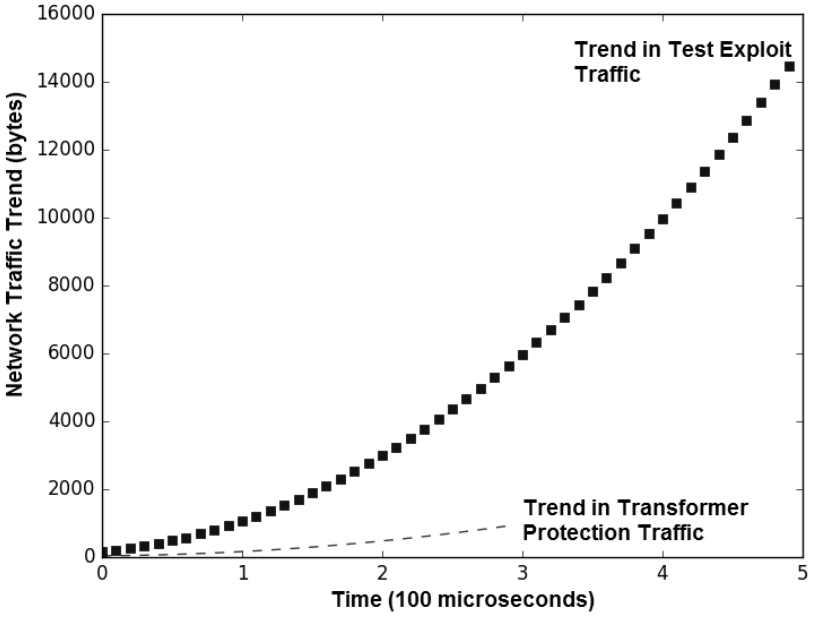

Figure 4. Time series trend comparison results for a test exploit.

tural deviations from the model graph. The target selection was random as were the times when the network probes and test exploit were launched. Even when the launch time coincided (by chance) with the time that a protection function was exercised by an intelligent electronic device, the structural deviations from the model graph caused the subgraph isomorphism search algorithm to fail. Thus, no further time series analysis was necessary in the subgraph isomorphism search.

The time series analysis became necessary when the network scans and test exploit operated with full knowledge of the power system state. The network probes and test exploit were launched at the time that a power transformer protection function was exercised by the intelligent electronic devices. Furthermore, the probes and exploit were launched from a compromised intelligent electronic device to another intelligent electronic device, both of which were exchanging network traffic when the power transformer protection function was exercised. The graph structure of the network traffic matched that of the model graph. However, as shown in Figure 4, time series analysis detected deviations in the trend. This was due to the test exploit generating large quantities of network traffic when performing its heap sprays. Its time window also extended beyond the maximum 4 ms time window of the power transformer protection function. The subgraph isomorphism search failed because the edge weights

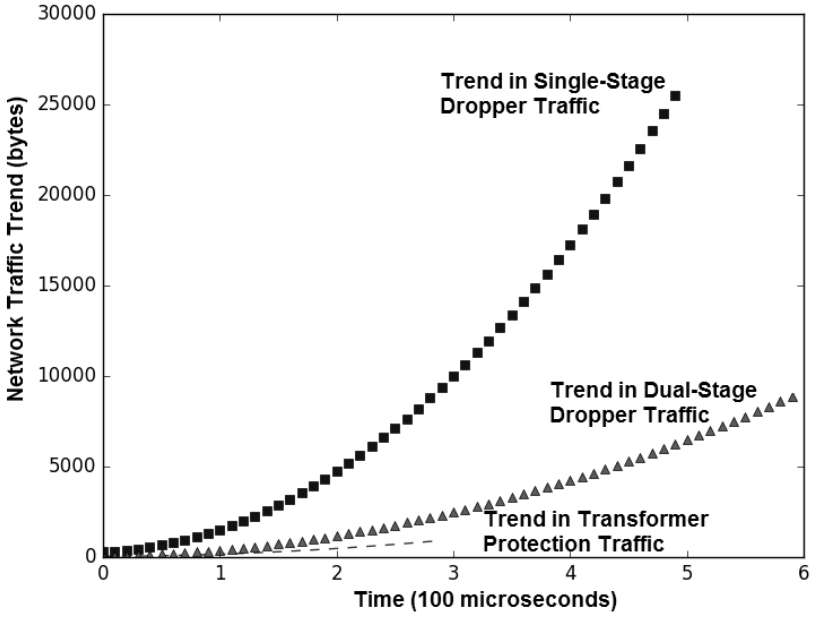

Figure 5. Time series trend comparison results for a test malware installation.

in the concrete network traffic graph deviated substantially from those in the model graph.

When malware installation was performed without any knowledge of the power system state, the network traffic caused by the dropper did not match a valid power system protection function (unless a coincidence occurred). The mismatch of the network traffic and substation protection function caused the subgraph isomorphism search to fail, resulting in the immediate detection of the malicious network packets.

A challenging situation for a defender occurs when the network traffic caused by the dropper is interwoven with the network traffic of a valid protection function. If the dropper were to be downloaded on a compromised intelligent electronic device with a high degree of mimicry, then the structural properties of the concrete graph would match those in the model graph. Since the network traffic would appear to be consistent with the substation protection function, the subgraph isomorphism search would have to rely on the time series analysis of the network traffic.

The weights in the concrete graph corresponding to the single-stage dropper deviated considerably from the weights in the model graph. The single-stage dropper incorporated several malware modules and, thus, its downloading on the compromised intelligent electronic device was characterized by heavy network traffic and the higher trend shown in Figure 5. In contrast, because the

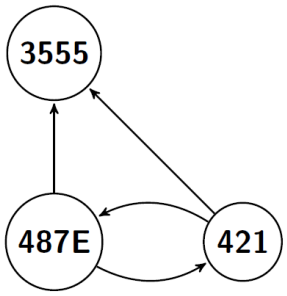

Figure 6. Excerpt topology of a concrete graph of normal network activity.

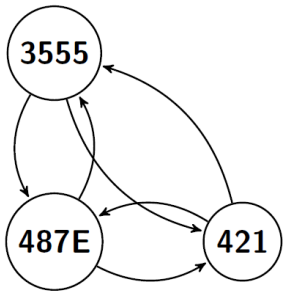

Figure 7. Excerpt topology of a concrete graph of malicious network activity.

dual-stage dropper was a thin client module that downloaded the malware modules over the network at a later time, much less traffic was observed than in the case of the single-stage dropper. Nevertheless, the time series in Figure 5 shows large deviations for the single-stage dropper and dual-stage dropper network traffic from the network traffic corresponding to a power transformer protection function.

The `nmap` tool and Havex malware generated network traffic with strong deviations in the trends, which caused the subgraph isomorphism searches to fail. The deviations in the trends were similar to the trends shown in Figures 4 and 5. The concrete graphs constructed from `nmap` and Havex network traffic as well as those constructed with the majority of the emulated malicious activities also demonstrated structural deviations from the model graph, which again caused the corresponding subgraph isomorphism searches to fail. Typical failures resulted from network traffic being sent to intelligent electronic devices that were not supposed to participate in the exercising of a function; this created additional vertices and edges in the concrete graphs, causing the subgraph isomorphism searches to fail. Figures 6 and 7 present portions of concrete graphs corresponding to normal traffic and malicious traffic, respectively. In general, the concrete graphs constructed from normal network traffic were easily determined to be subgraphs of the model graph.

6. Conclusions

The proposed network forensic approach treats an electrical substation as a cyber-physical system and explores its strong interrelations to develop a model graph that deeply characterizes the structural and timing properties of normal network traffic in the substation automation network. A similar concrete graph is constructed for unknown (potentially) network traffic. Unlike authorized code that executes on intelligent electronic devices, exploit code and malware generate network traffic that deviates from the monitoring and control tasks performed in an electric power system. In particular, the traffic generated by malicious code has structural and timing properties that differ from those of normal traffic. Thus, the problem of determining whether or not unknown network traffic is malicious involves a subgraph isomorphism comparison of the concrete graph against the model graph. Normal network packets yield a concrete graph that is a subgraph of the model graph whereas malware-generated network packets produce a graph that is not a subgraph of the model graph. Experiments involving real industrial control system malware demonstrate that the resulting approach can reliably identify malicious packets in large traffic captures from a substation automation network.

Acknowledgement

This research was supported in part by Defence R&D Canada and by the U.S. Office of Naval Research under DURIP Contract No. N00014-15-1-2891.

References

[1] I. Ahmed, S. Obermeier, M. Naedele and G. Richard, SCADA systems: Challenges for forensic investigators, *IEEE Computer*, vol. 45(12), pp. 44–51, 2012.

[2] K. Brand, C. Brunner and I. de Mesmaeker, How to use IEC 61850 in protection and automation, *Electra*, no. 222, pp. 11–21, 2005.

[3] J. Lange, F. Iwanitz and T. Burke, *OPC: From Data Access to Unified Architecture*, VDE-Verlag, Berlin, Germany, 2010.

[4] Y. Liang and R. Campbell, Understanding and Simulating the IEC 61850 Standard, Technical Report UIUCDCS-R-2008-2967, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 2008.

[5] M. Rahman and B. Jeyasurya, A state-of-the-art review of transformer protection algorithms, *IEEE Transactions on Power Delivery*, vol. 3(2), pp. 534–544, 1988.

[6] A. Srivastav, C. Ortega, P. Ahuja, M. Christian and A. Cardenas, Exploratory analysis of Modbus and general IT network flows in a water SCADA system, presented at the *Industrial Control System Security Workshop*, 2015.

[7] J. Ullman, An algorithm for subgraph isomorphism, *Journal of the ACM*, vol. 23(1), pp. 31–42, 1976.

[8] T. Wu, J. Pagna Disso, K. Jones and A. Campos, Towards a SCADA forensics architecture, *Proceedings of the First International Symposium on ICS and SCADA Cyber Security Research*, pp. 12–21, 2013.