BEHAVIOR-BASED APPROACH TO MISUSE DETECTION OF A SIMULATED SCADA SYSTEM

Brien A. Jeffries and Dr. J. Wesley Hines
Department of Nuclear Engineering
University of Tennessee
Knoxville, TN 38996, USA
bjeffri1@vols.utk.edu; jhines2@vols.utk.edu

Dr. Kenny C. Gross
Oracle Physical Sciences Research Center
San Diego, CA 92121, USA
kenny.gross@oracle.com

ABSTRACT

This paper presents the initial findings in applying a behavior-based approach for detection of unauthorized activities in a simulated Supervisory Control and Data Acquisition (SCADA) system. Misuse detection of this type utilizes fault-free system telemetry to develop empirical models that learn normal system behavior. Future monitored telemetry sources that show statistically significant deviations from this learned behavior may indicate an attack or other unwanted actions. The experimental test bed consists of a set of Linux based enterprise servers that were isolated from a larger university research cluster. All servers are connected to a private network and simulate several components and tasks seen in a typical SCADA system. Telemetry sources included kernel statistics, resource usages and internal system hardware measurements. For this study, the Auto Associative Kernel Regression (AAKR) and Auto Associative Multivariate State Estimation Technique (AAMSET) are employed to develop empirical models. Prognostic efficacy of these methods for computer security used several groups of signals taken from available telemetry classes. The Sequential Probability Ratio Test (SPRT) is used along with these models for intrusion detection purposes. The different intrusion types shown include host/network discovery, DoS, brute force login, privilege escalation and malicious exfiltration actions. For this study, all intrusion types tested displayed alterations in the residuals of much of the monitored telemetry and were able to be detected in all signal groups used by both model types. The methods presented can be extended and implemented to industries besides nuclear that use SCADA or business-critical networks.

Key Words: SCADA, misuse detection, Linux, SPRT

1 INTRODUCTION

A growing trend in the nuclear industry and many other industries in recent years is to shift towards a digital I&C culture. This shift is mainly due to the lower costs and increased sophistication in computer and communications technologies. The Supervisory Control and Data Acquisition (SCADA) system is an example of a combination of the previously mentioned technologies, using computers for process control/monitoring and sensing/communications equipment for data transfers. SCADA has seen use in many industries, including transportation, the energy sector and water treatment. Because these systems monitor and control the production and distribution of important resources, the network and related telemetry need protection from those with malicious intentions. Along with basic security measures such as plant network isolation, robust intrusion detection methods should be investigated and implemented to meet the needs of a growing digital industry and the unique new challenges that may arise.

This research presents a new application of empirical-based modeling, which until now has been used for degradation detection of plant sensors or process equipment, to cyber-security of a simulated

SCADA system. These data-driven modeling methods are the Auto Associative Kernel Regression (AAKR) and Auto Associative Multivariate State Estimation Technique (AAMSET). The basic idea behind using these two methods is that unwanted activities may show alterations in the behavior of monitored telemetry sources that are different from what was learned by the developed model as normal behavior. Models were developed using groups of signals selected from available telemetry to determine which signal groups offered superior prognostic security results. The Sequential Probability Ratio Test (SPRT), a binary hypothesis test that has seen wide use across many fields, is used in tandem with these models for quantification of alerts from intrusion-injection activities.

2 METHODOLOGY

This section will provide an overview of the various tools and methods used for this cyber-security research. First, the basic terminology and different types of intrusion detection methodologies that have been developed in prior research will be discussed. Next, the basic steps used for machine learning model development are given, followed by a discussion of the AAKR and AAMSET modeling methods. The section concludes with an overview of the SPRT, which is the main algorithm used quantify the alerts obtained from intrusion-injection activities as normal or faulted.

2.1 Intrusion Detection Methods

Intrusion Detection Systems (IDS) can be thought of as a combination of software and/or hardware with the purpose of monitoring current telemetry, such as network traffic or log files, for unauthorized activities [1, 2]. Detection of these activities can alert operators or take reactive measures to evict intruders, if implemented. The IDS also needs to have several properties to be considered useful. These include low overhead compute cost, timeliness in detection of alerts and ability to use telemetry from different sources [3]. For the purposes of this research, intrusion detection is network based rather than host based. This means that the activities of the network and all components are monitored rather than the activities or behaviors of a single host computer. As a variety of IDS have been developed, to avoid confusion these systems can be generalized into three basic types: Dictionary-based, Behavior-based and Specification-based [4].

Dictionary-based IDS use signatures or features stored in what is known as an "attack dictionary". These are features of known attacks or exploits that are compared with features of monitored signals. If there is a match, then an alert is given. The main strength of this type of IDS is that there is a low amount of false alarms and is rather simple to implement. Some weaknesses are that attack signatures that do not match the dictionary will be considered as normal, meaning that zero-day attacks can never be detected. Also, the dictionary must be updated often since new attack features are always being discovered, along with new ways to avoid detection by the attack dictionary.

Behavior-based IDS use empirical models such as AAKR, Neural Networks or Genetic Algorithms to learn the patterns of normal behavior for the system in question. The telemetry sources for these models can include data from physical process measurements or software/system resource usage metrics. After the model is developed, any future telemetry sources that show statistically significant deviations from the learned normal behavior will be considered as anomalous. One of the main strengths of this method is that zero-day attacks can be detected easily, which is not possible with the previous method. Also, new operating conditions can easily be incorporated into these machine learning methods, should the need arise. One of the main weaknesses is that there is a higher rate of false alerts, but this rate can be reduced with proper model tuning that will be discussed later in this section. It is noted that this type of IDS is not meant to replace the Dictionary-based IDS, but rather they should work in tandem with each other for added layers of computer prognostic security, or a "defense-in-depth" approach.

The last IDS is termed Specification-based, this method specifies behaviors or policies beforehand by a human security expert. Though this may sound similar to the previous two methods, the features or policies in the Specification-based IDS are not extracted features from an attack dictionary or learned normal system behavior. The main strengths of this method are that there is a low rate of false alerts given and there is no system profiling stage. A weakness of this method is that all specifications must be developed and programmed by a human operator, this can lead to errors or poorly defined policies.

2.2 Auto Associative Kernel Regression

In this and the following subsection, the empirical models used for this research will be discussed. The AAKR and AAMSET models were developed using algorithms contained in the Process and Equipment Monitoring (PEM) toolbox created at UT, which is a MATLab based set of tools [5, 6]. Developing these machine learning models can be broken down into several simple steps. The first step is model training, which requires data from when the system is operating in a fault-free state. The model is then optimized so that there is minimum error or uncertainty between the input data and model predictions. The last step is model validation, which uses data the model has not seen before to test overall performance. After these steps, current sensor information or faulted data can be run through the model to generate predictions and residuals. The residuals are obtained by subtracting the input data from the model predictions of that data and can be considered as model errors between input data and predictions. These residuals can then be used in a detection algorithm such as the SPRT to quantify normal versus faulted behavior.

The first of the empirical models to be discussed is AAKR, which is a class of nonlinear, nonparametric (NLNP) regression that generates predictions of input data. The auto associative name indicates that both the predictor and response variables are identical. To develop an AAKR model, all that is needed is a set of data, hereafter termed training data, which is fault-free and covers all operating ranges of the system. This assumption needs stating because this model type cannot accurately predict values outside of the range of data used to train the model. A subset of observations is taken from each signal in the training data and is used for model training. This subset is termed the memory matrix and can be thought of as representative vectors of the entire training set. Current inputs to the model will be compared with these memory vectors for similarity. For this work, a Euclidean distance function is used to calculate these distances, shown in Equation 1.

$$d_j = \sqrt{\sum_{i-1}^{n} (X_{j,i} - x_i)} \tag{1}$$

In Eqn. (1), d are the distances for each sensor j, $X_{j,i}$ is the current memory vector observation and x_i is the current input. These distances are used in a Gaussian kernel function to obtain weights that are used to generate model predictions; this kernel function is shown next in Equation 2.

$$K_j = \frac{1}{\sqrt{2\pi h^2}} \exp\left(\frac{-d^2}{2h^2} \right) \tag{2}$$

In the preceding equation, d are the distance values calculated and h is called the kernel bandwidth. The resulting weights K for each sensor j are then finally used to generate the model predictions. The kernel bandwidth is a parameter that needs to be optimized so that under or over-fitting of model predictions is avoided. This means that poor optimization can lead to an introduction of excess model bias or excess model variance. In other words, attempting to increase the model accuracy can introduce model variance and attempting to reduce the model uncertainty can lead to an increase of model bias. As stated previously, subtracting the model predictions from the input data generates residuals used for detection.

2.3 Auto Associative Multivariate State Estimation Technique

The AAMSET data-driven model was based upon a similar class of NLNP mathematical methods that was used to develop the MSET algorithm at Argonne National Laboratory (ANL) and later commercialized by SmartSignal [6, 7, and 8]. It is stressed that the AAMSET in the PEM toolbox does not use the same proprietary kernel that was used to develop the ANL MSET. The training data that is used to develop the AAKR models can also be used to develop an AAMSET model. Beyond this point, there are some differences in how predictions are generated using this technique.

The first main difference is that the memory matrix is much smaller than that used for the AAKR models. For this technique, the number of memory vectors selected for each signal is only four times the number of signals in the data set. For example, a particular data set may have 20 signals. An AAKR model would then use 500 observations for each signal to develop the memory matrix, while an AAMSET model would only require 80 observations for each signal. The obvious advantage here is that the overall compute cost required to develop and run an AAMSET model is significantly reduced when compared with AAKR.

Once this memory matrix is developed, the distances between this matrix and an input vector are calculated using Eqn. (1) to obtain what is know as a similarity matrix. To obtain the final model predictions, a pseudo-inverse is employed. The similarity is inverted using this method to net a normalized similarity matrix. In keeping with the previous example, this normalized matrix would have a size of 80 x 80. Using this normalized matrix, the weights between this and a query or input vector can then be calculated using Eqn. (3). The weights, normalized similarity matrix and query vector are combined to then obtain the final predictions, the combined equation is shown next.

$$\mathbf{Y_p}= w'*((\mathbf{X}^{\mathbf{T}}*\mathbf{X})^{-1}*\mathbf{X}^{T})*Q \tag{3}$$

In the preceding equation, $\mathbf{Y_p}$ are the final model predictions, w are the weights, $\mathbf{X}$ is the similarity matrix and Q is a vector of inputs. As in the AAKR model, the residuals are then obtained by subtracting the predictions from the input data. One drawback of this method that is worth mentioning is that an ill-conditioned matrix can lead to poor model predictions. This can be alleviated by regularization of the memory matrix during the pseudo-inverse process.

2.4 Sequential Probability Ratio Test

The last method to be discussed and the main anomaly/intrusion detection engine used is the SPRT, which is a binary statistical hypothesis test developed by Wald [9]. The main assumption in using this test is that the data under observation be Gaussian distributed. In practice this may not always be true, though there are more complex non-parametric SPRTs available that could be used for these cases. As the name implies, the SPRT examines observations sequentially to arrive at a determination of being in a normal or faulted state. For this test, the normal or unfaulted state is assigned to the null hypothesis, H_0. The anomalous or faulted state is assigned to an alternative hypothesis, H_1. The test then determines which of the two hypotheses the current observation of the data belongs to. The SPRT does this by calculating what is known as the log-likelihood ratio, this test statistic is a ratio of the probabilities of the observation being in a faulted versus unfaulted state. The equation for this likelihood ratio is shown next.

$$L_N = \frac{p(\{x_n\})/H_1}{p(\{x_n\})/H_0} \tag{4}$$

In Eqn. (4), L_N is the log-likelihood ratio or test statistic. The numerator is a probability ratio that the current observation x_n is in a faulted state or belongs to H_1. The denominator is also a probability ratio that the current observation x_n is in a normal state or belongs to H_0.

To be able to apply the test statistic shown in Eqn. (4), an upper and lower test bounds must be defined. These test bounds are a decision criteria that the test statistic is compared against, calculation of these test bounds in shown in the next equation.

$$A = \ln\left(\frac{\beta}{1-\alpha}\right), \quad B = \ln\left(\frac{1-\beta}{\alpha}\right) \tag{5}$$

In Eqn. (5), A is the lower test bound and B is the upper test bound. Alpha is the false alarm probability or probability of making a Type I error. Beta is the missed alarm probability or probability of making a Type II error. If the test statistic for the current observation is less than A, then the observation is in a normal state. If the current observation exceeds the upper limit B, then the observation is considered to be in a faulted state. If the test statistic value is between these two limits, then there is not enough information to arrive at a decision and testing then continues.

For this research, there were four possible alternative hypotheses that were tested. The first two alternative states are a positive or negative shift in the mean, with constant variance, when compared to the mean of the distribution defined by H_0. Mean shifts in physical systems usually indicate that some form of degradation is present. The other two alternative states are an increase or decrease of the variance, with constant mean, when compared to the variance of the distribution defined by H_0. Variance shifts in physical systems usually indicate that unwanted variability is present. Fig. 1 gives a graphical representation of these different alternative states.

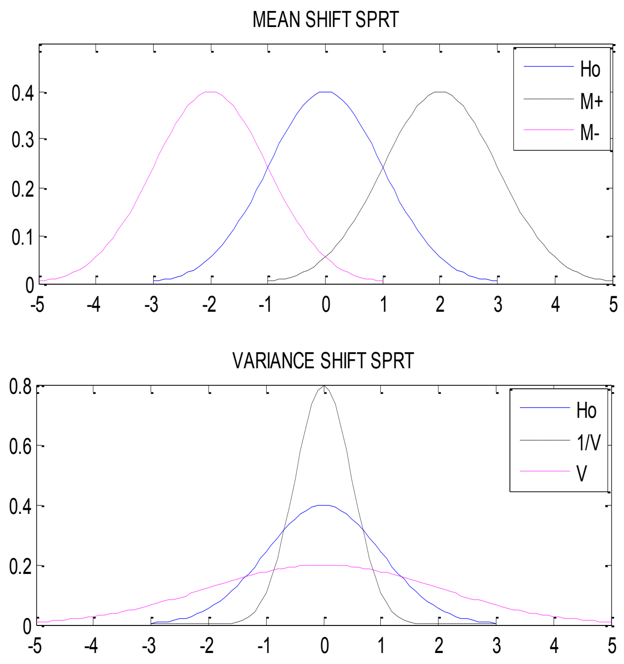

Figure 1. SPRT Hypothesis Distributions

In the previous figure, H_0 is defined as a normal distribution with zero mean and unit variance. M+ indicates a positive mean shift from H_0 and M- indicates a negative mean shift. The variance factors V and 1/V indicate that the variance has increased or reduced, respectively. A more thorough treatment on the derivation of the mean and variance SPRTs described here can be found in [10]. Now that the empirical-based modeling methods and fault detection algorithms have been defined, we now turn to the experiment test bed and a description of what intrusions were tested and detected with these methods.

3 EXPERIMENT SETUP & INTRUSION TESTING

The simulated SCADA test bed utilizes a set of Linux based enterprise servers that were isolated from the larger UT Nuclear Engineering Research Cluster. Isolation was required to ensure that activities on the test bed and on the larger cluster would not be reflected in the dynamics of each system. Isolation also ensures that intrusion-injection experiments would not impair the performance of the larger cluster. One server in the test bed acts as the Master Terminal Unit (MTU) in a typical SCADA system. This component is tasked with monitoring field equipment, issuing command actions and a variety of data processing actions. The remaining isolated servers act as a Remote Terminal Unit (RTU), this equipment is tasked with maintaining field equipment operations and data transfer activities. Also included in the setup is a separate server that acts as a Human Machine Interface (HMI) or engineering workstation. In a typical SCADA system, this station serves to monitor the system and to issue control actions to the MTU. In the test bed the HMI is used to monitor the simulation and to perform intrusion-injection experiments. The HMI also uses Kali Linux as the OS, which is one of the vulnerability testing tools for this work [11].

As mentioned previously, the experiment is a simulation of a typical SCADA system. This is necessary because, for security reasons, actual sanitized process data from a SCADA or business-critical network is very difficult to obtain. A simulation does not devalue the research or results, as many of the advances made in the nuclear industry began with small scale simulations and experiments that were then shown to have scalability to larger systems. With this in mind, the simulation is based on several assumptions. First, the dynamics of the MTU in a typical industry or asset manufacturing setting can exhibit periodic components. This is valid due to the fact that the MTU performs specific tasks such as data polling and processing on a regular basis. It is also assumed that there will be a large amount of network traffic for larger systems with hundreds of components and sensors.

Task scripts were developed to first generate a baseline working profile of the MTU. These tasks are performed on a fairly periodic manner and load all cores to various levels. This simulates the data processing activities performed by the MTU. While these task scripts are running, the MTU is also collecting data from several sources and writing to a data store. This simulates the data polling and storage activities of this component. Additional task scripts are run by both MTU and RTUs to simulate command actions, data polling and other network related activities. All of these particular network actions are sent over SSH. The combination of MTU and network actions results in telemetry with periodic and random components that would be observed in the dynamics of a typical SCADA system. Monitored telemetry included resource and network related statistics taken from the Linux kernel, as well as internal hardware power usage and tachometer measurements.

Intrusion testing tools utilized for this research included Kali Linux, Metasploit and past/current exploits related to Linux based systems from the Common Vulnerabilities and Exposures (CVE) database [11, 12, and 13]. Kali contains several different software packages that can be used to test for vulnerabilities on private networks or web-based applications. The CVE database contains past and current exploits, often with testing code, for a variety of systems. It is noted that many of the tools and exploit codes were not applicable to our simulation or caused no perceivable change in observed system dynamics. The data-driven models shown in this paper would not be suited for detection of these intrusion types and would need a dictionary-based IDS or some form of post-attack log analysis to detect if there was an intrusion event. From a computer security standpoint, any actions taken by an unauthorized user on the network will be considered as an intrusion activity. For this work, several different types of intrusions were tested, including host/network discovery, Denial of Service (DoS), brute force login, privilege escalation and exfiltration actions. These were tested in succession, with short rest periods between certain attacks to allow for the system dynamics to return to normal. The idea behind running these different intrusion types was to determine which caused changes in telemetry and could be detected with these methods. Also, the research was focused on which set or subset of monitored telemetry could be used for security purposes instead of building a model with hundreds of telemetry sources.

4 MODEL RESULTS

The first topic to be discussed in this last section is related to variable selection for model development. As stated previously, one focus of this research was determine the prognostic efficacy of using small sets of signals taken from available telemetry rather than developing one large model using all signals. In total there were 685 signals taken from various telemetry sources, though many of these signals were constant valued or had few changes in signal values over testing. These offer no information for modeling or security purposes and were removed prior to final variable selection. The remaining signals are grouped based on related cross-correlation values, which is an indicator of linear relationships among signals [14]. Cross-correlation values for this work are scaled between 0 and 1. Signals with strong relationships have a cross-correlation value of .7 and above, moderate relationships have a value between .3 and .7, and signals with values below .3 have little or no relationships. Using this method, four separate groups were extracted. The first uses a combination of memory and CPU usage metrics, the second signals selected from all telemetry sources, while the remaining two groups used only network related and disk usage signals, respectively.

Next, for all SPRT tests used in both model types, the alpha and beta values were set at 1%. Recall that these are the probabilities of making a Type I or Type II error, the prescribed 1% means that were are willing to accept a small occurrence of false and missed alerts. For prognostic security purposes, detection success was defined as at least one signal in each group had to detect all six intrusions tested. By this definition, all groups of both model types were successful at detecting all intrusions. However, it is also worth mentioning which group of signals performed the best and worst. Of the previously listed groups, Group 1 had only 2 signals that caught all intrusions, Groups 2 and 4 were both able to detect all six in 85% of all signals used. The best was Group 3, which used all TCP/IP signals, where all signals used were able to detect all intrusions.

To conclude this section, intrusion detection results from both model and SPRT types will be shown. In this particular data set, a tool called SPARTA was employed for network discovery, followed by three DoS attacks were used against three different running processes. Next, access to the system was gained by a brute force attack, immediately followed by rapid privilege escalation. The attack concludes with the theft of several files from the target server over a half hour period. In the upcoming figures, the top subplot is the model residual for a particular signal. Recall that the residual is the original input signal subtracted from the model predictions. The bottom subplot is the SPRT fault hypothesis scores, a value of 0 indicates an unfaulted state and a value of 1 indicates a faulted state. For reference in the upcoming figures, the observation ranges for each of the intrusions presented is listed next:

SPARTA – 245:345

NTP Fuzzer – 430:550

SSH Fuzzer – 580:640

Smb2 Fuzzer – 655:710

SSH Brute Force/Privilege Escalation – 880:960

Information Theft – 975:1100

The first result shown will be for AAKR Model 3. This model used all strongly correlated TCP/IP signals. In Fig. 2, the residual for the selected "OutOctets" signal is shown, along with the SPRT fault results. The residual indicates large positive and negative mean shifts during intrusions. The variance shift SPRT for the both model types of this group also gave the same results as that seen in Fig. 2.

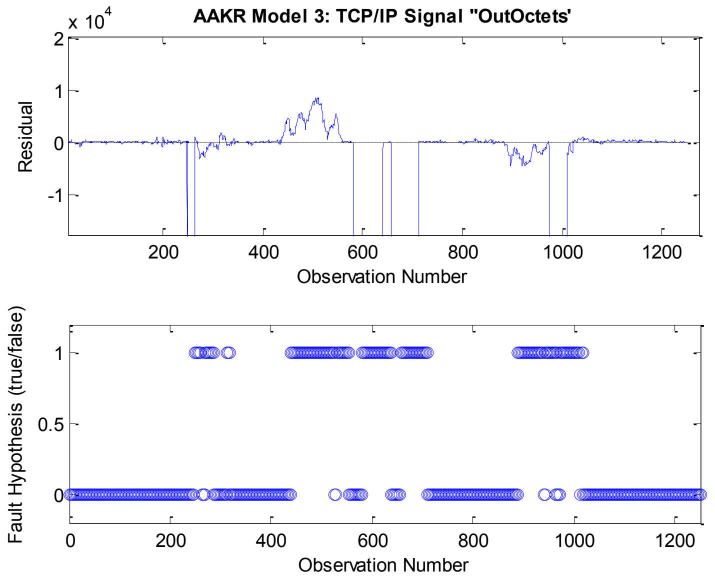

Figure 2. AAKR Model 3: TCP/IP Signal "OutOctets"

In the previous figure, the magnitude of the large residuals that extend beyond the range of the graph was on the order of 2 x 10^7. Compare this to the small magnitude of the first fuzzer and brute force attack, both which were several orders of magnitude smaller. This is important to mention because a simple threshold technique might miss the smaller magnitude intrusions, however, the SPRT was able to correctly identify all six different intrusion types with vastly different magnitudes. Also, not all signals in this group showed the same obvious change in residuals as in the figure, yet all intrusions were detected by these as well. Next, a disk usage signal from AAMSET Model 4 is shown in Fig. 3. The residual shown for this signal used the variance shift SPRT, and the residual shows large variance changes in the ranges of several of the tested intrusions.

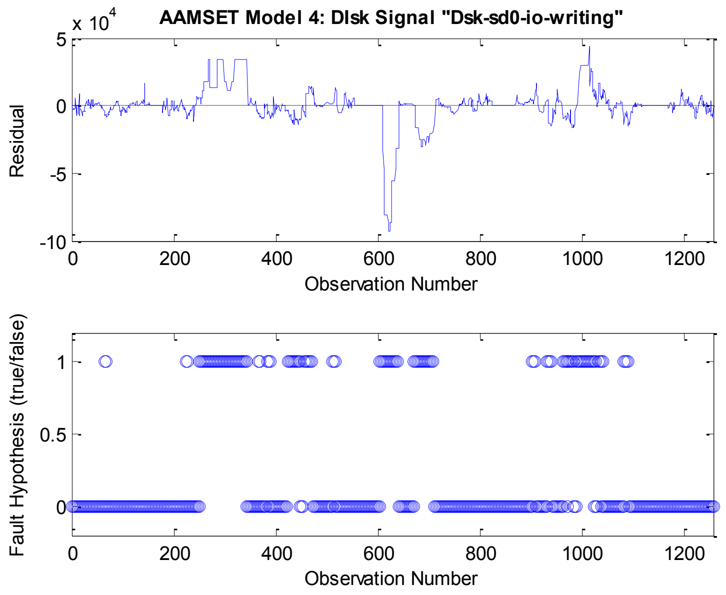

Figure 3. AAMSET Model 4: Disk Signal "Dsk-sd0-io-writing"

The results shown in Fig.3 apply the variance shift SPRT to a disk usage signal. Similar to the results of Fig. 2, the NTP fuzzer and password attack signatures for this residual and all others in this group showed the least change in magnitude, but were still detected. The SSH fuzzer attack caused the greatest change seen in this signal out of all intrusions tested. This result is expected because all data transfers and related actions are performed over SSH. Thus, an attack that servers to disrupt the SSH process while it is active shows a larger amount of disk usage to handle excess load and packet losses.

The final result that will be discussed is for AAKR Model 2, which used signals selected from the memory and CPU usage related metrics. As stated previously, this particular group performed the poorest at detecting many intrusions tested. The results shown in Fig. 4 are for the "Processes-running" signal.

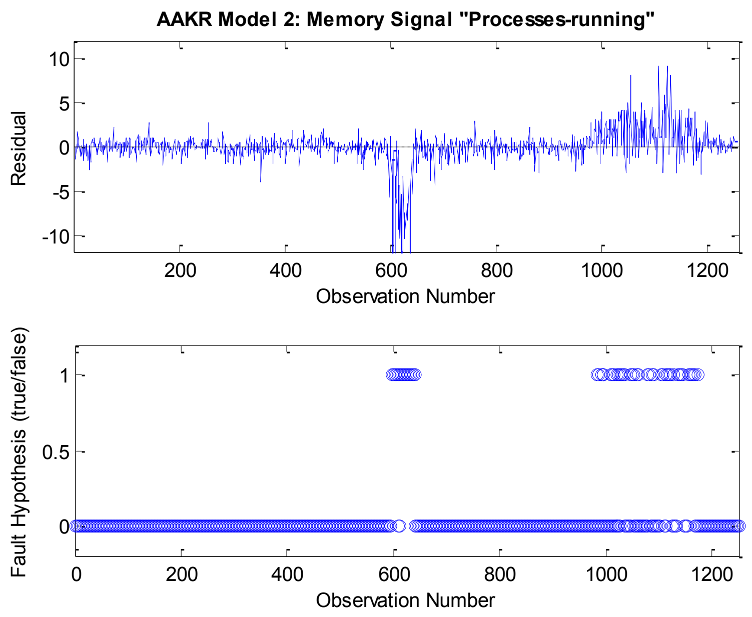

Figure 4. AAKR Model 2: Memory Usage Signal "Processes-running"

In the previous figure, it is immediately noticed that only the SSH fuzzer and exfiltration activities alter the behavior of the residuals. The other four intrusions that were tested are indistinguishable from normal behavior. This is also evident when the residuals in Fig. 4 are compared the previous two figures. In fact, the results shown in Fig. 4 were seen in nearly all signals for this particular grouping for both model types. The main takeaway in showing a signal grouping that did not catch all intrusions is that first, it is unreasonable to assume or expect every signal in a system to react to all known or unknown exploits. Second, because almost all residuals showed the same trends for specific intrusions, these features could be added to an attack dictionary for added prognostic security. Another interesting result from this group is that all memory and CPU usage signals take longer to settle back to normal behavior after the information theft attack than what was seen for all other groups. Listed previously, this attack was ended at observation 1100; all other residuals for the remaining groups took much less time than the memory/CPU group did to settle back to normal behavior. This behavior would also be the same if an attacker uploaded various files to the target machine. Considering this, the trend in these residuals could be used to verify that an inside attacker or disgruntled employee has stolen various files from the system. Also, there may be specific exploits developed in the future that would not be detected by TCP/IP signals, but would be easily detected by the memory/CPU signals. Finally, if a specific exploit was not able to be detected by any signal type or SPRT test, then a machine learning application would not be appropriate and some other detection method must be employed.

5 CONCLUSIONS

In conclusion, both the AAKR and AAMSET models presented were useful in from a cyber-security approach because both model types were able to generate residuals in all signal groups that showed clear indicators of many different attacks. Also, all of these different attack types induced minor to major changes in signal/residual behavior, and were able to be accurately quantified by both the mean and variance shift SPRTs. From a cyber-security standpoint for a nuclear power plant or SCADA network related industry, these results show that various sources of telemetry collected from either system or network traffic related metrics are quite useful for detection of known and new attacks. Also, because many attackers often spend days to weeks in host/network discovery related activities; these actions may also be detected by the methods presented as attacks that cause even small changes in residual behavior can be detected. Next, if monitoring small groups of signals is preferred, then attack types that generate or disrupt the network traffic will be easily detected when using TCP/IP telemetry. The models that use disk usage metrics were just as effective at detection of all attack types and would also enhance overall security. Models that use memory or CPU usage signals by themselves would not be effective in detecting most traffic related attacks. However, an interesting result is that Group 1, which used a combination of all signals from all sources, performed just as well at detecting the intrusions as the disk usage or TCP/IP groups. This result shows how combining signals that by themselves may not be effective for detecting all intrusions with other signal classes can enhance overall prognostic security. Using these methods with a dictionary IDS would result in a hybrid approach to security that would be able to detect known attacks, while also bringing the benefit of being able to detect "zero-day" attacks.

The final conclusion of our investigations in using these data-driven methods, combined with a SPRT for anomaly detection, can be a very valuable and useful technique for prognostic security purposes. Given that variants of the methods presented here have already seen commercial success, implementation or adaptation of these methods to enhance overall plant security could easily be achieved. It is noted that these behavior-based methods are meant to augment and not replace conventional dictionary-based detection in industry or business critical SCADA networks. In future work, we plan to extend the behavior-based techniques presented here that have shown high success for SCADA network security, to other enterprise systems and networks. Finally, the SCADA test bed will be adapted to drive a realistic simulation of a nuclear power plant or related industrial process to offer further validation to the efficacy of the methods when applied to cyber-security.

6 ACKNOWLEDGMENTS

The authors would like to thank Oracle Corporation for funding and support during Phase 1 and the current Phase 2 of this cyber-security research.

7 REFERENCES

1. V. Jyothsna and V. Prasad, "A Review of Anomaly based Intrusion Detection Systems", International Journal of Computer Applications, Vol. 28, No. 7 (2011).
2. A. Patel, Q. Qassim and C. Willis, "A survey of intrusion detection and prevention systems", Information Management Computer Security, 18(4), pp. 270:277 (2010).
3. D. Vinchurkar and A. Reshamwala, "A Review of Intrusion Detection System Using Neural Network and Machine Learning Technique", IJESIT, Vol. 1, Issue 2, Nov. (2012).
4. S. Han, M. Xie, H. Chen and Y. Ling, "Intrusion Detection in Cyber-Physical Systems: Techniques and Challenges", IEEE Systems Journal, Vol. 8, No. 4, Dec. (2014).

5. D. Garvey and J W. Hines, "The Development of a Process and Equipment Monitoring (PEM) Toolbox and its Application to Sensor Calibration Monitoring", *Quality and Reliability Engineering International*, **22**, 1-13, 2007.

6. A. V. Gribok, J. W. Hines, A. Urmanov and R. E. Uhrig, "Use of Kernel Based Techniques for Sensor Validation in Nuclear Power Plants", *Statistical Data Mining and Knowledge Discovery*, pp. 217-231, Chapman and Hall/CRC Press, 2004.

7. K. C. Gross, R. M. Singer, S. W. Wegerich, J. P. Herzog, R. VanAlstine, and F. Bockhorst, "Application of a Model-based Fault Detection System to Nuclear Plant Signals", Proc. 9th Intnl. Conf. On Intelligent Systems Applications to Power Systems, pp. 66-70, Seoul, Korea (July 6-10, 1997).

8. R.M. Singer, K. C. Gross, J. P. Herzog, R. W. King, and S. Wegerich, "Model-Based Nuclear Power Plant Monitoring and Fault Detection: Theoretical Foundations", Proc. 9th Intnl. Conf. On Intelligent Systems Applications to Power Systems, pp. 60-65, Seoul, Korea (July 6-10, 1997).

9. A. Wald, Sequential Analysis, J. Wiley & Sons, New York, 1947.

10. K. C. Gross, K. Vaidyanathan and S. Valiollahzadeh, "Advanced Pattern Recognition for Optimal Bandwidth and Power Utilization for Wireless Intelligent Motes for IoT Applications", World Computer Conf., 2016.

11. Kali Linux: https://www.kali.org

12. Armitage: https://www.offensive-security.com

13. CVE Database: https://www.cve.mitre.org

14. J. W. Hines, A. Usynin and S. Wegerich, "Autoassociative Model Input Variable Selection for Process Modeling, *58th Meeting of the Society of Machinery Failure Prevention Technology*, Virginia Beach, Virginia, April 26-30, 2004.