# Detection of Compromised Smart Grid Devices with Machine Learning and Convolution Techniques

Cengiz Kaygusuz, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac Cyber-Physical Systems Security Lab Department of Electrical & Computer Engineering, Florida International University 10555 West Flagler St. Miami, FL 33174 Email: {ckayg001, lbabu002, haksu, suluagac}@fiu.edu

Abstract—The smart grid concept has transformed the traditional power grid into a massive cyber-physical system that depends on advanced two-way communication infrastructure to integrate a myriad of different smart devices. While the introduction of the cyber component has made the grid much more flexible and efficient with so many smart devices, it also broadened the attack surface of the power grid. Particularly, compromised devices pose great danger to the healthy operations of the smart-grid. For instance, the attackers can control the devices to change the behaviour of the grid and can impact the measurements. In this paper, to detect such misbehaving malicious smart grid devices, we propose a machine learning and convolution-based classification framework. Our framework specifically utilizes system and library call lists at the kernel level of the operating system on both resourcelimited and resource-rich smart grid devices such as RTUs, PLCs, PMUs, and IEDs. Focusing on the types and other valuable features extracted from the system calls, the framework can successfully identify malicious smart-grid devices. In order to test the efficacy of the proposed framework, we built a representative testbed conforming to the IEC-61850 protocol suite and evaluated its performance with different system calls. The proposed framework in different evaluation scenarios yields very high accuracy (avg. 91%) which reveals that the framework is effective to overcome compromised smart grid devices problem.

Index Terms—Smart Grid, Compromised Devices, Cybersecurity, Machine Learning, Call List.

#### I. INTRODUCTION

The ability to sense and react to what is happening in the power grid by smart devices has revolutionized the power industry. By measuring the grid parameters, smart grid devices are able to control the electrical grid much more safely and efficiently than ever before [1]. Indeed, the introduction of the smart devices into the decade-old power grid definitely is a giant step that modernize the traditional grid; however, it also brings challenging security problems that are critical to tackle [2].

One of the most critical security problems in the power domain involves compromised smart grid devices. Compromising the smart devices such as sensors that measure the behavior of the power grid or controllers that either directly or indirectly controlling the behavior of the grid can have dire consequences: For instance, a sensor supplying false information may cause the control device to raise the voltage, possibly overloading the grid. Similarly, a malicious activity on a control device may accomplish the same hazard directly, making electricity unavailable. To ensure a healthy supply of such a critical resource, it must be ensured the smart grid devices on the grid must behave as expected and provide healthy operations.

In this paper, we propose a new framework to detect compromised smart devices in a smart grid environment. Specifically, the framework extracts statistics of system and library calls at the kernel level in the operating system which is subsequently fed into a machine-learning based classification model and convolution process. Analyzing the detailed metrics of how two call lists differ on type, length, distribution, and ordering, the proposed framework is able to identify benign devices from the compromised ones in all the evaluated cases. In addition, the framework obtains high accuracy when applying it on the data gathered from a representative testbed of smart grid devices conforming to IEC61850 protocol suite.

Our key contributions are listed as follows:

- We propose a detection framework that combines information extracted from system and library call lists (type of calls, length of call lists, and ordering of the calls), convolution, and machine learning algorithms to identify compromised smart grid devices based on the devices' behavior.
- We propose an adversary model that considers three fundamental malicious activities stemming from compromised devices: direct grid control, indirect grid control, and surveillance activity from attackers.
- We demonstrate the efficacy of the proposed framework by evaluating 5 different realistic cases that specify how behaviour of authentic and compromised devices can differ in the smart grid.
- Finally, we obtained high accuracy (91% average) on the detection of compromised smart grid devices for all the different analyzed cases.

The remainder of the paper is structured as follows: Section II presents the related work. Section III discusses the smart grid context and explicitly defines the adversary model and the problem being solved. Section IV describes the analysis of call list patterns. Section V discusses how to make a distinction for each case. Section VI evaluates the performance of the framework and finally, Section VII concludes the paper and discusses future work.

## II. RELATED WORK

Existing works investigating the security of smart grid and industrial control systems (ICS) mostly deal with identifying counterfeit devices in the supply chain domain as the main focus for the compromised devices. In [3] and [4], the authors offer different approaches for detecting fake electronic parts. Outbound beaconing, intelligent secure packaging, and better tracking systems are some of the countermeasures that are proposed to fight against counterfeiting on the supply chain side [5]. Common theme of these research is a focus on hardware level detection. In a similar fashion, authors in [6] dealt with detecting counterfeit devices on software layer using statistical correlation. Simulating three different attack models on a smart grid testbed, the authors compared the data gathered from devices to ground truth. A hand-picked threshold value were used to decide whether the device was compromised.

On the other hand, the analysis of network traffic with the intention of classifying device behavior is a well studied subject.Ahmed et al. presented a plethora of statistical and machine learning methods for behavioral network traffic analysis [7]. Specifically in ICS, the authors of [8] are using network traffic to identify counterfeit ICS devices.

The techniques that are employed in this paper are intersecting with some anomaly detection approaches. Existing data mining approaches has been reviewed by Agrawal et al. [9]. In the smart grid context, authors in [10] use a neural network model to detect malicious voltage control actions. In [11], researchers apply a rule based detection mechanism to detect attacks in smart grid.

Difference from existing work- Our work differs from the existing research as follows: The proposed framework operates on software level (at the kernel), and works for devices both in the supply chain as well as outside of it. Compared to approaches that solely focuses on the analysis of network data, the use of system and library call data offers a unique insight on the behavior of devices and is immune to random factors that affects the network state. To the best of our knowledge, this is the first study to provide a comprehensive framework with machine-learning and convolution techniques for the analysis of system and function call lists in the context of smart grid.

### III. SMART GRID CONTEXT AND ADVERSARY MODEL

This section discusses the adversary model focusing on the compromised device problem and the representation of the behavior of smart grid devices as a state machine.



Figure 1: A partial state machine representing reactive nature of an IED. In case IED senses a high voltage, it does necessary computations and routine calls to adjust the voltage; and in case of a frequency anomaly, the device, then, adjusts frequency. This behavior could be generalized to other smart grid devices.

#### A. Adversary Model

As discussed earlier, the adversary model in this work deals with an unauthorized control of the smart grid devices via compromising. In case of such a malicious event, the activity of a malicious actor could be categorized in three distinct classes:

- Direct grid control with specific commands: A compromised command & control device, such as an IED, may allow the attacker to issue commands directly to affect the state of the grid.
- Indirect grid control via fake measurements: A compromised sensor may send fake measurements to indirectly exert control over the smart grid.
- Surveillance of sensitive data: A compromised device may allow sensitive and confidential data and measurements to be gathered from the devices.

## B. Characterization of Smart Grid Device Processes

Majority of the critical field devices that are utilized in the grid include RTUs, PLCs, PMUs, and IEDs. In this work, the proposed framework specifically focus on these field devices:

- Remote Terminal Unit (RTU): Monitors relevant parameters about a system of subject and transmits this data to a central unit. This is an example of a *resource limited* device, where memory and computation power is rather limited.
- Programmable Logic Circuit (PLC): Directly controls actuators to manipulate physical phenomena in the grid.
- **Phasor Measurement Unit (PMU):** Measures electrical waves in an electrical grid.
- Intelligent Electornic Devices (IED): Receives data from sensors and issues control commands to regulate the grid. IED is an example of a *resource-rich* device, where in contrast to resource-limited devices, memory and computation power is abundant.

A key observation about these devices is that they are reactive: they respond to the events they receive in the smart grid in a deterministic fashion. Figure 1 illustrates this behavior with a state machine representation.

Moreover, most of the computation nowadays is done through programs that run on operating systems. A program must interact with the operating system to utilize system's



Figure 2: A state machine showing the initial, the benign computing, and the malicious (compromised) computation states.

resources through libraries that directly or indirectly use a standard library, such as allocating memory and sending a packet over the network to communicate with other computers. By obtaining system and/or library call traces over time, it is possible to identify which state the computing unit was operating on by analyzing the call list [12]. In other words, a computing unit responds to an event by a series of computation of which leaves a deterministic trace of system and library calls.

## C. Problem Definition

After defining the adversary model and characterizing the smart grid devices in terms of state machines, here we introduce another state machine representation which serves as the explicit definition of the compromised behavior detection problem.

Let us consider the following state machine, which is a simplified version of the earlier state machine and shown in Figure 2, where

- $q_I$  is the idle state,
- *q<sub>B</sub>* is the *benign computation mode*, e.g., expected activity. This is the state which the intended computation takes place; for example, a sensor might receive a timed IRQ to gather the readings and send it to a central unit,
- *q<sub>M</sub>* is the *compromised mode*, e.g., unexpected activity. The device could be doing anything here, e.g. poisoning the sensor measurements or sending harmful control commands as discussed earlier.

The goal of this work is to identify whether the monitored device has ever operated in state  $q_M$ . With the assumption that every described state emits a deterministic trace of system and library call lists, it is possible to construct a framework that can identify whether a given device is compromised. Finally, it is assumed that benign and malicious activities do not produce identical call list patterns.

## IV. ANALYSIS OF CALL LIST PATTERNS

To effectively utilize call lists as a discriminatory point between computing states, it is necessary to define what constitutes a call list and examine various measures of how two call lists are different. In this section, we define the call list and introduce the set of measures that are used in this work. Afterwards, the cases, which are based on the values of the defined metrics may take, are discussed.

#### A. Definitions

A Call List  $L_S$  is a finite sequential list of system or library calls when a computing unit finishes its operation in an arbitrary state  $q_S$ .

$$SD\left(\begin{vmatrix} malloc \\ malloc \\ free \\ free \end{vmatrix}, \begin{vmatrix} malloc \\ free \end{vmatrix}\right) = 0 \tag{1}$$

Set Distance  $SD(L_1, L_2)$  is the measure of how two call lists are different according to the type of calls they inhibit. Let A be the set of calls in  $L_1$ , and B the set of calls in  $L_2$ . The function  $SD(L_1, L_2)$  is simply the number of *unique* elements (cardinality) that is contained in A, but not in B. Formally stated,  $SD(L_1, L_2) = |A - B|$ . Note that  $SD(L_1, L_2) \neq SD(L_2, L_1)$ . Equation 1 gives an example where SD = 0.

$$LD\left(\begin{array}{c} |malloc|\\ malloc|\\ malloc|\\ malloc|\\ free \\ free \\ \end{array}\right) = 0 \tag{2}$$

**Length Distance**  $LD(L_1, L_2)$  is simply the difference of number of system calls contained by two call lists. LD = 0indicates two call lists are of the same length, while  $LD \neq 0$ indicates one list is longer than another by given amount, without specifying which one it is. Equation 2 gives an example where LD = 0. Note that  $LD(L_1, L_2) = LD(L_2, L_1)$ .

$$ED\left(\begin{vmatrix} malloc\\malloc\\free\\free\end{vmatrix}, \begin{vmatrix} malloc\\free\\free\end{vmatrix}\right) = 0$$
(3)

**Euclidean Distance**  $ED(L_1, L_2)$  is a measurement unit that intermixes both type and length difference between two call lists.  $v_{L_i}$  is an N dimensional vector where each dimension is mapped to total number of calls made to that particular system or library function belonging to call list  $L_i$ . With this definition,  $ED(L_1, L_2)$  is simply equal to  $|v_{L_1} - v_{L_2}|$ , or  $ED(L_1, L_2) = |v_{L_1} - v_{L_2}|$ . Equation 3 gives an example where ED = 0.

$$HD\left(\begin{vmatrix} malloc\\malloc\\free\\free\\free\end{vmatrix}, \begin{vmatrix} malloc\\free\\free\end{vmatrix}\right) = 2$$
(4)

**Hamming Distance**  $HD(L_1, L_2)$  is simply the number of operations required to be undertaken in order to make two call lists identical. Note that  $HD(L_1, L_2) = 0$  implies two lists are identical. Equation 4 gives an example where HD = 2.

#### B. Call List Difference Cases

The following is the list of all the identified critical points. These are also utilized in the performance evaluations in Section VI.

1)  $SD(L_M, L_C) > 0$ : Malicious state makes a call that is not contained in the benign state.



Figure 3: Overview of the framework. Using the harvested system or library call list, summary statistics and activity signal values are computed, which subsequently fed into a Support Vector Machine (SVM) to make a decision.

- 2)  $SD(L_C, L_M) < 0$ : Benign state makes a call that is not contained in the malicious state. Subsequent cases assume  $SD(L_C, L_M) = SD(L_M, L_C) = 0$
- LD(L<sub>C</sub>, L<sub>M</sub>) ≠ 0: Two call lists inhibit same type of calls, but differing in length. Subsequent cases assume LD(L<sub>C</sub>, L<sub>M</sub>) = 0
- 4)  $ED(L_C, L_M) \neq 0$ : Two call lists are of the same length and contains the same type of calls, but their internal distribution is different. Next case assumes  $ED(L_C, L_M) = 0$
- 5)  $HD(L_C, L_M) \neq 0$  Two call lists are of the same length, contains same type of calls, their internal distribution is the same but their order is not identical.

## V. OVERVIEW OF THE FRAMEWORK

In this section, details of the proposed framework is introduced. As shown in Figure 3, the framework utilizes a classifier model to make a decision whether or not the observed device has been exhibiting malicious activity (i.e., compromised device behaviour). The feature set contains the following classes:

- 1) Total number of calls made for each call type,
- 2) Average number of calls for each call type,
- 3) A value derived from activity signal.

As the collected behavioral data exhibits linearly separable features, a binary Support Vector Machine (SVM) classifier with a linear kernel is used.

Activity signal is the measure of how the order of calls in the observed list is in compliance with ground truth. Two call lists cannot be separated using summary statistics if two call lists differ only in the order of calls made. To address this challenge, Algorithms 1, 2, and 3 are proposed to be used in succession. Figure 4 gives an overview on the data flow in the context of activity signal.

The main idea is to use a classification model that predicts the next sequence of calls by inspecting the previous list of calls. If the predictions are true, it is inferred the call list is exhibiting expected patterns. On the other hand, if the predictions are failing, it is inferred the call list is exhibiting unexpected patterns; thus, the framework concludes the malicious activity is present. To quantify the mis-prediction ratio, a cascade of convolution and max-pooling operations are employed in the framework.

Upon receiving the call list, the first step is to pre-process the call list through an operation called *bucketing*. Algorithm 1 describes the computation required for this operation.

Algorithm 1: Bucketing						
	<b>input</b> : system or library call list					
	output: bucketed call list					
1	begin					
2	$S_{bucket} \leftarrow \text{configure}()$					
3	$S_{window} \leftarrow \text{configure}()$					
4	$T \leftarrow configure()$					
5	for $i \leftarrow 1$ to input.length - $S_{bucket}$ do					
6	for $j \leftarrow 1$ to $S_{bucket}$ - 1 do					
7	$R[i][j] \leftarrow input[i + j - 1]$					
8	$\begin{bmatrix} \mathbf{R}[\mathbf{i}].\mathbf{last} \leftarrow \mathbf{input}[\mathbf{i} + S_{bucket} - 1] \end{bmatrix}$					
9	return R					

Algorithm	2:	Constructing	Raw	Prediction	Signal

	input : bucketed call list
	output: raw prediction signal
1	begin
2	$S_{bucket} \leftarrow \text{configure}()$
3	$S_{window} \leftarrow \text{configure}()$
4	$T \leftarrow configure()$
5	for $i \leftarrow 1$ to R.length do
6	$C_{target} \leftarrow \mathbf{R}[\mathbf{i}].$ last
7	$C_{predict} \leftarrow \text{predict}(R[i])$
8	if $C_{predict} = C_{target}$ then
9	$P[i] \leftarrow 0$
0	else
1	$ $ P[i] $\leftarrow$ 1
2	return P

Over a sliding window of a configured size called *bucket length*, each call is mapped into one feature column, with the last one being the call the framework is trying to predict. The experimental results indicated a bucket length of 32 calls is an acceptable trade-off between run-time performance and accuracy.

After the bucketing procedure, the pre-processed data is fed into the prediction signal generator. Algorithm 2 describes the computation required to accomplish this operation. For each entry resulting in the dataset after bucketing operation, the predictor yields 0 if it was correctly predicted, 1 if the prediction failed. The resulting array of binary values is named as *activity signal*. For this process, the framework is utilizing random forest machine learning algorithm as the predictor as it provides the best performance.

Lastly, Algorithm 3 reduces the binary array obtained from Algorithm 2 to a single integer value using a cascade of convolution and max-pooling. The convolution kernel is simply sums all the values in the sliding window. Maxpooling picks the maximum element over a sliding window. Since max-pooling procedure uses non-overlapping windows, the data size is reduced by a factor of *window size* each time this operation is conducted. The resulting value is aimed to be higher in unexpected activity, and lower in expected activity. The cascade of convolution and max-pooling idea originates from convolutional neural networks (CNN), where a much more complex architecture involving these techniques



Figure 4: Demonstration of how data is transformed when computing the activity signal value. The received call list is first pre-processed through a procedure called *bucketing*. The pre-processed data is then fed into the classification model, which emits a 0 for correct and 1 for incorrect prediction. Resulting array of binary values is then sum-convoluted with a kernel size of 100. The convoluted values are then max-pooled with a non-overlapping sliding window of size 100. The output of the max-pooling is again processed with another round of convolution and max-pooling, which is continued until a single integer is obtained.





Figure 5: Accuracy of the framework applied individually to each case as defined in Section IV-B. The framework was able to exploit call list differences in all the cases. The red line indicates the average accuracy of 99%.

are used for image recognition tasks with success [13].

#### VI. PERFORMANCE EVALUATION

To test the overall classifier against the cases identified in Section IV, the state machine depicted in Figure 7 was utilized to generate data representing the behavior of authentic and compromised devices. Each execution of the state machine on the smart grid device is called an *experiment*. This state machine operates on *events*. Given an event, the smart grid device makes a transition to the state concerned, emits a call list, and returns to the idle state.



Figure 6: Results on data gathered from resource rich (RR) devices and resource limited (RL) devices. The framework accurately identifies compromised devices when looking solely at library calls, which indicate one of the two data sources may contain discriminatory information while the other one does not. The red line indicates the average accuracy of 91.25%



Figure 7: State machine representation of the test cases on a smart grid device.

The smart grid device's state machine operated on the following principles:

- The machine starts at  $q_I$ .
- At each turn, the machine transitions to either state  $q_C$  or  $q_M$ , with state transition probabilities  $p_C$  and  $p_M$ .
- At state  $q_C$  and  $q_M$ , the machine outputs a list of predetermined call list and transitions back into  $q_I$ .

- The total amount of events to be run in an experiment is randomly chosen according to a Gaussian distribution of mean of 10000 and standard deviation of 3000 events. A probability distribution is used to introduce variance in total number of calls. Mean and standard deviation numbers are picked by trial and error to ensure amount of data doesn't affect the outcome of the experiments.
- After the total amount of events are processed, the machine transitions to terminal state  $q_T$  and halts without producing any call trace.

For each case to be mentioned, there are a total of 60 experiments:

- 30 experiments run with  $p_C = 1$  and  $p_M = 0$ , and the resulting data are assumed to be coming from authentic devices.
- 30 experiments run with  $p_C = 0.99$  and  $p_M = 0.01$ .

The machine learning classification model (i.e., random forest) that is dealing with the generation of activity signal is trained on one single instance of the authentic call list. Then, the decision model is trained with 2/3 of the experiments, and the remaining 1/3 is used for testing.

The results of aforementioned setting is presented in Figure 5. In cases 1 through 5, the framework was able to exploit differences in call lists to accurately identify authentic devices from compromised ones. In case 6, the framework did as good as randomly guessing the device authenticity as benign and malicious computation emit completely identical call lists.

The framework is further tested on the dataset obtained in our previous study [6]. The data was obtained using a representative smart grid implemented by utilizing open source IEC61850 library *libiec61850*. The dataset consists of data obtained from resource-rich devices as well as resourcelimited devices after emulating three explicit attacks: information leakage, measurement poisoning, and saving data into the device memory to be sent later to attackers. Devices are also run without emulating any malicious activity, resulting in call lists data that is utilized as ground truth. As with previous test, 2/3 of the dataset is utilized for training, and remaining 1/3 is used for testing.

The results are shown in Figure 6. The algorithm was able to make good use of both system and library call lists in resource rich devices. Though, it was only able to obtain a high accuracy using only system calls in resource-limited devices. This fact indicates one of the two data sources may contain discriminatory information while the other one does not.

#### VII. CONCLUSIONS AND FUTURE WORK

Compromised devices pose great danger to the healthy operations of the smart grid. The attackers may utilize compromised devices to change the behaviour of the grid and modify critical measurements. In this paper, we proposed a novel detection framework that combines information extracted from system and library call lists, convolution, and machine learning algorithms to detect compromised smart grid devices. The performance of the proposed framework on a realistic smart-grid testbed conforming to the IEC-61850 protocol suite was evaluated on 5 different realistic cases. The test cases specified how behaviour of authentic and compromised devices could differ in the smart grid. The evaluation results demonstrated that the proposed framework can perform with very high accuracy (average 91%) on the detection of compromised smart grid devices. Although the proposed framework yielded highly accurate results, we ill consider other features from the devices to enhance to framework in our future work.

#### VIII. ACKNOWLEDGEMENTS

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000779.

#### REFERENCES

- X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid the new and improved power grid: A survey," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 944–980, Fourth 2012.
- [2] W. Wang and Z. Lu, "Cyber security in the smart grid: Survey and challenges," *Computer Networks*, vol. 57, no. 5, pp. 1344 – 1371, 2013. [Online]. Available: http://www.sciencedirect.com/ science/article/pii/S1389128613000042
- [3] S. Sathyanarayana, W. H. Robinson, and R. Beyah, "A network-based approach to counterfeit detection." in *IEEE International Conference* on Technologies for Homeland Security, ser. HST, Waltham, Massachusetts, 2013, NS.
- [4] A. Kanovsky, P. Spanik and M. Frivaldsky, "Detection of electronic counterfeit components," in 2015 16th Int. Scientific Conf. on Electric Power Engineering (EPE). Kouty nad Desnou: IEEE, May 2015, pp. 701 – 705.
- [5] D. van Opstal, U.S. Resilience Project, "Supgrid chain solutions for smart security: Buildply best practices." [Online]. ing on business Sep 2012. Available: http://usresilienceproject.org/wp-content/uploads/2014/09/ report-Supply Chain Solutions for Smart Grid Security.pdf
- [6] L. Babun, H. Aksu, and A. S. Uluagac, "Identifying counterfeit smart grid devices: A lightweight system level framework," in 2017 IEEE International Conference on Communications (ICC), May 2017, pp. 1–6.
- [7] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, no. Supplement C, pp. 19 – 31, 2016.
- [8] U. Guin, D. Forte, and M. Tehranipoor, "Anti-counterfeit techniques: From design to resign," in *Proceedings of the 2013 14th International Workshop on Microprocessor Test and Verification*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 89–94.
- [9] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, no. Supplement C, pp. 708 – 713, 2015, knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915023479
- [10] A. M. Kosek, "Contextual anomaly detection for cyber-physical security in smart grids based on an artificial neural network model," in 2016 Joint Workshop on Cyber- Physical Security and Resilience in Smart Grids (CPSR-SG), April 2016, pp. 1–6.
- [11] Y. Sun, X. Guan, T. Liu, and Y. Liu, "A cyber-physical monitoring system for attack detection in smart grid," in 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), April 2013, pp. 33–34.
- [12] M. M. Yasin and A. A. Awan, "A study of host-based ids using system calls," in 2004 International Networking and Communication Conference, June 2004, pp. 36–41.
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.