A System-level Behavioral Detection Framework for Compromised CPS Devices: Smart-Grid Case

LEONARDO BABUN, HIDAYET AKSU, and A. SELCUK ULUAGAC,
Florida International University

Cyber-Physical Systems (CPS) play a significant role in our critical infrastructure networks from power-distribution to utility networks. The emerging smart-grid concept is a compelling critical CPS infrastructure that relies on two-way communications between smart devices to increase efficiency, enhance reliability, and reduce costs. However, *compromised devices* in the smart grid poses several security challenges. Consequences of propagating fake data or stealing sensitive smart grid information via compromised devices are costly. Hence, early behavioral detection of compromised devices is critical for protecting the smart grid's components and data. To address these concerns, in this article, we introduce a novel and configurable system-level framework to identify compromised smart grid devices. The framework combines system and function call tracing techniques with signal processing and statistical analysis to detect compromised devices based on their behavioral characteristics. We measure the efficacy of our framework with a realistic smart grid substation testbed that includes both resource-limited and resource-rich devices. In total, using our framework, we analyze six different types of compromised device scenarios with different resources and attack payloads. To the best of our knowledge, the proposed framework is the first in detecting compromised CPS smart grid devices with system and function-level call tracing techniques. The experimental results reveal an excellent rate for the detection of compromised devices. Specifically, performance metrics include accuracy values between 95% and 99% for the different attack scenarios. Finally, the performance analysis demonstrates that the use of the proposed framework has minimal overhead on the smart grid devices' computing resources.

CCS Concepts: • **Security and privacy → Systems security**; **Distributed systems security**;

Additional Key Words and Phrases: CPS security, smart grid security, compromised devices, IEC61850, system calls, function calls

ACM Reference format:
Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2019. A System-level Behavioral Detection Framework for Compromised CPS Devices: Smart-Grid Case. *ACM Trans. Cyber-Phys. Syst.* 4, 2, Article 16 (November 2019), 28 pages.
https://doi.org/10.1145/3355300

This material is partially supported by the U.S. Department of Energy under Award Number DE-OE0000779 and by the U.S. National Science Foundation under Award Number NSF-1663051. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.
Authors' addresses: L. Babun, H. Aksu, and A. S. Uluagac, Electrical and Computer Engineering Department, College of Engineering and Computing, Florida International University, 10555 West Flagler St., Miami, FL 33174; emails: {lbabu002, haksu, suluagac}@fiu.edu.

2378-962X/2019/11-ART16 $15.00
https://doi.org/10.1145/3355300

1 INTRODUCTION

Critical infrastructure networks such as utility, production, and distribution systems are pillars of any nation and economy. They depend on intelligent and advanced Cyber-Physical Systems (CPS) to guarantee the efficient and reliable delivery of the data generated within these networks. These vital delivery systems have recently been going through a massive effort to modernize their CPS infrastructure.

In the specific case of the power grid, a substantial effort has already been made to modernize the traditional decade-old grid to the next generation of technology, i.e., *smart grid*. The core concept of the smart grid relies on the integration of the underlying electrical distribution with two-way communications capabilities between the smart CPS devices in the grid. The uses of CPS devices in the grid allows new functionalities and state-of-the-art computing systems for the smart grid infrastructure over the traditional power grid [84]. Nonetheless, new security concerns stem from the use of CPS devices by the modern power grid.

Indeed, with all its dependency upon device operations and communications, the smart grid is highly vulnerable to any security risk stemming from devices. Notably, the use of compromised devices can wreak havoc on the smart grid's critical functionalities [18, 63] and can cause catastrophic consequences to the integrity of the smart grid data and operations. Recent examples like the Stuxnet and Sandworm worm attacks [21, 27, 68] have proven that compromised devices represent a serious threat to the smart grid. Specifically, in the case of Stuxnet, the worm first targeted computers controlling Programmable Logic Controllers (PLCs)), to then change the configuration of the PLCs and cause the uranium centrifuges to behave erratically [50]. The same way, in the case of Sandworm, the attack first targeted computing systems using the BlackEnergy Trojan [44] to gain control over Remote Terminal Units (RTUs) and substation breakers to cause power blackouts [39]. Due to these real attacks, understanding the behavior of the smart devices, particularly the compromised ones, has become more critical than ever. Several government agencies focus their efforts to protect the critical infrastructure using behavioral-based approaches [61].

In this work, we propose a configurable system-level framework to detect compromised devices performing unauthorized operations inside the smart grid [6, 45]. Specifically, the proposed framework utilizes system and function call tracing techniques, signal processing, and statistical analysis to detect compromised devices based on their unexpected behavior. To test our framework, we designed a realistic representative smart grid substation testbed in which generic CPS devices performed essential operations conforming to the International Electrotechnical Commission 61850 (IEC61850[1]) [33–36] protocol suite. The proposed testbed includes both resource-limited (e.g., RTUs, PLCs, and resource-rich (e.g., Phasor Measurement Units (PMUs) and Intelligent Electronic Devices (IEDs)) CPS devices. In the testbed, the devices use open-source *libiec*61850 libraries [57] to exchange smart grid time-critical messages using the GOOSE format [10].

In addition, the adversary model complies with the security requirements specified by the standardization organizations [79] for the smart grid. In total, we consider six different types of compromised devices defined by different combinations of device computing resources and attack payloads.

Finally, we evaluate the performance of our framework by detecting and analyzing behavioral differences between compromised and ground-truth devices using three different detection methods. Experimental results demonstrate that the proposed framework achieves an excellent detection rate. Performance metrics reveal accuracy values between 95% and 99% for the different types of devices and detection methods analyzed. Additionally, detailed performance analysis shows

[1]IEC61850 is a protocol suite that defines the communication standards for electrical substation automation systems [10].

minimum overhead on the use of the smart grid devices' computing resources (i.e., CPU and memory). On average, memory and CPU utilization does not increase more than 0.03% and 1.9%, respectively.

Contributions: The contributions of this work are as follows:

(1) We designed a configurable system-level framework that combines system and function call tracing techniques with signal processing and statistical analysis to detect compromised smart grid devices. To the best of our knowledge, this is the first work that utilizes these techniques in detecting compromised devices in the smart grid.
(2) To test the efficacy of our framework, we designed a realistic smart grid substation testbed that included both resource-limited and resource-rich devices. These devices followed a GOOSE publisher-subscriber communication model using open-source *libiec*61850 libraries. The proposed testbed represents a valuable configurable benchmark for this, and other research works on CPS security via behavioral analysis.
(3) In the adversary model, we considered six different types of compromised devices with different computational resources and attack payloads.
(4) We evaluated the performance of our framework by detecting behavioral differences between the compromised device and ground-truth devices. We obtained accuracy results over 95% and precision results over 93% for all the different attacks scenarios and types of devices analyzed. These metrics demonstrated that the proposed framework is highly effective to recognize compromised smart grid devices using behavioral analysis.
(5) Finally, our analysis shows that the proposed framework does not represent a significant overhead in terms of computing resources.

Organization: The remaining of the article is organized as follows. Section 2 presents the background information and some critical implementation assumptions. Then, in Section 3, we describe the attacker model. Then, in Section 4, we detail the architecture of the proposed framework. In Section 5, we analyze and discuss experimental results, performance metrics, and benefits of our work. Finally, Section 6 presents the related work, and Section 7 concludes the article.

2 BACKGROUND AND ASSUMPTIONS

In this section, we provide insights into the behavioral analysis of the smart grid devices at the system level.

2.1 Overview of System-level Smart Grid Substation Architecture

The National Institute of Standards and Technologies (NIST) defines the smart grid as a set of seven different interconnected domains [62]. Specifically, two of these domains are responsible for the generation and transmission of electricity, while the other four provide business, operations, and customer support. Finally, at the center of the smart grid architecture, the distribution domain (i.e., smart grid substations) acts as a communication and control hub for the entire infrastructure, which makes it especially attractive to cyberattackers [84].

In Figure 1, we present a simplified version of the smart grid distribution domain architecture. Here, three main operation layers can be highlighted [37, 40, 85]:

- *Process Level*: permits the data acquisition and control at the lowest level of the smart grid substation architecture. The devices at the process level (i.e., merging units) extract state information from sensors, transducers, and actuators and deliver command controls from the upper layers.

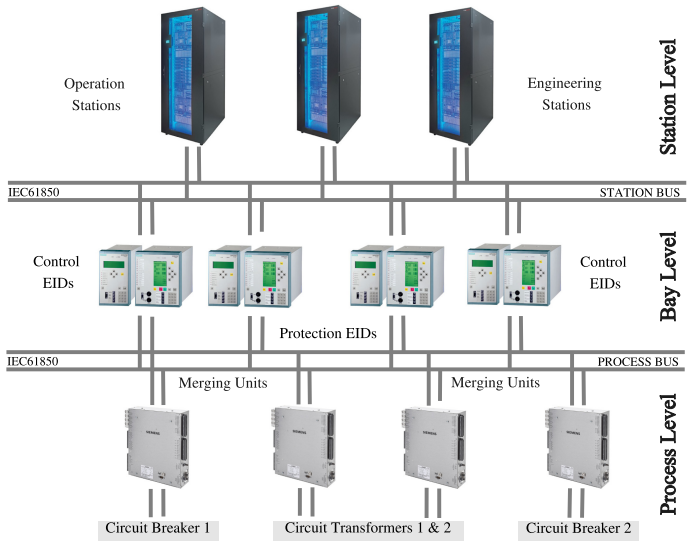


Fig. 1. System-level interaction of smart grid substation devices. The two-way communications under protocol suite IEC61850 can be established both horizontally (between devices from the same level) and vertically (between devices from different levels).

- *Bay Level*: permits the two-way communication between the process level and the upper operation layers of the smart grid substations. Here, Industrial Ethernet switches interconnect different control and protection EIDs to allow the following: (1) protection and control of the data exchanged between bay level and upper and lower layers and (2) protection of the data exchanged between devices located inside the bay level.
- *Station Level*: provides user interfaces and enable applications for engineering and control of the lower layers. Here we can highlight operations from the communication system, the time synchronization system, the substation data collection and control, and servers and workstations.

The IEC61850 protocol suite enables the real-time communications between devices from different substations levels (vertical communications) and devices within the same level (horizontal communications) using Manufacturing Message Specification (MMS), Generic Object Oriented Substation Events (GOOSE), and Sampled Measured Values (SMV) messages [10, 70]. Specifically, this standard includes many underlying protocol stacks to support and monitor a variety of time-critical services. Indeed, IEC61850 supports real-time operations, abstracts services, and interoperability between devices used in energy automation [35, 36].

2.1.1 Behavioral Analysis of Substation Smart Grid Devices. For this work, we focus on the behavioral characteristics of the smart grid substation devices while they communicate and perform either intra- or extra-level operations (i.e., horizontal and/or vertical communications) in the smart grid substation. We define *behavioral characteristics of devices at the system level* as the effect of the device's substations activities on the device's kernel. Let assume that there is a device O performing control operation at the Bay level. These operations can be represented as the set O_P, where

$$O_P = \{O_{P_0}, O_{P_1}, O_{P_2}, \ldots O_{P_N}\}, \tag{1}$$

then, we define the *system-level behavioral characteristic* of O as the function set BC due to the reflection of O_P at the device's kernel level [66, 87], that is,

$$BC = f_{kernel}(\{O_P\}). \tag{2}$$

In all the cases, we characterize the devices' kernel activity while the devices perform their regular smart grid substation operations. *Indeed, utilizing BC for the compromised device classification allows for a proper generalization of our framework so the proposed solution can also be successfully applied in other CPS domains outside the smart grid.*

2.2 Genuine Smart Grid Devices

We consider a smart grid device as *genuine* when no hardware nor software alteration or tampering has been performed on the device before, during, or after the manufacturing process. To further characterize and identify genuine devices, we define the parameter Index of Likeness (ILI). The ILI computes the similarity between individual operations O_i performed by a single device while executing a specific task T in different time intervals. Similar modeling approaches have been utilized in the literature to characterize CPS [13, 52]. The universe of operations performed by a device to complete a task T at time instant $t = 0$ can be defined as

$$T(t = 0) = \left\{ \cup_{i=0}^{\infty} O_i : \exists O_i \in T \right\}, \tag{3}$$

and the value of ILI for different t can then be expressed as

$$\rho_{ILI_t ILI_{t+i}} = \frac{\sum O_t O_{t+i} - n\overline{O_t O_{t+i}}}{n s_{O_t} s_{O_{t+i}}}, \tag{4}$$

where O_t represents the set of operations O_i performed by the device to complete the task T at the time instant t and O_{t+i} represents the set of operations performed to complete the same task T at the time instant $t + i$. In the same equation, n represents the cardinality of O and s_{O_t} and $s_{O_{t+i}}$ represent the standard deviation of O.

Based on our model, a genuine or *ground-truth* smart grid device is expected to have a high value of ILI on average. This assumption has been supported in the literature by other research works that characterize Cyber-Physical Systems (CPS) devices (including smart grid devices) as highly deterministic systems [17]. In general, for processes running over time, ILI is expected to take values between 0 and 1: 0 is the result of entirely uncorrelated O_is and 1 is the result of remarkably high correlated O_is. For a more realistic analysis, our work considers some inherent level of randomness within the device operations. This assumption prevents two Os from being completely identical even if one same device performs similar tasks repeatedly over time.

2.2.1 Ground-truth Devices. In the context of this work, ground-truth devices constitute particular cases of devices that are known as genuine. We assume full availability to ground-truth devices from every device class present in the smart grid. The proposed framework utilizes these ground-truth devices during its learning process. In the next sections, we define the practical values of ILI that allow for the characterization of ground-truth devices.

2.3 Compromised Smart Grid Devices

The smart grid (and other CPS) devices can be compromised either directly and indirectly. The direct method occurs in cases where the devices are compromised during any of the steps of the supply chain process [4, 38] or via insiders by *directly* changing the configuration of the devices or their executing apps. Here, the attackers directly target the CPS devices without any other intermediate device. However, indirect methods are most commonly used and usually require initial access to the computing systems controlling the CPS devices in the network. Once the attacker

gains access to those computers, they can change the configuration and behavior of the edge devices [77].

We envision that the proposed framework can be utilized to detect compromised devices in both the supply chain and in the field. For that reason, our work considers that genuine devices can be compromised during any stage of the manufacturing and application process. Specifically for our analysis, *we consider a compromised smart grid device as a genuine device with some malicious function installed on it.* The malicious function can be due to compromised hardware or software component [42, 73]. Also, the malicious function is expected to change the basic operations of the genuine device. In general, this function can be injected before, during, or after the device's manufacturing process. In Listing 1, we show realistic samples of a compromised device due to code injection. In this specific example, the malicious functions aim to (1) cause degradation on the device's resources and (2) save critical data on a file to be sent later to attackers.

```
1  void stress_mem()
2  {
3      srand(time(NULL));
4      long size = rand()%2147483647;
5      malloc(size);
6  }
7
8  void save_and_send_later (GooseSubscriber subscriber)
9  {
10     FILE *f = fopen("/root/baduser/data.dat", "a");
11         fprintf(f, "%" PRIu64 "\n", GooseSubscriber_getCriticalValue(subscriber));
12         fclose(f);
13 }
```

Listing 1. Example of malicious code injected to compromised smart grid devices.

To further describe the compromised devices, we recall Equation (4). Here, the set of operations O is compromised with a malicious subset O_m executed to perform the malicious activity [43]. That is, for compromised devices, the malicious activity impacts the value of ILI by inserting malicious operations O_{m_i} to O. Such operations change the device's kernel behavior (Equation (2)) so additional function or system calls are generated (see Listings 2 and 3). In general, the set O_m is expected to follow certain statistical distribution as detailed later in our adversary model. Finally, for compromised devices, Equation (4) takes the form:

$$\rho_{ILI_t ILI_{t+i}} = \frac{\sum O_t O_{m_t} O_{t+i} O_{m_{t+i}} - n\overline{O_t O_{m_t} O_{t+i} O_{m_{t+i}}}}{n s_{O_t} s_{O_{m_t}} s_{O_{t+i}} s_{O_{m_{t+i}}}}, \tag{5}$$

where the term O_{m_t} represents the malicious operations executed at time t and $O_{m_{t+i}}$ represents the malicious operations executed at time $t + i$.

```
1  pthread_detach
2  malloc
3  malloc
4  free
5  free
6  signal
7  malloc
8  malloc
9  free
10 free
11 .
12 .
13 .
14 .
15 .
16 .
```

Listing 2. System calls extracted from a genuine device.

```
1  pthread_detach
2  malloc
3  malloc
4  malloc
5  malloc
6  open
7  free
8  free
9  signal
10 malloc
11 malloc
12 malloc
13 malloc
14 open
15 free
16 free
```

Listing 3. System calls extracted from a compromised device.

2.4 Challenges on the behavioral identification of smart grid devices

Our framework utilizes changes in kernel's behavioral patterns to identify compromised devices. There are three main architectural challenges that our framework needs to overcome:

(1) Challenge 1: *The device class needs to be considered.* Different types of devices are expected to have different behavior; however, similar devices can also behave differently based on their specific tasks. Such ambiguity can lead to mistakenly identify genuine devices as compromised. For that reason, our framework incorporates (1) device resources (e.g., CPU and memory), (2) type of device, and (3) device task context into the analysis.
(2) Challenge 2: *Device classes are very diverse.* Device class classification would represent an implementation challenge due to the high device diversity present in the smart grid [62]. Additionally, after the initial classification, the list of devices would need to be checked periodically due to possible changes in network topology or new devices added to the network.
(3) Challenge 3: *Smart grid devices operations are not fully deterministic.* OS operations possess some degree of randomness that reflects on the device operation list O. During the detection process, the framework needs to discriminate between additional operations present in the call lists due to legitimate random processes and real malicious activities.

2.5 Classes of Smart Grid Devices

For this work, we group the smart grid devices into different classes. Then, we expect that devices from different classes have different behavior. To correctly group the devices, we consider three main features that address the challenges above: device's computing resource availability, device's type, and device's task context.

Resource availability—we define two different types of devices based on the availability of their computing resources: *resource-rich* and *resource-limited* devices.

- *Resource-limited devices*: These devices have simple hardware and software architecture. They run with low-performance CPUs and have minimal memory capability. In general, the randomness of the resource-limited devices' kernel behavior highly depends on their software architecture [17]. Also, these devices are built to execute specific tasks inside the smart grid network. Some devices in this group are PLCs and RTUs.
- *Resource-rich devices*: These smart grid devices are close in configuration to full-capacity computers. They have a full Operating System (OS), faster multi-core processors, and significantly higher memory than the resource-limited devices. This type of devices executes specialized tasks inside the smart grid network. Some devices in this group are IEDs and PMUs.

Moreover, we group the devices depending on their specific application, brand, and model. For instance, PMUs from the same model and manufacturer can be grouped together while RTUs and PLCs are not considered of the same type. We consider this classification, because the devices from different classes have found to behave differently, even if they perform similar tasks.

Finally, the class-classification process of smart grid devices considers the device's task context. For our purposes, the task context involves the type of activity that the devices are performing and their specific logical location inside the smart grid network. That is, we consider that devices of the same type can behave differently if they are handling different types of data from different parts of the network.

In general, we consider that the devices perform similar and repetitive tasks over time [17]. Then, our framework takes advantage of this mode of operation to detect compromised devices based on changes in their expected behaviour.

2.6 Open-source Approach

Our smart grid testbed utilizes open-source *libiec*61850 libraries [57] to exchange smart grid time-critical messages using the GOOSE format [10]. The use of open-source software provides some additional design advantages: (1) Our solution is more flexible, (2) the framework is more open to customizations that translate on being highly configurable, and, finally, (3) our solution can be easily adapted to other open standards, which increases interoperability. Therefore, to keep the proposed framework open-source, we implemented our solutions on Linux-based systems. This approach is considered realistic, since a very high percentage of smart grid devices still utilize some variant of Unix-based OS [67]. We believe that, due to the open-sourced and configurable nature of our testbed, it constitutes an effective benchmark to test the performance of this and other security tools designed to protect the smart grid that follows the behavioral analysis.

2.7 Extracting Operations from Smart Grid Devices

We utilize system and function call tracing techniques to extract the set of individual operations O from the devices. These operations are analyzed while the devices perform specific smart grid tasks T. We combine function and system call analysis, so the device's activity is detailed from both kernel and application-level, which increases the robustness of the framework. For attackers trying to exploit the calls to stealth their activities, the inconsistencies between system and function calls triggered by the same process can also indicate the presence of malicious activities. We take advantage of the open-sourced Unix-based nature of our testbed to effectively utilize library interposition and ptrace as system and function call tracing techniques, respectively.

Tracing system calls with library interposition—We use dynamic library interposition (LI), since this is a general-purpose system call tracing method that can be applied to most C-compiled programs [54]. LI takes advantage of the use of a shared object defined inside the runtime library. This object is in charge of fetching the system calls at the kernel level. At runtime, LI hooks this shared object to intercept the calls and take control of the applications' behavior.

Tracing function calls with ptrace—At the user level, we use Process Trace (i.e., ptrace), a popular Unix-based tool to trace function calls. Ptrace uses an external process that acts as a parent for the C compiled program that wants to be traced. Once the external process attaches to its child, the parent application has full control of every time the traced application makes a function call.

Finally, for cases where the smart grid devices do not use Unix-based OS (e.g., Real-Time Operating System (RTOS)), similar approaches are utilized to trace the system and function calls. Similar hooking techniques are possible to use, because these other systems behave in similar ways as Linux, since they are also POSIX-compliant OS. In general, the tracing technique utilized for hooking into the system and function calls is a configurable feature that depends on every specific application [54].

3 ADVERSARY MODEL

Our adversary model considers, conforming to the NIST guidelines, three possible threats in the smart grid that are directly related to the use of compromised devices [60]:

(1) *Threat 1 (Information leakage)*: The compromised device opens additional communication channels to leak valuable smart grid information to the adversary (another untrusted insider or outsider) in real-time.

Table 1. Threats to the Smart Grid Devices

Adversary Model			
Name	*CPS Device resource availability*	*Computing resources impacted*	*Security services compromised*
CD_1	Limited	Memory, CPU, communications	Confidentiality
CD_2	Limited	Memory, CPU	Integrity
CD_3	Limited	Memory, CPU	Authenticity, confidentiality
CD_4	Rich	Memory, CPU, communications	Confidentiality
CD_5	Rich	Memory, CPU	Integrity
CD_6	Rich	Memory, CPU	Authenticity, confidentiality

(2) *Threat 2 (Measurement poisoning)*: The compromised device generates fake data that can be used to poison the real status of the smart grid.
(3) *Threat 3 (Store-and-send-later)*: The compromised device stores information in hidden files that are recovered later by an attacker.

Based on these three well-defined threats and considering both resource-limited and resource-rich smart grid devices, we further define six different types of compromised devices as part of the adversary model:

(1) *Compromised Device 1 (CD_1)*: The resource-limited device creates additional instances of the IEC61850 GOOSE publisher object and starts leaking information through unauthorized communication channels.
(2) *Compromised Device 2 (CD_2)*: The resource-limited device allocates small and unauthorized amounts of memory to create fake data and poison real measurements.
(3) *Compromised Device 3 (CD_3)*: The resource-limited device creates unauthorized hidden files to store critical information that is retrieved later by the attacker.
(4) *Compromised Device 4 (CD_4)*: The resource-rich device creates additional instances of the IEC61850 GOOSE subscriber object and starts leaking information through unauthorized communication channels.
(5) *Compromised Device 5 (CD_5)*: The resource-rich device allocates small and unauthorized amounts of memory to create fake data and poison real measurements.
(6) *Compromised Device 6 (CD_6)*: The resource-rich device creates unauthorized hidden files to store critical information that is retrieved later by the attacker.

A summary of the adversary model, its impact on device resources, and the targeted security services of such attacks in the smart grid infrastructure are given in Table 1.

We also assume that the compromised devices perpetrate their attacks following a Poisson distribution. Poisson allows for randomly and efficiently spacing the attacks and constitutes a valid model to emulate the randomness of such events [69].

The behavior of the compromised device is modeled as follows. Consider $t = [0, T]$, the communication interval between the two smart grid devices. The probability of having an attack from a compromised device $CD_i \in \{CD_1, CD_2, CD_3, CD_4, CD_5, CD_6\}$ can be expressed as follows:

$$P_{cd} = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k \in \mathbb{R}, \tag{6}$$

where λ is the average number of attacks in the interval of time t and k is the total number of attacks in the same interval.

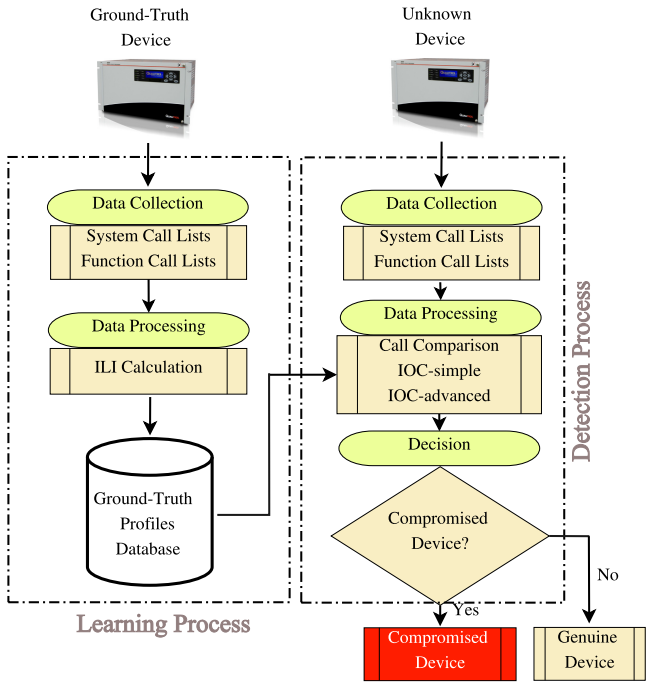


Fig. 2. Configurable framework proposed to monitor and detect compromised smart grid devices. The learning process creates signatures based on ground-truth devices that are utilized later to decide on potentially compromised devices.

4 OVERVIEW OF THE PROPOSED FRAMEWORK

In this section, we describe the proposed framework to detect compromised devices in the smart grid. Also, we present the details of our detection approaches and decision algorithm. Figure 2 depicts the general architecture of the framework. As discussed before, the main goal of the framework is to decide if an unknown smart grid device is genuine or compromised [51]. For this work, the term *unknown* refers to the level of uncertainty regarding the smart grid device being compromised or not. Initially, as part of the *learning process*, a ground-truth device from a specific device class is evaluated to generate its corresponding device-class signature or Ground-Truth Profile (GTP). This signature contains behavioral profiling information from the device and is utilized to decide whether an unknown device from the same class is genuine or compromised. Once the signature is obtained, it is stored in the *Ground-Truth Profiles Database*. In our implementation, we define a separate service to execute the learning process. Such a service separation permits the generation of new signatures every time that new devices join the network. Also, an independent learning process guarantees the replacement of old signatures every time that known devices assume new roles in the smart grid network.

The second part of the framework (also known as *detection process*) starts by extracting a similar profiling signature from the unknown device. Here, we assume that we have enough information to classify the device into some specific device class. Then, three different detection methods are applied to compare and correlate the unknown signature to the corresponding GTP from a similar device class stored in the database. Comparison and correlation results are then used to remove uncertainty and decide if the unknown device has been compromised or not.

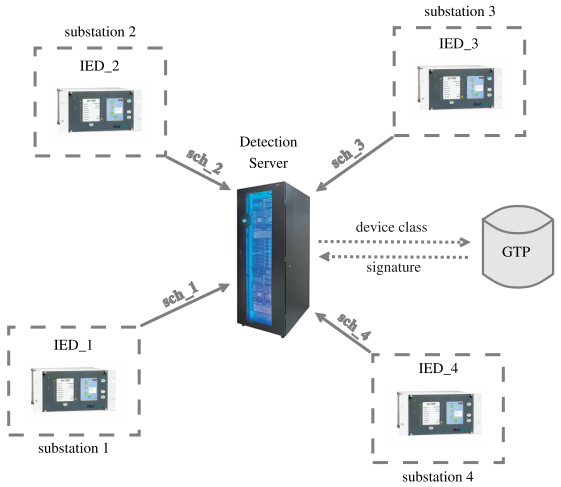


Fig. 3. Example implementation of the proposed framework.

We envision the proposed framework as a secured, centralized, and supervised agent virtually located inside the smart grid network. There are several advantages from this implementation model; first, our framework would be compliant with the security challenges of the smart grid [29, 30, 40]; second, a centralized solution represents a better option to monitor remotely located devices from different networks; and, third, a supervised agent allows for monitoring group of devices without degrading or interrupting critical tasks inside the smart grid. Figure 3 depicts a simplified implementation example of our framework. Here, IED devices exchange information between different substation level networks while a detection agent is monitoring them. Inside the devices, a lightweight scheduler (sch) runs parallel processes at the kernel level to hook into the devices' tasks and extract behavioral information. The collected information is sent to the server along with specific device class information using secure TCP-IEC61850 channels via either proxy or VPN-tunnel protected (depending on the smart grid device capability). Then, on the server-side, every scheduled action is processed using either priority or first-in-first-out– (FIFO) based queues. The priority is assigned depending on the device class and may also regulate the frequency of the scheduler's execution. For every detection process, the server executes queries to the GTP database using the device class ID and receiving the corresponding behavioral signature. Finally, the server correlates the scheduler data with the stored signature and decides on the devices as being compromised or not.

4.1 Probability of detecting a compromise smart grid device

In Section 3, we presented the probability of having a specific attack during a time interval t, considering the device is compromised. In this section, we formally describe the probability of detecting such attacks by using the proposed approach. To generalize, we consider that the statistical relationship between the two discrete random variables X and Y that represent the ground-truth signatures and the timed operation of the unknown smart grid devices follow a bi-variate distribution B. From here, we assume that the probability of having a particular specific sequence of calls in the GTP is $P(X)$. The same way, we assume a specific sequence of calls extracted from the unknown device with probability $P(Y)$.

When an attack occurs, and it is detected, the expected value $E(X)$ and $E(Y)$ of the random variables representing both the GTP and the unknown device call list are $P(X)$ and $P(Y)$, respectively.

From here, we can determine the variance V of the attack indicator[2] ϕ_x and ϕ_y from both the GTP and the unknown call lists as follows:

$$Var(\phi_X) = E(\phi_X^2) - E(\phi_X)^2 = P(X)(1 - P(X)), \tag{7}$$

$$Var(\phi_Y) = E(\phi_Y^2) - E(\phi_Y)^2 = P(Y)(1 - P(Y)). \tag{8}$$

We directly establish the statistical correlation between the random variable X and Y as the co-variance of these attack indicators,

$$\begin{aligned} Cov(\phi_X, \phi_Y) &= E(\phi_X \phi_Y) - E(\phi_X)E(\phi_Y), \\ &= E(\phi_{X \cap Y}) - E(\phi_X)E(\phi_Y), \\ &= P(X \cap Y) - P(X)P(Y). \end{aligned} \tag{9}$$

Then, we can define the correlation between ground-truth device signatures and the unknown smart grid devices based on the probability of detecting the attacks,

$$\begin{aligned} \rho(\phi_X, \phi_Y) &= \frac{P(X \cap Y) - P(X)P(Y)}{\sqrt{P(X)(1 - P(X))P(Y)(1 - P(Y))}}, \\ &= \frac{(P(X|Y) - 1)P(Y)}{\sqrt{P(X)(1 - P(X))P(Y)(1 - P(Y))}}, \end{aligned} \tag{10}$$

where $P(X|Y)$ represent the conditional probability of detecting an attack on a smart grid device after assuming a ground-truth signature from the same device class has been found. In general, we describe the successfulness of the proposed detection approach to be the jointly bi-variate variable (X_i, Y_j) with probability of occurrence $P(X_i > X_j | Y_i > Y_j)$ for any pair of calls i, j.

4.2 Learning Process

The primary goal of the learning process is to populate the Ground-Truth Profiles Database that contains all the GTPs from device classes in a specific smart grid network region. The execution of the learning process solves the first two architectural challenges of our framework described in Section 2. The learning process classifies the ground-truth devices into device classes and keeps the GTP database up-to-date. For every different class of devices, the learning process performs two specific tasks: (1) GTP data collection and (2) GTP data processing.

GTP Data Collection—This stage applies library interposition and ptrace to extract the lists of system and function calls, respectively. These operations are performed while the ground-truth devices execute regular smart grid substation tasks T. For every different device class, specific tasks are repeatedly executed over time while the framework hooks every iteration. As a result, for every iteration of T, the learning process generates new lists of system and function calls from the ground-truth device. In the end, the data collection process generates a set of system and function call lists. Every list contains detailed information about the specific operations that the devices executed at both the kernel and the user level in every different run of T.

GTP Data Processing—The data processing stage calculates the ILIs for every different ground-truth device class. The concept of ILI introduced in Section 2 evaluates how much deterministic the performance of a ground-truth device is over time. The more deterministic, the higher the ILI value and the more suitable the ground-truth device is to obtain its GTP. In total, the framework calculates two different values of ILIs, one from the set of system call lists and one from the set of function call lists, respectively. To successfully calculate the ILIs, the framework assigns a different weight δ_i to every different type of system or function call in the order that they appear. The

[2] An attack indicator is represented by the value of the random variable that would indicate the presence of an attack.

assignment of δ_i weights constitutes another configurable feature of our framework. This can be done randomly (the weights are considered normally distributed for simple processes where the different system or function calls have the same level of impact on the completion of the task T) or by following a specific assignment criterion (adaptive assignment). The adaptive assignment depends on the importance of the specific calls and the type of application that is being evaluated. As a result of the assignment step, the framework generates a random variable R that takes values between δ_{min} and δ_{max}. This variable describes the behavior of O for every different system or function call list. Finally, the framework calculates the ILIs using the Equation (4). In the end, the ILI values are compared against a configurable threshold σ. Initially, the framework selects an initial value for the threshold based on the device class, and then it continues adjusting this value until the average performance reaches the desired target value for that specific class. If both ILI values are above σ, then the GTP is accepted and stored in the database.

Equation (11) represents the general format of the GTP used in our work. The final profile contains information about the device class (DeC), the entire set of system and function call lists (SCL), and the threshold σ. At the end of the learning process, the Ground-Truth Profiles Database contains all the possible signatures that characterize the different device classes within a specific smart grid network region,

$$GTP = \{DeC, SCL, \sigma\}. \tag{11}$$

4.3 Detection Process

The main goal of the detection process is to use the profile information stored in the Ground-Truth Profiles Database to determine whether the unknown devices are being compromised. This process performs three main tasks:

(1) *Data collection*: This step follows almost the same sequence of operations detailed in the learning process. However, this time, the framework obtains the call lists from a single execution of T on the unknown devices from the smart grid networks. No T task is fixed for computing purposes nor is repeated over time.
(2) *Data processing*: This constitutes the core of the detection process. In this step, the framework combines three different detection mechanisms to detect compromised devices. The application of every detection approach is decided on-demand, which has a positive impact on reducing the total overhead introduced by our framework.
(3) *Decision*: Finally, the decision algorithm processes the results from data collection and processing to decide if the unknown device is genuine or compromised.

In the following, we provide details about the three detection mechanisms.

4.3.1 Detection Mechanisms. Our framework implements three different detection mechanisms. To utilize computational resources efficiently, the detection mechanisms are applied orderly on-demand. That means, our framework utilizes each detection approach in an ordered fashion, and it always uses the best effort to make a final decision by applying the minimum number of detection steps.

System and Function call list comparison—The simplest detection approach directly compares the SCL from the GTP to the system and function call lists extracted from the unknown device. The comparison schema considers the type and amount of system and function calls in both GTP and the newly extracted lists. This mechanism is implemented, as shown in Equation (12). Specifically, the comparison approach generates a *call vector* that contains the total number of different calls extracted from the unknown device of class c and normalized against the term $GTP(c; SCL)$.

Equation (12) details this process:

$$call\_vector = \left\{ \frac{unkc_0}{GTP_{SCL_0}}, \frac{unkc_1}{GTP_{SCL_1}}, \ldots, \frac{unkc_n}{GTP_{SCL_n}} \right\}, \tag{12}$$

where the term $unkc_0$ refers to the amount of system or function calls of type 0 extracted from the unknown device and the term GTP_{SCL_0} refers to the amount of system or function calls of type 0 extracted from the GTP of the same device class. As inferred from Equation (12), call vector's items of value 0 represent types of calls that are present in the GTP SCL but not in the lists of calls acquired from the unknown device. On the contrary, call vector's items of value ∞ represent calls extracted from the unknown device, but that cannot be found in the corresponding GTP. In general, the execution of this first detection approach is very light in terms of computing resources.

Index of Correlation Simple—A second detection mechanism calculates the statistical correlation between call lists from the unknown device and the GTP. In this case, in addition to the type and amount of calls, the framework considers the order in which these calls are being triggered. The result from calculating such statistical correlation is known as *Index of Correlation simple* (IOC-simple). IOC-simple is similar to the ILI value obtained during the learning process. The main difference between both is that IOC-simple first determines the statistical correlation between call lists from the unknown device and the corresponding GTP class. Here, an assignment criterion is also used to convert calls into specific δ_i values,

$$IOC - simple_{O_{GTP}, O_{unk}} = \frac{\sum o_{GTP} o_{unk} - n\overline{o_{GTP} o_{unk}}}{ns_{o_{GTP}} s_{o_{unk}}}, \tag{13}$$

where o_{GTP} represents the set of individual calls in the GTP and o_{unk} represents the set of individual calls extracted from the unknown device.

Index of Correlation Advanced—As mentioned in Section 2, one should not expect smart grid devices to perform operations in a completely deterministic pattern. This limitation exposes the third architectural challenge of our framework (see Section 2), since legitimate random operations can be mistaken as compromised behavior. To overcome this constraint, we further apply a more advanced IOC calculation (IOC-advanced). In IOC-advanced, our framework combines the values from O_{i_unk} to O_{i+h_unk} in O_{unk}. This operation results in a new random vector O'_{unk} smaller in size and with a lower random component. The index h represents the number of individual calls from the original list that are combined to create the new set O'_{unk}. This index value h is proportional to the amount of randomness that one intends to remove from the original O_{unk} and constitutes another configurable parameter in our framework.

ALGORITHM 1: Steps for the detection and decision processes.

```
compromised ← 0
UNK(DeC_unk, SCL_unk) ← unknown device profile
GTP(DeC_gtp, SCL, σ) ← GTPs from Database
Detection:
if Exists DeC_gtp & DeC_unk == DeC_gtp then
    Calculate IOC
end if
Decision:
if IOC < β then
    compromised ← 1
end if
```

4.4 Decision Process

The final step of our framework is the decision process. In this step, our framework compares results from the three detection mechanisms against a threshold β to decide if the unknown smart grid device is compromised or not. The value of β depends on the device class, and it is always a function of the threshold σ determined during the training process and stored in the GTP. The relationship between σ and β values depends on the targeted accuracy performance for every device class. In general, for devices with a higher deterministic behavioral pattern, a higher value of β is recommended. This design approach reduces the chances of false negatives during the decision process. However, for devices with lower deterministic behavior, a lower value of β may be sufficient to reduce false negatives. Finally, note that this decision threshold is also configurable. The initial value of β for every device class can be adjusted to an optimal in real-time and while the framework monitors the devices in the field. In the next section, we analyze practical values of β for different types of device classes.

Finally, Algorithm 1 details the detection-decision process of the proposed framework. In lines 1, 2, and 3 the variables *compromised*, *UNK*, and *GTP* are initialized with 0, the profile of the unknown device, and all the signatures from the database, respectively. Then, if a signature of the unknown device's class exists (Line 4), then the values *IOC* (simple and advanced) are calculated in Line 5. Finally, if the value of *IOC* is lower than the threshold β, then the device is decided as *compromised*.

5 PERFORMANCE ANALYSIS AND DISCUSSION

In this section, we analyze the performance of the proposed framework. In all the cases, we obtain the results after averaging 30 different runs of all the covered scenarios. The scenarios include six different types of attackers as a result of the combination of three different threats and two different types of devices based on their computational resources, as described in Section 2. Also, we assume that the devices are correctly grouped based on their type. Moreover, we measure the accuracy of our framework with accuracy, precision, recall, and specificity metrics. Finally, we evaluate the performance of the proposed framework in terms of its overhead (e.g., CPU utilization, memory usage, and execution time).

5.1 Evaluation Methodology with a Realistic Testbed

Our framework considers a realistic scenario from a smart grid substation. The testbed's configuration includes a publisher-subscriber two-way communications configuration that sends and receives IEC61850-compliant GOOSE messages [10]. For this purpose, we utilize an open-source version of IEC61850 [57] protocol running on Linux-based systems. Our resource-limited devices (i.e., GOOSE publishers) run on a Raspberry Pi 2B, using Advanced RISC Machine (ARM) 32 bits architecture with limited memory and CPU. However, the resource-rich devices (i.e., GOOSE subscribers) run on a Linux Ubuntu 14.04 system with a more powerful CPU and higher memory configuration. Finally, we utilize two different hooking techniques: ptrace (that performs function call tracing) and library interposition (that performs system call tracing). In our configuration, the publishers open the communication session and wait for the subscribers to connect. Once the devices create and open the communication sockets, the publishers start sending GOOSE messages to the subscribers every one second for a total time interval t of 60 s. After the t seconds, the devices close their communication channels. For every compromised device, the malicious threat is active n times during the communication sessions as described in the adversary model (see Section 3). Finally, as detailed in Table 1, compromised devices CD_1, CD_2, and CD_3 correspond to resource-limited devices of any class that have been compromised with Threats 1, 2, and 3, respectively (see

Table 2. Normalized Rate of the System and Function Calls Captured after Using Our Framework to Detect Compromised *Resource-limited Devices* (e.g., RTUs, PLCs): Calls Due to Malicious Activities Are Grayed

Call Tracing Technique	Type of Call	Genuine	CD_1	CD_2	CD_3
ptrace	*brk*	~1	~1	6.7	~1
	clone	~1	12.5	~1	~1
	close	~1	~1	~1	3.2
	fstat64	~1	~1	~1	8.8
	lseek	~1	~1	~1	~1
	mmap2	~1	2.4	4.4	2.4
	mprotect	~1	2.8	1.1	1
	munmap	~1	~1	2	13
	open	~1	~1	~1	5
	rt_sigprocmask	~1	8.7	0.3	0.3
	rt_sigaction	~1	~1	3	3
Interposition	*close*	~1	~1	~1	~1
	free	~1	3.2	~1	~1
	malloc	~1	3.3	~1	~1
	memcpy	~1	~1	~1	~1
	memset	~1	~1	~1	~1
	mmap	~1	12.5	~1	~1
	mprotect	~1	12.5	~1	~1
	pthread_create	~1	12.5	~1	~1
	sendto	~1	4.3	~1	~1
	signal	~1	24	~1	~1
	socket	~1	~1	~1	~1
	usleep	~1	3.5	~1	~1

Section 3) and compromised devices CD_4, CD_5, and CD_6 correspond to resource-rich devices of any class that have been compromised with Threats 1, 2, and 3, respectively. Despite that the initial application of our testbed was intended to evaluate the performance of the proposed framework in realistic scenarios, we believe that, due to its open-source and configurable nature, it can also be used as a benchmark to effectively evaluate the performance of other security tools applied to the smart grid.

5.2 Detection Performance

In the following, we detail the performance of our framework after applying the three detection mechanisms proposed in Section 4.

5.2.1 System and Function Call Lists Comparison. Tables 2 and 3 summarize some of the system and function calls captured from the resource-limited and the resource-rich devices, respectively. Columns *Genuine* and CD_i (i: 1 to 6) in both tables list the average rate of the system and function calls normalized against the GTP for genuine and compromised devices, respectively.

Values greater than ~1 (marked in gray) in columns CD_1 to CD_3 and CD_4 to CD_6 represent extra system or function call activity due to the presence of malicious operations. That is, extra call activity reveals the presence of malicious activity in the devices. One can notice that, by using ptrace, our framework identified all cases of compromised devices. However, in the case of library

Table 3. Normalized Rate of System and Function Calls Captured after Using Our Framework to Detect Compromised *Resource-rich Devices* (e.g., PMUs, IEDs): Calls Due to Malicious Activities Are Grayed

Call Tracing Technique	Type of Call	Genuine	CD_4	CD_5	CD_6
ptrace	*brk*	~1	~1	8.3	~1
	clone	~1	23	~1	~1
	close	~1	6.5	6.8	6.75
	fstat	~1	12	12.5	12.25
	mmap	~1	4.1	6.64	2.6
	mprotect	~1	3.4	~1	~1
	munmap	~1	23	26	24
	open	~1	6.5	6.75	6.8
	rt_sigaction	~1	8.3	~1	~1
Interposition	*free*	~1	15.6	~1	~1
	malloc	~1	15.6	~1	~1
	memcpy	~1	17.8	~1	~1
	memset	~1	24	~1	~1
	mmap	~1	24	~1	~1
	mprotect	~1	24	~1	~1
	pthread_create	~1	24	~1	~1
	$pthread_detach$	~1	24	~1	~1
	recvfrom	~1	15.7	~1	~1
	signal	~1	24	~1	~1
	socket	~1	24	~1	~1
	usleep	~1	15.7	~1	~1

interposition, only CD_1 and CD_4 were properly detected. Also, the reader can notice that in the case of genuine devices, the normalized rate values of system and function calls are very close to 1 in all the cases.

5.2.2 IOC-simple. The first detection approach could not identify Threats 2 and 3 when the framework utilized library interposition. To overcome this limitation, we applied our second detection mechanism, IOC-simple. As explained in Section 4, to utilize the framework efficiently, the framework applies the different detection approaches in an ordered fashion as needed.

Figure 4(a) shows the results after applying IOC-simple to system and function call lists from GTP and compromised devices. In this figure, *R-R* refers to resource-rich devices, and *R-L* refers to resource-limited devices.

The reader can observe that, by using ptrace, we obtain low IOC values (in the range of 0.15 to 0.35) between function call lists from GTP and compromised devices. By setting the correlation strength threshold to 0.6 (moderate to high correlation [69]), our framework detects all the cases of the compromised devices. For the case of library interposition, the framework performs very well for resource-rich compromised devices. However, for resource-limited compromised devices IOC-simple under-performs when the framework applies library interposition. In this particular case, IOC-simple from genuine devices falls under the threshold, triggering false positive results. We relate these results to higher random activity in the resource-limited compromised devices' kernel [16].

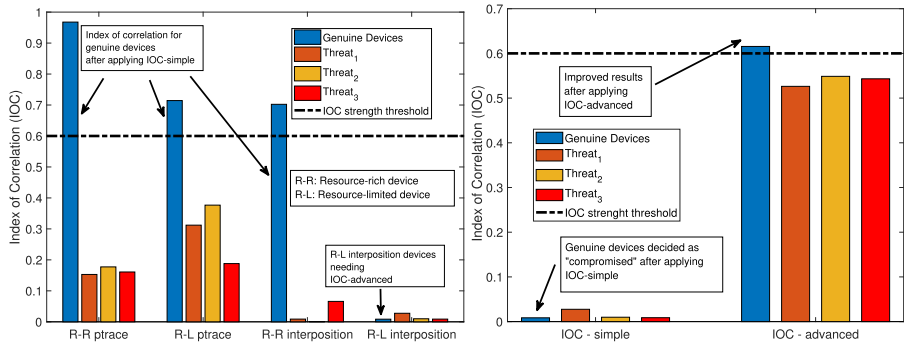


Fig. 4. Index of Correlation between GTP and unknown devices: (a) Resource-rich and resource-limited devices after applying our IOC-simple and (b) IOC-advanced results comparison between genuine and compromised resource-limited devices (using system call lists from library interposition only).

5.2.3 IOC-advanced. To overcome the previous limitation, we can apply the IOC-advanced technique. By using this approach, our framework can obtain new call lists with more deterministic behavior from the resource-limited devices and enhance the statistical correlation between these type of devices and their corresponding GTP. In Figure 4(b), the reader can observe how IOC values from resource-limited genuine devices overcome the threshold mark while the compromised devices are still under the borderline. There exists a tradeoff between the amount of randomness that can be removed from system call lists without impacting the decision process. If the value of h is too significant, then critical behavioral information can also be potentially removed from the call lists, limiting the performance of the decision algorithm in cases where tasks T are too simple.

5.3 Performance Metrics

To further measure the efficacy of our detection methods, we calculate the standard performance metrics of accuracy, recall, precision, and specificity. These metrics are defined in Equations (14), (15), (16), and (17):

$$A_{CC} = \frac{(T_P + T_N)}{(T_P + T_N + F_P + F_N)}, \tag{14}$$

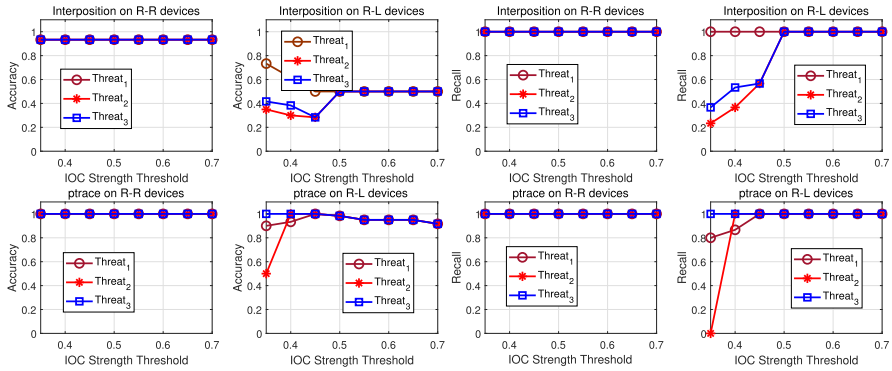
$$R_{EC} = \frac{T_P}{(T_P + F_N)}, \tag{15}$$

$$P_{REC} = \frac{T_P}{(T_P + F_P)}, \tag{16}$$

$$S_{pec} = \frac{T_N}{(T_N + F_P)}, \tag{17}$$

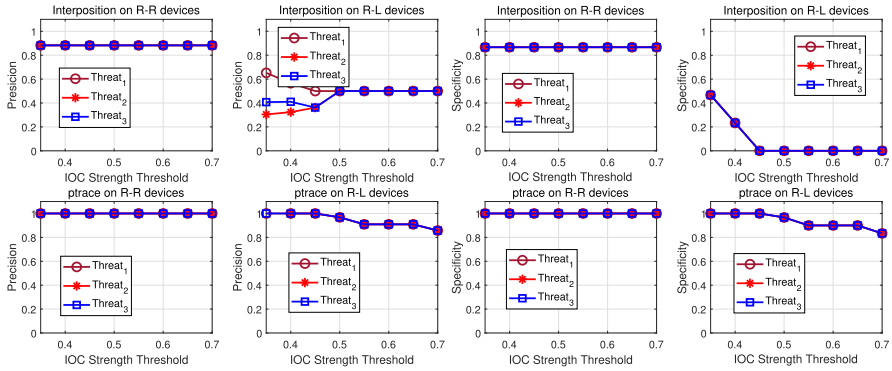
where T_P stands for true positive or the case where a compromised device is decided as compromised; T_N stands for true negative or the case where a genuine device is decided as genuine; F_P stands for false positive or the case where a genuine device is decided as compromised; and, finally, F_N stands for false negative or the case where a compromised device is decided as genuine. First, we evaluate the performance of our framework with IOC-simple. Then, the improved results are shown after applying IOC-advanced.

In Figure 5(a), we evaluate the overall accuracy of our detection techniques over the six different types of compromised devices. Since accuracy comprises T_P and T_N results, this metric describes



(a) Accuracy (b) Recall

Fig. 5. Figures compare the performance of the IOC-simple algorithm on six different types of compromised devices after using library interposition and ptrace: (a) Accuracy and (b) Recall.



(a) Precision (b) Specificity

Fig. 6. Figures compare the performance of the IOC-simple algorithm on six different types of compromised devices after using library interposition and ptrace: (a) Precision and (b) Specificity.

how well the framework can positively decide between genuine and compromised devices without errors. In general, for ptrace, our framework achieves an excellent accuracy performance (between 0.95 to 1) for all types of devices. However, this analysis also reveals the performance limitations of the framework for detecting resource-limited compromised devices in the case of library interposition (top-right case in Figure 5(a)). Here, the framework achieves a low accuracy value of 0.5.

In Figure 5(b), we evaluate the overall recall performance of the framework. In this case, recall metrics show how well our framework detects the six different types of compromised devices. Based on these results, the reader can observe that the framework achieves the maximum recall (maximum value of T_Ps) for the selected threshold $\beta = 0.6$. In the case of resource-rich devices, recall performance was high for all the threshold values. However, for resource-limited devices, we can notice low recall values for Threats 2 and 3 when the threshold values are under 0.6 for the case of library interposition.

Figure 6(a) depicts the precision evaluation. Precision values represent the statistical relationship between the number of successfully detected compromised devices against the number of times that the framework fails to correctly decide a device as genuine. By looking at the precision

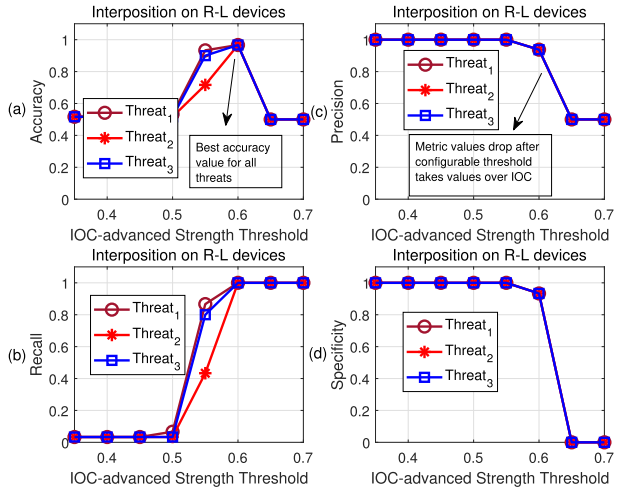


Fig. 7. Performance metrics after applying IOC-advanced for the detection of resource-limited devices when library interposition is utilized: (a) Accuracy, (b) Recall, (c) Precision, and (d) Specificity.

results, one can observe that our framework under-performs in the case of library interposition for resource-limited devices.

Finally, we utilize specificity metrics to evaluate the true negative rate, that is, how effectively our framework discriminates genuine devices. In Figure 6(b) (top right), one can observe that, for the case of resource-limited devices with library interposition, the framework achieves very low specificity. These results limit the application of IOC-simple to decide on this particular type of devices. Specificity value of 0 at β threshold between 0.45 and 0.7 demonstrates that a device was not correctly decided as genuine in this case. However, in all the remaining three cases, the framework performs very well.

By analyzing the results in Figures 5 and 6, one can compare the performance of the proposed framework on resource-limited and resource-rich devices for the two hooking techniques applied. Most evaluation metrics diminish their performance when the framework applies the IOC-simple algorithm to detect resource-limited devices using library interposition. These results reflect on the fact that for this type of devices, a more robust detection mechanism is necessary. To improve these results, we utilize the framework with the IOC-advanced algorithm. Figure 7 depicts the improvements in all the performance metrics after applying IOC-advanced for compromised resource-limited devices with library interposition. In this figure, one can observe that the correlation threshold of 0.6 provided the best results overall for this particular testbed. Also, the framework obtained significant improvements in accuracy and precision if compared with the case of IOC-simple (accuracy improved from 0.5 to 0.96, and precision improved from 0.5 to 0.93). Finally, recall metrics retained its high performance at the selected threshold value (recall = 1).

5.4 System Overhead

We expect our framework to perform with high accuracy and scalability without introducing too much overhead. Table 4 summarizes the average of system overhead on resource-limited and resource-rich devices. The metrics *RT*, *ST*, *UT*, *Mem*, and *CPU* correspond to the values of real-time, system-time, user-time, memory, and CPU, respectively. In this table, *NF* (No Framework) represents the case where devices were evaluated without applying our the proposed framework, and *WF* (With Framework) represents the cases where we evaluated the performance

Table 4. Average System Overhead on Resource-rich and Resource-limited Devices after Using the Framework

Metrics	*NF*		*WF*			
	value	value	*ptrace (%)*		*LI (%)*	
	R-R	*R-L*	*R-R*	*R-L*	*R-R*	*R-L*
RT (s)	60.00	60.11	0.05	3.8	0.01	0.1
ST (s)	0.49	3.60	8.1	3.6	10.2	5.5
UT (s)	0.31	0.49	16.1	0.31	6.4	2.0
Mem (KB)	1967.5	1827.5	1.1e-3	4.3e-5	3.0e-2	1.0e-3
CPU (%)	1	6.02	0	1.9	0	1

Table 5. Specification Values for Remote Terminal Unit RT2020 [31]

Item	*Specification Values*
Processor	Dual Core ARM A9 667 MHz
Dynamic Memory (RAM)	128 MB
Program Memory (Flash)	4 MB
Nonvolatile Memory	4 Mb
Real Time Clock Resolution	1 ms
Execution Cycle Time	≤100 ms

while applying the framework. Additionally, *LI* represents the cases where we applied library interposition. Finally, *R-R* refers to resource-rich devices, and *R-L* refers to resource-limited devices. Results in Table 4 demonstrate that the utilization of the detection framework does not introduce significant overhead on the devices. Particularly, in the case of resource-limited devices, the framework utilizes 0.03% more of memory (out of the total memory available in the devices) and 1.9% more of the CPU. For resource-rich devices, the framework utilizes 0.001% more of memory (out of the total memory available on the device) and an almost negligible amount of CPU. In summary, for both resource-limited and resource-rich devices, library interposition introduces the most overhead to the system. However, this overhead is considerably low if compared with similar applications proposed in the literature [23, 74].

To further study the impact of our framework, we analyzed this overhead considering a real resource-limited smart grid device. In Table 5, we summarize the main specifications of Remote Terminal Unit RT2020. Looking at Table 5, we can conclude that for the worst case of resource utilization (library interposition on a resource-limited device), the increment in execution time because of the use of our framework would only represent up to 2.3 cycle times. Additionally, our framework would only take 0.1% of the total memory of a real resource-limited smart grid device.

5.5 Benefits and Features

There are several benefits associated with the design of our framework:

(1) *Excellent detection rate*: The proposed framework demonstrated an excellent rate for the detection of compromised smart grid devices by combining three different detection methods: system and function call comparison, IOC-simple, and IOC-advanced.
(2) *Minimum overhead*: The proposed framework does not represent significant overhead on the use of computing resources.
(3) *Specific vs. generic solution*: The proposed framework is designed to address the specific problem of compromised smart grid device detection. The adversary and system model

proposed in this work follow the security requirements and architecture characteristics of the smart grid. However, the approaches proposed here for the detection of compromised smart grid devices are perfectly suitable for other CPS security domains outside the smart grid domain.

(4) *Comprehensive adversary model*: The adversary model used in this work considers both resource-limited and resource-rich compromised devices. Also, it combines three different threats affecting the smart grid.

(5) *Compromised device diversity*: Our framework is suitable for a great range of different compromised devices. The design of our system-level framework makes it also suitable for detecting hardware counterfeiting [1, 12, 25] as observed from the system level. System and function call comparison and statistical techniques are powerful tools capable of detecting changes in hardware and system configuration. This makes our framework an appealing solution to monitor and detect a wide range of different types of compromised devices.

6 RELATED WORK

In this section, we present the related work. There are several works studying security challenges in the smart grid [47, 72, 78, 81]. In general, cyberattacks against smart grid are categorized into four different groups: denial of services (DoS) attacks, malicious data injection attacks, traffic analysis attacks, and high-level application attacks [81]. In References [3, 9, 19, 82], the authors provide several examples of DoS attacks impacting different parts of the smart grid architecture. Most of these attacks are executed from compromised hosts, servers, and devices inside the smart grid.

Malicious data injection attacks are analyzed in References [19, 46, 71]. One compelling case is studied in Reference [88]. In this article, the authors analyze four different types of attacks in the state estimation process and examine the least-effort data injection attack to find the optimal attack vector.

In the case of traffic analysis attacks, authors in Reference [75] describe how an attacker can monitor and intercept the frequency and timing of transmitted messages to deduce information and user's behavior. In Reference [60], high-level application attacks are described as the way an attacker can disrupt the essential functions of a power system (i.e., state estimation and power flow measurement).

In general, these are all useful studies, but none of them directly covers the threat of compromised devices in the smart grid. Additionally, in cases where the attackers rely on the use and control of compromised smart grid devices to perform the attacks, only one type of smart grid threat was considered at a time. In this work, our adversary model considers a combination of three different threats impacting the smart grid combined with two different device resource availabilities.

Smart grid compromised device detection: In general, the topic of compromised devices has not been extensively studied in the literature. In most cases, researchers focus on proposing anomaly detection mechanisms [49] for different types of attacks in the smart grid [22, 28, 64, 76], without particularizing on the attack sources (e.g., compromised devices). In a few cases, however, the behavior of the smart grid device is considered. In Reference [86], the authors study the minimal number of compromised sensor that can be used to manipulate a given number of smart grid states effectively. Further, they consider the optimal PMU placement to defend against this type of data integrity attacks. Some works have been proposed in other CPS and industrial environments. In Reference [59], the authors propose a vector-valued model-based cumulative sum procedure to identify compromised sensors in CPS. Even though this work achieves promising results in simulation environments, its threat model only considers false data injection attacks. Also, no results are

shown on the overhead introduced to the CPS devices, essential to consider suitable security applications for real-time critical infrastructures like the smart grid. In a different approach, integrity measurement and attestation systems have been proposed to evaluate the integrity of applications in CPS and the Internet of Things (IoT) devices [11, 15, 55, 65]. Also, the authors of Reference [80] apply attestation approaches to detect comprised devices in the CPS. In this work, however, they utilize stimulant-response mechanisms to detect compromised devices based on their specific reaction to controlled inputs, which can also be impractical for the smart grid and results can depend on several undesired networks' and physical channels' dynamics. Other relevant works propose similar attestation approaches [14, 53] to detect attacks in CPS. However, these works focus on building models of the entire CPS network instead of focusing on individual devices, which impacts the overhead and the general performance of the proposed solutions. Finally, most of these works apply to Wireless Sensor Networks (WSN) and are not directly applicable to the smart grid domain. Finally, in more general approaches, some works proposes the use of data collected from devices to detect malicious operations or specific behavior [2, 5].

Intelligent, secure packaging, outbound beaconing, and better tracking systems are some of the countermeasures that are being proposed to prevent the introduction of compromised devices in the smart grid supply chain [18, 56, 83]. However, skilled attackers could have remote access to legitimate devices (e.g., RTUs, PMUs, and IEDs) outside the supply chain and create opportunities for tampering smart grid devices in the field.

Function and system call tracing techniques for security applications: Function and system call tracing techniques constitute a powerful method for regulating and monitoring applications behaviour [7, 8, 45], so they have been largely used in security applications [24]. System and function call tracing techniques can be found in applications like intrusion detection and confinement [41], binary detection of OS functions [32], sandboxing [48], and software portable packages [26]. Specifically, in Reference [20], the authors use system call tracing to implement intrusion detection systems (IDS). Also, in Reference [23] and Reference [58], the authors proposed anomaly detection mechanism based on information obtained from system calls behavior analysis. In these cases, the implementation of the security tools resulted too heavy in terms of system overhead. One similar application with improved system overhead can be found in Reference [74]. In this case, the proposed solution is required to run continuously and serves the purpose of complementing antivirus software.

Difference from existing work: *Our framework is different from other discussed solutions that, in most cases, focus on specific threats to the smart grid instead of considering multiple types of threats acting on different type of devices (e.g., resource-rich and resource-limited). As discussed, there are also cases where different approaches are used for the detection of compromised devices and/or monitoring application behavior. Only in a few of these cases, the solution is intended to be applied in the smart grid domain. In addition, to succeed, these solutions need to monitor constantly changing environments like network traffic and computational systems or need to challenge the devices with specific inputs to study their response, which constitutes a limitation in terms of system overhead, resource utilization, and real-time analysis. Differently, our framework has a simpler model and is lightweight in terms of system overhead while providing excellent detection rate of the compromised smart grid devices while they are performing typical real-time CPS operations.* Also, we propose a configurable framework for both the supply chain and the smart grid operation field that is envisioned friendly and adaptive enough to be easily applied either within supply chain testing scenarios and while the devices are performing real-time operations inside the smart grid infrastructure. Finally, our work can also complement the existing security mechanisms in the smart grid domain with its open-source and configurable nature.

7 CONCLUSIONS

The smart grid vision depends on the secure and reliable two-way communications between smart devices (e.g., IEDs, PLCs, PMUs). Nonetheless, compromised smart grid devices constitute a serious threat to a healthy and secure distribution of data in the grid. In this work, we designed a system-level configurable framework capable of monitoring and detecting compromised smart grid devices. Our framework combines system and function call tracing techniques (i.e., ptrace, library interposition), signal processing, and statistical analysis (basic and advanced) to detect compromised device behavior. To the best of our knowledge, this is the first work that utilizes these techniques in detecting compromised devices in the smart grid. Moreover, we evaluated the performance of our framework on six different types of compromised devices, conforming to realistic smart grid scenarios. Such devices exchanged smart grid GOOSE messages utilizing an open-source version of the IEC61850 protocol suite. Specifically, we analyzed the efficacy of our framework under six different adversarial settings affecting devices with different resource availability. Experimental results demonstrated that our framework successfully detects different types of compromised device behavior in a variety of different environments with high accuracy. Also, our performance analysis reveals that the use of the proposed detection framework yield minimal overhead on the smart grid devices' computing resources.

REFERENCES

[1] A. Kanovsky, P. Spanik, and M. Frivaldsky. 2015. Detection of electronic counterfeit components. In *Proceedings of the 2015 16th International Scientific Conference on Electric Power Engineering (EPE'15)*. IEEE, 701–705.

[2] H. Aksu, L. Babun, M. Conti, G. Tolomei, and A. S. Uluagac. 2018. Advertising in the IoT Era: Vision and challenges. *IEEE Commun. Mag.* 56, 11 (Nov. 2018), 138–144. DOI: https://doi.org/10.1109/MCOM.2017.1700871

[3] J. D. Ansilla, N. Vasudevan, J. JayachandraBensam, and J. D. Anunciya. 2015. Data security in smart grid with hardware implementation against DoS attacks. In *Proceedings of the 2015 International Conference on Circuits, Power and Computing Technologies (ICCPCT'15)*. 1–7. DOI: https://doi.org/10.1109/ICCPCT.2015.7159274

[4] Bjørn Egil Asbjørnslett. 2009. *Assessing the Vulnerability of Supply Chains*. Springer US, Boston, MA, 15–33. DOI: https://doi.org/10.1007/978-0-387-79934-6_2

[5] Leonardo Babun, Amit Kumar Sikder, Abbas Acar, and A. Selcuk Uluagac. 2018. IoTDots: A digital forensics framework for smart environments. *CoRR* abs/1809.00745 (2018). arxiv:1809.00745 http://arxiv.org/abs/1809.00745.

[6] Leonardo Babun, Hidayet Aksu, and Selcuk A. Uluagac. 2018. Detection of Counterfeit and Compromised Devices Using System and Function Call Tracing Techniques. Retrieved from http://www.freepatentsonline.com/10027697.html.

[7] Leonardo Babum, Hidayet Aksu, and Selcuk A. Uluagac. 2019. Method of Resource-limited Device and Device Class Identification Using System and Function Call Tracing Techniques, Performance, and Statistical Analysis. Retrieved from http://www.freepatentsonline.com/10242193.html.

[8] Reverend Bill Blunden. 2013. *The Rookit Arsenal: Escape and Evasion in the Dark Corners of the System* (2nd ed.). Cathleen Sether, Burlington, MA.

[9] N. Boumkheld, M. Ghogho, and M. El Koutbi. 2016. Intrusion detection system for the detection of blackhole attacks in a smart grid. In *Proceedings of the 2016 4th International Symposium on Computational and Business Intelligence (ISCBI'16)*. 108–111. DOI: https://doi.org/10.1109/ISCBI.2016.7743267

[10] C. Kriger, S. Behardien, and J. Retonda-Modiya. 2013. A detailed analysis of the GOOSE message structure in an IEC 61850 standard-based substation automation system. *Int. J. Comp. Comm.* 8, 5 (Oct. 2013), 708–721.

[11] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. 2018. Sensitive information tracking in commodity IoT. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security'18)*. USENIX Association,1687–1704.

[12] Ch. Wong and M. Wu. 2015. A study on PUF characteristics for counterfeit detection. In *Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP'15)*. IEEE, 1643–1647.

[13] Y. Chen, C. M. Poskitt, and J. Sun. 2018. Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP'18)*. 648–660. DOI: https://doi.org/10.1109/SP.2018.00016

[14] Y. Chen, C. M. Poskitt, and J. Sun. 2018. Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP'18)*. 648–660. DOI: https://doi.org/10.1109/SP.2018.00016

[15] J. Clemens, R. Pal, and B. Sherrell. 2018. Runtime state verification on resource-constrained platforms. In *Proceedings of the 2018 IEEE Military Communications Conference (MILCOM'18)*. 1–6. DOI: https://doi.org/10.1109/MILCOM.2018.8599862

[16] Henry Corrigan-Gibbs and Suman Jana. 2015. Recommendations for randomness in the operating system or, how to keep evil children out of your pool and other random facts. In *Proceedings of the 15th USENIX Conference on Hot Topics in Operating Systems (HOTOS'15)*. USENIX Association, 25–25.

[17] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah. 2016. Who's in control of your control system? Device fingerprinting for cyber-physical systems. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'16)*.

[18] D. van Opstal, U. S. Resilience Project. 2012. Supply Chain Solutions for Smart Grid Security: Building on Business Best Practices. Retrieved from http://usresilienceproject.org/wp-content/uploads/2014/09/report-Supply_Chain_Solutions_for_Smart_Grid_Security.pdf.

[19] Y. Deng and S. Shukla. 2012. Vulnerabilities and countermeasures - A survey on the cyber security issues in the transmission subsystem of a smart grid. *J. Cyber Secur. Mobil.* 1, 4 (2012), 251–276.

[20] E. Eskin, W. Lee, and S. J, Stolfo. 2001. Modeling system calls for intrusion detection with dynamic window sizes. In *Proceedings of the DARPA Information Survivability Conference & Exposition II, 2001 (DISCEX '01)*. IEEE, 165–171.

[21] European Network and Information Security Agency (enisa). 2012. Smart Grid Security. Annex II: Security Aspects of the Smart Grid. Retrieved from https://www.enisa.europa.eu/topics/critical-information-infrastructures-and-services/smart-grids/smart-grids-and-smart-metering/ENISA_Annex%20II%20-%20Security%20Aspects%20of%20Smart%20Grid.pdf.

[22] A. Farraj, E. Hammad, A. A. Daoud, and D. Kundur. 2016. A game-theoretic analysis of cyber switching attacks and mitigation in smart grid systems. *IEEE Trans. Smart Grid* 7, 4 (Jul. 2016), 1846–1855. DOI: https://doi.org/10.1109/TSG.2015.2440095

[23] Henry Hanping Feng, Oleg M. Kolesnikov, Prahlad Fogla, Wenke Lee, and Weibo Gong. 2003. Anomaly detection using call stack information. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP'03)*. IEEE Computer Society, 62–. http://dl.acm.org/citation.cfm?id=829515.830554

[24] Tal Garfinkel. 2003. Traps and pitfalls: Practical problems in system call interposition based security tools. In *Proceedings of the Network and Distributed Systems Security Symposium*. 163–176.

[25] Ujjwal Guin, Domenic Forte, and Mohammad Tehranipoor. 2013. Anti-counterfeit techniques: From design to resign. In *Proceedings of the 2013 14th International Workshop on Microprocessor Test and Verification*. IEEE Computer Society, 89–94. http://dx.doi.org/10.1109/MTV.2013.28

[26] P. J. Guo and D. Engler. 2011. CDE: Using system call interposition to automatically create portable software packages. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference (USENIXATC'11)*. USENIX Association, Berkeley, CA, 21–21.

[27] Aaron Hansen, Jason Staggs, and Sujeet Shenoi. 2017. Security analysis of an advanced metering infrastructure. *Int. J. Crit. Infrastruct. Protect.* 18, C (2017), 3–19. https://doi.org/10.1016/j.ijcip.2017.03.004

[28] J. Hao, R. J. Piechocki, D. Kaleshi, W. H. Chin, and Z. Fan. 2014. Optimal malicious attack construction and robust detection in smart grid cyber security analysis. In *Proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm'14)*. 836–841. DOI: https://doi.org/10.1109/SmartGridComm.2014.7007752

[29] D. He, S. Chan, and M. Guizani. 2017. Cyber security analysis and protection of wireless sensor networks for smart grid monitoring. *IEEE Wireless Commun.* 24, 6 (Dec. 2017), 98–103. DOI: https://doi.org/10.1109/MWC.2017.1600283WC

[30] D. He, S. Chan, and M. Guizani. 2017. Win-win security approaches for smart grid communications networks. *IEEE Netw.* 31, 6 (Nov. 2017), 122–128. DOI: https://doi.org/10.1109/MNET.2017.1700065

[31] Honeywell. 2014. RTU2020 Remote Terminal Unit Specifications. Retrieved from https://www.honeywellprocess.com/library/marketing/tech-specs/SC03-300-101-RTU-2020.pdf.

[32] G. Hunt and D. Brubacher. 1999. Detours: Binary interception of Win32 functions. In *Proceedings of the 3rd Conference on USENIX Windows NT Symposium - Volume 3 (WINSYM'99)*. USENIX Association, Berkeley, CA, 14–14. http://dl.acm.org/citation.cfm?id=1268427.1268441

[33] IEC 61850-1. 2003. Communication Networks and Systems in Substations Introduction and Overview. Retrieved from https://webstore.iec.ch/p-preview/info_iec61850-1%7Bed1.0%7Den.pdf.

[34] IEC 61850-7-2. 2003. Communication Networks and Systems in Substations—Basic Communication Structure for Substation and Feeder Equipment Abstract Communication Service Interface (ACSI). Retrieved from https://webstore.iec.ch/p-preview/info_iec61850-7-2%7Bed1.0%7Den.pdf.

[35] IEC 61850-8-1. 2003. Communication Nnetworks and Systems in Substations—Specific Communication Service Mapping (SCSM) Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3. Retrieved from https://webstore.iec.ch/p-preview/info_iec61850-8-1%7Bed1.0%7Den.pdf.

[36] IEC61850-7-1. 2003. Communication Networks and Systems for Power Utility Automation. Part 7-1: Basic Communication Structure—Principles and Models. Retrieved from https://webstore.iec.ch/publication/6014.

[37] D. M. E. Ingram, P. Schaub, R. R. Taylor, and D. A. Campbell. 2013. Performance analysis of IEC 61850 sampled value process bus networks. *IEEE Trans. Industr. Inf.* 9, 3 (Aug. 2013), 1445–1454. DOI: https://doi.org/10.1109/TII.2012.2228874

[38] Interos Solutions, Inc. 2018. Supply Chain Vulnerabilities from China in U.S. Federal Information and Communications Technology. Retrieved from https://www.uscc.gov/sites/default/files/Research/Interos_Supply%20Chain%20Vulnerabilities%20 from%20China%20in%20U.S.%20Federal%20ICT_final.pdf.

[39] J. Ellperin and A. Entous. 2016. Russian operation hacked a Vermont utility, showing risk to U.S. electrical grid security, officials say. Retrieved from https://www.washingtonpost.com/world/national-security/russian-hackers-penetrated-us-electricity-grid-through-a-utility-in-vermont/2016/12/30/8fc90cc4-ceec-11e6-b8a2-8c2a61b0436f_story.html?utm_term=.7445133366ca.

[40] P. Jafary, S. Repo, J. Seppälä, and H. Koivisto. 2017. Security and reliability analysis of a use case in smart grid substation automation systems. In *Proceedings of the 2017 IEEE International Conference on Industrial Technology (ICIT'17)*. 615–620. DOI: https://doi.org/10.1109/ICIT.2017.7915429

[41] K. Jain and R. Sekar. 1999. User-level infrastructure for system call interposition: A platform for intrusion detection and confinement. In *Proceedings of the Network and Distributed Systems Security Symposium*.

[42] K. Huang, J. M. Carulli, and Y. Makris. 2013. Counterfeit electronics: A rising threat in the semiconductor manufacturing industry. In *Proceedings of the IEEE International Test Conference (ITC'13)*. IEEE, 1–4.

[43] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur. 2016. Model-based security analysis of a water treatment system. In *Proceedings of the 2016 IEEE/ACM 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS'16)*. 22–28. DOI: https://doi.org/10.1109/SEsCPS.2016.012

[44] Kaspersky. 2016. BlackEnergy APT Attacks in Ukraine. Retrieved from https://usa.kaspersky.com/resource-center/threats/blackenergy.

[45] C. Kaygusuz, L. Babun, H. Aksu, and A. S. Uluagac. 2018. Detection of compromised smart grid devices with machine learning and convolution techniques. In *Proceedings of the 2018 IEEE International Conference on Communications (ICC'18)*. 1–6. DOI: https://doi.org/10.1109/ICC.2018.8423022

[46] K. Khanna, B. K. Panigrahi, and A. Joshi. 2016. Feasibility and mitigation of false data injection attacks in smart grid. In *Proceedings of the 2016 IEEE 6th International Conference on Power Systems (ICPS'16)*. 1–6. DOI: https://doi.org/10.1109/ICPES.2016.7584204

[47] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke. 2010. Smart-grid security issues. *IEEE Secur. Priv.* 8, 1 (Jan. 2010), 81–85. DOI: https://doi.org/10.1109/MSP.2010.49

[48] T. Kim and N. Zeldovich. 2013. Practical and effective sandboxing for non-root users. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference (USENIX ATC'13)*. USENIX Association, Berkeley, CA, 139–144. http://dl.acm.org/citation.cfm?id=2535461.2535478

[49] A. M. Kosek. 2016. Contextual anomaly detection for cyber-physical security in smart grids based on an artificial neural network model. In *Proceedings of the 2016 Joint Workshop on Cyber—Physical Security and Resilience in Smart Grids (CPSR-SG'16)*. 1–6. DOI: https://doi.org/10.1109/CPSRSG.2016.7684103

[50] D. Kushner. 2013. The real story of stuxnet. *IEEE Spectrum* 50, 3 (Mar. 2013), 48–53. DOI: https://doi.org/10.1109/MSPEC.2013.6471059

[51] L. Babun, H. Aksu, and A. S. Uluagac. 2017. Identifying counterfeit smart grid devices: A lightweight system level framework. In *Proceedings of the IEEE ICC International Conference on Communications*. IEEE, 7.

[52] L. D. Lago, O. Ferrante, R. Passerone, and A. Ferrari. 2018. Dependability assessment of SOA-based CPS with contracts and model-based fault injection. *IEEE Trans. Industr. Inf.* 14, 1 (Jan. 2018), 360–369. DOI: https://doi.org/10.1109/TII.2017.2689337

[53] M. LeMay, G. Gross, C. A. Gunter, and S. Garg. 2007. Unified architecture for large-scale attested metering. In *Proceedings of the 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. 115–115. DOI: https://doi.org/10.1109/HICSS.2007.586

[54] Juan Lopez, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2017. A survey on function and system call hooking approaches. *J. Hardw. Syst. Secur.* 1, 2 (1 Jun 2017), 114–136. DOI: https://doi.org/10.1007/s41635-017-0013-2

[55] Peter A. Loscocco, Perry W. Wilson, J. Aaron Pendergrass, and C. Durward McDonell. 2007. Linux kernel integrity measurement using contextual inspection. In *Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing (STC'07)*. ACM, New York, NY, 21–29. DOI: https://doi.org/10.1145/1314354.1314362

[56] M. Q. Saeed, Z. Bilal, and C. D. Walter. 2013. An NFC based onsumer-level counterfeit detection framework. In *Proceedings of the 2013 11th Annual International Conference on Privacy, Security and Trust (PST'13)*. IEEE, 135–142.
[57] M. Sillgith. 2016. Open source library for IEC 61850: Release 0.9. Retrieved from http://libiec61850.com/libiec61850/.
[58] Narcisa Andreea Milea, Siau Cheng Khoo, David Lo, and Cristian Pop. 2012. NORT: Runtime anomaly-based monitoring of malicious behavior for windows. In *Proceedings of the 2nd International Conference on Runtime Verification (RV'11)*. Springer-Verlag, Berlin, 115–130. DOI: https://doi.org/10.1007/978-3-642-29860-8_10
[59] C. Murguia and J. Ruths. 2016. CUSUM and chi-squared attack detection of compromised sensors. In *Proceedings of the 2016 IEEE Conference on Control Applications (CCA'16)*. 474–480. DOI: https://doi.org/10.1109/CCA.2016.7587875
[60] N. Komninos, E. Philippou, and A. Pitsillides. 2014. Survey in smart grid and smart home security: Issues, challenges and countermeasures. *IEEE Commun. Surv. Tutor.* 16, 4 (2014), 1933–1954. DOI: https://doi.org/10.1109/COMST.2014.2320093
[61] National Cybersecurity & Communications Integration Center (NCCIC), Department of Homeland Security. 2018. Russian Activity Against Critical Infrastructure. Retrieved from https://www.us-cert.gov/sites/default/files/c3vp/Russian_Activity_Webinar_Slides.pdf.
[62] NIST Special Publication 1108r3. 2014. NIST Framework and Roadmap for Smart Grid Interoperability Standards, release 3.0. Retrieved from http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1108r3.pdf.
[63] A. Nourian and S. Madnick. 2015. A systems theoretic approach to the security threats in cyber physical systems applied to stuxnet. *IEEE Trans. Depend. Sec. Comput.* 15, 1 (2015), 2–13. DOI: https://doi.org/10.1109/TDSC.2015.2509994
[64] M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni, and H. V. Poor. 2016. Machine learning methods for attack detection in the smart grid. *IEEE Trans. Neur. Netw. Learn. Syst.* 27, 8 (Aug. 2016), 1773–1786. DOI: https://doi.org/10.1109/TNNLS.2015.2404803
[65] J. A. Pendergrass, S. Helble, J. Clemens, and P. Loscocco. 2018. A platform service for remote integrity measurement and attestation. In *Proceedings of the 2018 IEEE Military Communications Conference (MILCOM'18)*. 1–6. DOI: https://doi.org/10.1109/MILCOM.2018.8599735
[66] M. Pendleton and S. Xu. 2017. A dataset generator for next generation system call host intrusion detection systems. In *Proceedings of the 2017 IEEE Military Communications Conference (MILCOM'17)*. 231–236. DOI: https://doi.org/10.1109/MILCOM.2017.8170835
[67] D. B. Rawat and Ch. Bajracharya. 2015. Cyber security for smart grid systems: Status, challenges and perspectives. In *Proceedings of the IEEE Southeast Conference*. IEEE, 1–6. DOI: https://doi.org/10.1109/SECON.2015.7132891
[68] Reuters. 2016. U.S. Firm Blames Russian 'Sandworm' Hackers for Ukraine Outage. Retrieved from https://www.reuters.com/article/us-ukraine-cybersecurity-sandworm/u-s-firm-blames-russian-sandworm-hackers-for-ukraine-outage-idUSKBN0UM00N20160108.
[69] Sheldon M. Ross. 2001. *Probability Models for Computer Science* (1st ed.). Academic Press, Inc., Orlando, FL.
[70] S. Fries, H. J. Hof, and M. G. Seewald. 2010. Security of the smart grid - enhancing IEC 62351 to improve security in energy automation control. *Int. J. Adv. Secur.* 3, 4 (2010). DOI: https://doi.org/10.1.1.474.6536
[71] A. Sanjab and W. Saad. 2016. Data injection attacks on smart grids with multiple adversaries: A game-theoretic perspective. *IEEE Trans. Smart Grid* 7, 4 (Jul. 2016), 2038–2049. DOI: https://doi.org/10.1109/TSG.2016.2550218
[72] Anibal Sanjab, Walid Saad, Ismail Güvenç, Arif I. Sarwat, and Saroj Biswas. 2016. Smart grid security: Threats, challenges, and solutions. *CoRR* abs/1606.06992 (2016).
[73] H. Sedjelmaci and S. M. Senouci. 2016. Smart grid security: A new approach to detect intruders in a smart grid Neighborhood Area Network. In *Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM'16)*. 6–11. DOI: https://doi.org/10.1109/WINCOM.2016.7777182
[74] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni. 2001. A fast automaton-based method for detecting anomalous program behaviors. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy (SP'01)*. IEEE Computer Society, 144.
[75] B. Sikdar and J. H. Chow. 2011. Defending synchrophasor data networks against traffic analysis attacks. *IEEE Trans. Smart Grid* 2, 4 (2011), 819–826.
[76] Y. Sun, X. Guan, T. Liu, and Y. Liu. 2013. A cyber-physical monitoring system for attack detection in smart grid. In *Proceedings of the 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'13)*. 33–34. DOI: https://doi.org/10.1109/INFCOMW.2013.6970712
[77] Symantec. 2018. Sandworm Windows Zero-day Vulnerability Being Actively Exploited in Targeted Attacks. Retrieved from https://www.symantec.com/connect/blogs/sandworm-windows-zero-day-vulnerability-being-actively-exploited-targeted-attacks.
[78] K. Tazi, F. Abdi, and M. F. Abbou. 2015. Review on cyber-physical security of the smart grid: Attacks and defense mechanisms. In *Proceedings of the 2015 3rd International Renewable and Sustainable Energy Conference (IRSEC'15)*. 1–6. DOI: https://doi.org/10.1109/IRSEC.2015.7455127

[79] The smart grid interoperability panel - cyber security working group. 2010. Introduction to NISTIR 7628: guidelines for smart grid cyber security. http://www.nist.gov/smartgrid/upload/nistir-7628_total.pdf.

[80] J. Valente, C. Barreto, and A. A. Cardenas. 2014. Cyber-physical systems attestation. In *Proceedings of the 2014 IEEE International Conference on Distributed Computing in Sensor Systems*. 354–357. DOI: https://doi.org/10.1109/DCOSS.2014.61

[81] W. Wang and Z. Lu. 2013. Survey cyber security in the smart grid: Survey and challenges. *Comput. Netw.* 57, 5 (Apr. 2013), 1344–1371. DOI: https://doi.org/10.1016/j.comnet.2012.12.017

[82] X. Li, I. Lille, X. Liang, R. Lu, X. Shen, X. Lin, and H. Zhu. 2012. Securing smart grid: Cyber attacks, countermeasures and challenges. *IEEE Commun. Mag.* 50, 8 (2012), 38–45. DOI: https://doi.org/10.1109/MCOM.2012.6257525

[83] Y. Obeng, C. Nolan, and D. Brown. 2016. Hardware security through chain assurance. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'16)*. IEEE, 1535–1537.

[84] Y. Yan, Y. Qian, H. Sharif, and D. Tipper. 2012. A survey on cyber security for smart grid communications. *IEEE Commun. Surv. Tutor.* 14, 4 (2012), 998–1010. DOI: https://doi.org/10.1109/SURV.2012.010912.00035

[85] L. Yang, P. A. Crossley, A. Wen, R. Chatfield, and J. Wright. 2014. Design and performance testing of a multivendor IEC61850 x2013;9-2 process bus based protection scheme. *IEEE Trans. Smart Grid* 5, 3 (May 2014), 1159–1164. DOI: https://doi.org/10.1109/TSG.2013.2277940

[86] Q. Yang, Rui Min, D. An, W. Yu, and X. Yang. 2016. Towards optimal PMU placement against data integrity attacks in smart grid. In *Proceedings of the 2016 Annual Conference on Information Science and Systems (CISS'16)*. 54–58. DOI: https://doi.org/10.1109/CISS.2016.7460476

[87] L. Zhou and Y. Makris. 2017. Hardware-based on-line intrusion detection via system call routine fingerprinting. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'17)*. 1546–1551. DOI: https://doi.org/10.23919/DATE.2017.7927236

[88] Y. Zhou and Z. Miao. 2016. Cyber attacks, detection and protection in smart grid state estimation. In *Proceedings of the 2016 North American Power Symposium (NAPS'16)*. 1–6. DOI: https://doi.org/10.1109/NAPS.2016.7747874

Received April 2018; revised April 2019; accepted July 2019