Distributed generation (DG) and microgrids (MG) are critical components of future power systems. However, the reliance of DGs and MG on resource-constrained embedded controllers for their operation renders them potential cyberattack targets. In this paper, we analyze the adversarial objectives of attackers attempting switching and control input modification attacks by manipulating controller firmware. We demonstrate the attack impact in the simulated Canadian urban distribution feeder system consisting of four DGs. To detect malicious firmware within the inverter controllers, we propose utilizing custom-built Hardware Performance Counters (HPCs) in conjunction with Time Series Classifiers (TSCs). TSCs respect the sequential order and attributes of the utilized custom-built HPCs sampling the controller’s firmware. Our experimental results demonstrate that malicious firmware can be successfully identified with 97.22% accuracy using a TSC trained on a single custom-built HPC.

© 2022 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Electric power systems are transitioning from unidirectional power flow architectures of the past – supported by bulk generation facilities – to distributed networks that facilitate bidirectional power flows. The growing penetration of DG improves the system resilience (i.e., through resource redundancy), but also introduces some unique challenges. The ‘active’ DG resources typically connect to the main grid utilizing voltage converters. These voltage converters operate fundamentally differently from synchronous machines, which were the focal point of traditional power system design. As a result, power system deployments need to be revised to account for the behavior of such low-inertia systems under accidental faults, i.e., factoring the inverter-based DG transient behavior, as well as malicious cyberattacks (Markovic et al., 2021).

Safeguarding inverter-based DGs against cyberattacks is critical to ensure the secure operation of future microgrid (MG) and smart grid setups. Researchers have tried to enumerate the vulnerabilities of inverter-based systems and propose attack detection and mitigation methods (Zografopoulos et al., 2022; Sebastian et al., 2019; Ye et al., 2021). However, challenges remain to be addressed (Youssef, 2021; Xenofontos et al., 2021), e.g., (i) the diversity of embedded systems utilized in inverter controllers provided by different vendors, and (ii) retrofitting security mechanisms to already deployed and legacy devices. Malicious firmware modifications (targeting the controllers) can have significant consequences ranging from inefficiencies in power generation (within DG systems) to grid failures. Hardware-based Detectors (HDs) are a prominent solution for the detection of malicious firmware (Malone et al., 2011; Ozsoy et al., 2016). Hardware Performance Counters (HPCs), a subclass of HDs, are dedicated registers that monitor low-level microarchitectural events such as the number of bus-cycles, branch instructions, and cache misses. The HPCs collected from the execution of a program enable obtaining a unique hardware footprint that can be utilized to train Machine Learning (ML) classifiers for malicious firmware detection (Sayadi et al., 2019; Wang et al., 2016).

In many cases, securing power systems with HPC-based detectors is challenging, since grid controllers utilize microcontroller-based solutions, due to their low cost and power consumption, which lack HPC support. To ameliorate this challenge, we have previously proposed the use of custom-built HPCs that keep track of assembly instruction sequences within the inverter’s firmware (Kuruvila et al., 2021b). Subsequently, ML can be applied to these custom-built HPCs for the classification and detection of malicious firmware. However, traditional ML models do not respect the sequential order of custom-built HPCs.
since the classifier will test each individual sample independently. Therefore, they still yield false positives because malicious firmware can incorporate benign instructions that result in similar custom-built HPC counts (Pan et al., 2020).

Assessing the impact of a compromised grid controller independently is the first step when designing cyberattack detection mechanisms. However, MGs are typically largely interconnected systems and attacks targeting multiple DG nodes simultaneously should be factored in cybersecurity analyses when implementing detection frameworks (Zografopoulos and Konstantinou, 2021). Adversaries can exploit DG interdependencies when crafting their attacks, to impact nominally operating grid assets which would attempt to absorb grid anomalies. As a result, undesirable impacts – when a subset of DGs is targeted – can propagate and destabilize the entire MG system, especially in weak grid architectures (as seen in Section 4) (Adib et al., 2018).

To address the aforementioned issues, in this paper, we propose respecting the sequential order of the custom-built HPCs by utilizing Time Series Classifiers (TSCs). TSCs consider both the data at the point at which they are observed and the observations at those points. Consequently, TSCs do not scramble the data because the sequential attributes of a custom-built HPC interval encapsulate better classification information when compared to individual samples. To the best of our knowledge, this is the first work that exploits TSCs in conjunction with custom-built HPCs for malicious firmware detection in a MG architecture involving four inverter-based DGs. Specifically, our contributions are as follows:

- We assess the comprehensive impact of switching and control input manipulation attacks targeting a MG with four inverter-enabled DGs.
- We leverage time series classification in conjunction with custom-built HPCs for the detection of malicious firmware in inverter clusters that lack HPC support.
- We assess the time series classification capabilities when utilizing multiple custom-built HPCs for the detection of attacks in DG controllers.

The rest of the paper is organized as follows. Section 2 discusses preliminary power system background information and previous works on cybersecurity and malware detection for securing grid infrastructure. In Section 3, we present our threat model and proposed methodology, while we demonstrate the attack impact and detection results in Section 4. Lastly, Section 5 concludes the paper and discusses future work.

2. Background and related work

2.1. Preliminaries for distributed grid systems and custom-built HPC designs

DG offer a plethora of advantages compared to traditional hierarchical power systems. They can supply flexible and economic power by replacing costly and geographically dispersed bulk generation facilities while eliminating potential transmission losses. Furthermore, DG can maintain power provision to critical loads during emergencies, enhancing power system resiliency to operational disruptions. However, power systems with increased DG also exhibit certain vulnerabilities. For example, voltage, frequency, and active or reactive instabilities happening on the DG-side can propagate to the MG or main grid sides, adversely affecting their operation (Zografopoulos et al., 2021b,a).

The introduction of inverter-based DG further aggravates these circumstances, since adversaries can mount stealthy attacks causing varying degrees of damage (from intermittent operation to blackouts, and cascading failures). In this paper, we utilize custom-built HPCs to detect such malicious attacks.

HPCs are special purpose registers that monitor low-level microarchitectural events, such as the number of branch instructions and cache-misses. The number of available HPCs is processor specific, and typically four to six HPCs can be concurrently observed (Demme et al., 2013). Although HPCs were developed for software profiling, previous research has demonstrated the feasibility of employing HPCs in conjunction with ML models for the detection of pernicious programs (Wang and Karri, 2015; Demme et al., 2013). HPCs, have also been used in commercial products such as Intel’s HPC-based Threat Detection Technology (TDT), which is utilized by Microsoft Defender for Endpoint (Konstantinou et al., 2022). Contrary to most modern processors furnishing HPC support, HPCs are not readily available in microcontroller-based devices such as the Texas Instruments (TI) C2000 microcontroller family, which provides real-time control for industrial, automotive, and electrical applications. Despite the lack of HPC support, many embedded devices utilize microcontrollers given their low power consumption and minimal costs; this is the case with solar inverters in DG nodes. Therefore, it is imperative to identify alternative hardware features to profile benign and malicious firmware (Malinowski and Yu, 2011).

To address the lack of HPC support in microcontroller devices, proposed methodologies extract distinct assembly instructions counts from a program’s static disassembly as custom-built HPC features (Rohan et al., 2019; Kuruvila et al., 2021b). Specifically, the assembly instructions profiled could include arithmetic, branch/jump, load, boolean, and store instructions, which we denote as $a$, $b$, $l$, $n$, and $s$, respectively. Additionally, custom-built HPCs could also profile possible assembly instruction combinations. As an example, the custom-built HPC $ab$ represents an arithmetic instruction that is followed by a branch/jump instruction. For the generation of custom-built HPCs, the sequence of the aforementioned assembly instructions and combinations would be periodically monitored in the disassembled firmware. To emulate the collection of traditional HPCs, these custom-built HPC values would be reset at a predefined interval. This interval is volatile and depends on the static disassembly, since utilizing a large value results in more information per custom-built HPC, at the cost of fewer samples furnished. In contrast, a smaller interval engenders more samples at the price of less information present per custom-built HPC. To further bolster the detection efficacy of custom-built HPCs, we use them in conjunction with TSCs that respect the sequential trace order. An overview of the HPC-based detection methodology is depicted in Fig. 1.

2.2. Related work

In the domain of identifying and defending against malicious firmware, various prior works have been proposed. GoodUSB, a technique that implements a regulating architecture in the Linux USB stack for defending against malicious firmware attacks compromising traditional USB devices that inject malevolent scripts, was presented in Tian et al. (2015). This technique involves a mediator module that permits or inhibits driver access based on the user’s expectations of the device’s functionality. Consequently, compromised USB devices are unable to execute their payload, as GoodUSB obstructs the permissions that would normally be granted. An approach that embedded lightweight intrusion checkers within a device’s firmware was proposed in Shila et al. (2016). This method involved utilizing source code instrumentation to detect anomalous activity by analyzing the current and previous recurrent operations for any deviations that would subsequently be flagged. For securing a computer mouse from the installation of pernicious firmware, a bootloader employing a signature verification code methodology was proposed.
in Maskiewicz et al. (2014). Additionally, VIPER, a software-based attestation technique that validates the authenticity of peripheral firmware but assumes the operating system and CPU have not been tampered with, was presented in Li et al. (2011).

In this paper, we employ HDs for detecting pernicious firmware modifications in DG nodes, i.e., inverter controllers, using a HPC-based approach. The advent of employing HPCs for identifying malicious applications was first proposed by Malone et al. (2011). However, this work was considerably ameliorated by utilizing the HPCs in conjunction with ML models for classifying programs (Demme et al., 2013). To detect kernel-level rootkits, HPC data was used to train a Support Vector Machine (SVM) classifier that was able to identify both encountered and novel rootkits in Singh et al. (2017). Furthermore, RAPPER, a ransomware detection technique that utilizes an Autoencoder to learn the HPC characteristics of benign behavior for identifying anomalous deviations, was introduced in Alam et al. (2020). An ensemble-based approach utilizing two to four HPCs per classifier for malware detection was presented in Sayadi et al. (2018). Feature selection was utilized to identify a smaller subset of HPCs that, when used in an ensemble, would furnish results consistent with models utilizing a larger set of HPC features. Finally, ConFirm, an HPC-based comparison technique, was developed for identifying firmware modifications in Wang et al. (2016). With respect to the domain of time series-based malware detection, an analysis using a multivariate time-series autoregressive moving average model that compares data for detecting Android Malware was presented in Kim and Choi (2015). Additionally, a shapelt TSC that extracts discriminatory features from an entropy time series-based signal for determining Malware in executable files was proposed in Patri et al. (2017). Moreover, an autoregressive model employing background network noise for detecting intrusions in network traffic was established in Bahaa-Eldin (2011).

While most modern processors have integrated HPCs into the architecture, microcontroller-based systems still lack adequate HPC support. Consequently, a plethora of techniques have been presented for detecting malware through the utilization of alternative HPCs. Researchers have defined and explored detecting malicious applications through ML models trained on sub-semantic features (Ozsoy et al., 2015, 2016). On the other hand, custom-built HPCs that count the ordering of specific instructions at the assembly level were used to train ML classifiers for detecting malware (Rohan et al., 2019). This work was extended to incorporate additional assembly instructions for detecting malicious firmware attacks in MCs, including Denial-of-Service (DoS) and Maximum Power Point Tracking (MPPT) modification threats (Kuruvila et al., 2021b). However, the domain of utilizing time series classification in conjunction with custom-built HPCs has yet to be explored. In this paper, we demonstrate how respecting the sequential characteristics of the custom-built HPC data is beneficial for ML models to differentiate between benign and malicious firmware.

3. Methodology

In this section, we discuss the threat model assumed in this work and provide the firmware attack design specifics. Then, we introduce our custom-built HPC-based time series attack classification methodology which is utilized for the identification of different types of firmware attacks on the DGs’ inverters.

3.1. Threat model for firmware attacks

The complex nature of distribution systems, given their interconnections between assets, i.e., DG resources, in addition to their cyber and physical counterparts, necessitates comprehensive threat modeling. We utilize the threat model methodology presented in Zografopoulos et al. (2021b) – conforming with the MITRE ATT&CK for Industrial Control Systems design and philosophy (MITRE, 2020a) – to outline the security assumptions made for the attacker capabilities and cyberattacks aiming to compromise the discussed MG architecture. We distinguish between the adversary model and the attack model, since the security implications are contingent upon the attacker capabilities, intentions and objectives as well as the attack specifics (e.g., reproducibility, functional level of the targeted asset, attack techniques and tactics used, etc.). These adversary and attack model details are also critical when performing cyberattack risk assessments, aiding the quantitative calculation of risk scores (factoring plausibility of threats) and the resource allocation prioritization dealing with malicious events.

We assume an oblivious adversary with restricted observability over the MG topology, and who could have direct access to a subset of DG inverter controllers. Our assumption is driven by the fact that the widespread decentralized deployment and control of commercial and consumer-level DERs (e.g., rooftop PVs, battery storage, etc.) expands the threat landscape. In such DER deployments, end-users are responsible for securing their assets, which in most cases are internet-connected and feature remote control and firmware update capabilities (Johnson, 2017). As a result, attackers could either be the end-users of inverter assets (i.e., possession) or could exploit existing vulnerabilities and/or zero-days in the remote functions of such assets (e.g. over-the-air firmware updates of programmable controllers) (Bettayeb et al., 2019; Sapir et al., 2022). The adversarial specificity, is presumed to be a non-targeted attack, i.e., any firmware modification that results in converter malfunctioning is favorable for our threat actor, while the adversarial resource requirements can vary from minimum (Class I adversaries) to state-funded criminals (Class II).

Regarding the attack model, we assume that to maximize the impact, attacks need to occur iteratively and must be performed multiple times. The asset under attack is the DG inverter controller, thus the attack level is Level 1. The attack technique exploits control logic code modifications (MITRE, 2020b), however other attack techniques have been utilized by adversaries in the past achieving similar results, e.g., modify controller tasking (MITRE, 2020c), modify parameters (MITRE, 2020d), modify program (MITRE, 2020e), etc. In our case, the attack premise

---

**Fig. 1.** Hardware Performance Counter (HPC) - based malicious firmware detection. (a) demonstrates the detection steps when benign firmware is uploaded to the inverter controller, while (b) presents the same process in the event of a malicious compromise.
targets the programmable controller of the inverter (Sapir et al., 2022). The attack model aims to compromise the inverter’s power conversion operation and efficiency – directly affecting the DG’s power output – by tampering with the control setpoints, and maliciously managing critical inverter submodules (e.g., gate driver modulation, output power stage switching, etc.). More details on the threat modeling terminology and the defined distinctions between the adversary and the attack model characteristics can be found in Zografopoulos et al. (2021b) and MITRE (2020a).

The objective of the proposed methodology is to detect malicious firmware before it can be uploaded to microcontroller-enabled devices; in our case, an inverter controller. The motivation comes from the fact that many industrial facilities rely on legacy embedded devices and controllers. According to IMS research, almost 85% of industrial devices are legacy systems with limited resources and security capabilities (Intel, 2015). Granted the lack of hardware malware detectors in many industrial controllers, we opt for a tailor-made HPC and time series-based detection method that can effectively enhance their security without any specific resource requirements (e.g., CPU, memory, etc.) or impact on their performance (e.g., computational latency, etc.). It should be noted that the presented detection scheme could be adapted for embedded devices supporting in-built HPCs, where it would not only be possible to detect malicious firmware versions before updates (in a secure update fashion), but also detect real-time anomalies via constantly monitoring HPC and time series events, at the expense of some additional computations.

### 3.2. Firmware attack design considerations

In this section, we will present two attack scenarios that adversaries can utilize to destabilize the DG nodes, causing anomalous behavior or total collapse. Specifically, we will focus on how attackers, by deliberately manipulating the inverter firmware functions (controlling the inverter conversion process) and strategically timing their attacks, can degrade the whole MG system operation and potentially damage critical grid assets. In Fig. 2 we demonstrate the inverter control loops, i.e., DC–DC and DC–AC conversion, that will be manipulated in the following attack models.

#### 3.2.1. Switching attacks

Switching attacks target the system availability by disabling the underlined power generating assets within the DG (e.g., solar inverters). For example, denial-of-service (DoS) attacks are a form of switching attacks where the output of a DG inverter can either be disabled by switching off its output stage, deactivating the inverter’s power control module (e.g., PWM block, transistor gate drivers), or by deactivating the whole inverter device (via remote commands). By manipulating the DG inverter timer and sensor measurements, we can craft such DoS switching attacks causing the inverter to operate intermittently. In Eq. (1) we demonstrate a simplified state space representation of the DC–DC control stage of the inverter. The current reference value provided by the MPPT control block will define the PWM duty cycle operating the flyback boost converter. If the real-time sensed measurements – essential for the MPPT block to calculate the PWM reference – are maliciously compromised (e.g., by altering their sensor filter gains – $C_A$), the inverter will operate abnormally. We will emulate the scenario where the photovoltaic power is not sufficient to initiate the inverter operation. In more detail, we can successfully compromise the inverter’s availability using switching attacks that trigger the aforementioned asset on or off at 0.1 s intervals causing significant voltage and frequency fluctuations, harmonics, etc.

\[
\begin{align*}
\dot{x}_{DC} &= A_{DC}x_{DC} + B_{DC}u_{DC} \\
y_a &= C_{DC}x_{DC}
\end{align*}
\]

where, $x_{DC}$ is the state variable vector that includes the DC voltage and current measurements from the PV, $A$, $B$, and $u_{DC}$ are the system, input matrices, and the control input vector, respectively. $y_a$ represents the compromised sensor measurement vector as a result of the manipulated sensor gains included in output matrix $C_a$.

#### 3.2.2. Control input attacks

The inverter controller is critical for the DG’s operation for a plethora of reasons. Inverter controllers regulate voltage and current outputs under varying solar insolation profiles, maximizing the power transfer from the solar panels to the DG system (e.g., maximum power point tracking algorithms) (Kuruvila et al., 2021b). Additionally, they can enhance grid stability via their inbuilt smart inverter volt–var and volt–watt functions, compensating reactive power injections and improving the power quality (i.e., rejecting harmonics). Given the importance of the inverter controller for the nominal DG operation, we have implemented attacks that manipulate the controller (DC–AC) inputs to destabilize its operation and cause the aforementioned unwanted events. Eq. (2) describes the control input attack model where optimal control decisions are manipulated by the malicious firmware ($g_0(u_{AC})$). We achieve that by periodically spoofing the controller inputs, leveraging the inverter’s timer, and by modifying the controller gains and reference setpoints with arbitrary values.

\[
\begin{align*}
\dot{x}_{AC} &= A_{AC}x_{AC} + B_{g_0}u_{AC} \\
y &= C_{AC}x_{AC}
\end{align*}
\]
Algorithm 1: Time Series Forest Feature Extraction.

```c
#define int loop, array arr_vals, data
arr_vals = Read Custom-built HPC Data
while loop < (X - val)
    for i < val do
        arr_vals.append(X[loop + i])
        i++
    end
    mean, std, slope = calculate_features(arr_vals)
data.append(mean, std, slope)
arr_vals ← empty Array
loop += val
end
```

A, and B, are the system and input matrices. The term $g_u(u_C)$ models the compromised control input vector, while $y$ and $C$ represent the sensor measurement vector and output matrix, respectively.

3.3. Time series-based classification

In order to detect the aforementioned malicious firmware attacks in Section 3.2, we utilize custom-built HPCs (Kuruvila et al., 2021b) as shown in Fig. 3. As previously explained in Section 2.1, custom-built HPCs profile distinct assembly instructions within a firmware’s static disassembly. Consequently, custom-built HPCs are dependent on the procurement of assembly instructions from the program running on the device to be secured. However, we note that many embedded systems and real-time applications utilize compiled languages like C/C++ because of their speed and resource management capabilities. Therefore, this enables the feasibility of a controller to generate the tailor-made HPC values from the disassembled firmware. In this paper, we monitor arithmetic, branch/jump, load, boolean, and store instruction, since they encompass a majority of the assembly instructions present in assembly code. Furthermore, we also monitor all possible custom-built HPCs assembly instruction combinations. Specifically, not only do we Profile an instruction succeeded by another, we also consider an instruction that is succeeded by itself. Therefore, we have 30 custom-built HPCs in our feature vector, as we have five individual instructions, $a, b, i, n,$ and $s$, as well as 25 different assembly instruction combinations. We note that custom-built HPCs are not limited to just individual and combinational pairs. It is feasible to extend this technique to include custom-built HPCs that monitor three or more consecutive assembly instructions. However, for our work in this paper, we did not require extending our feature vector to include these additional features. With our feature vector established, we can utilize our custom-built HPCs in conjunction with TSCs for determining the malicious firmware attack.

We leveraged time series classification with custom-built HPCs because traditional detection techniques have no regard for the temporal order of custom-built HPC data, i.e., the ML is trained on non-sequential data and inferences on individual samples, thereby disregarding any recognition of the sequential attributes. Disregarding the transient sequence of custom-built HPCs can result in erroneous classifications between firmware that share similar behaviors (Pan et al., 2020). Specifically, Time Series Classifiers (TSCs) consider both the data at the point at which it was observed and their respective values at those temporal periods. This is notionally represented as $x(t_1), x(t_2), x(t_3), ..., x(t_f)$ for observations at $t_1, t_2, t_3...t_f$ (Löning et al., 2019).

For our experimental results in Section 4, we employ the Time Series Forest (TSF) for all our time series classification experiments. We selected this classifier as prior works have shown that they are computationally more efficient while furnishing better classification metrics than other time series classifiers because of their ensemble nature (Deng et al., 2013). Specifically, the TSF is an interval-based model that extracts and incorporates information from different intervals of an input series (Bagnall et al., 2017). Algorithm 1 provides an overview of the operational steps of the model. First, the classifier operates by taking input data and splitting it into user-specified intervals. This step is imperative as separating the data into intervals instead of individual samples ensures the sequential properties of the data are respected. From the divided intervals, the TSF will extract different features such as the mean, standard deviation, and slope. Subsequently, the model is then trained on these extracted features (Amidon, 2020).

The interval features of mean ($f_1$), standard deviation ($f_2$), and slope ($f_3$) for given intervals $t_1$ and $t_2$ and values at specific times $v_i$ are defined as follows:

$$f_1(t_1, t_2) = \frac{\sum_{i=t_1}^{t_2} v_i}{t_2 - t_1 + 1}$$  \hspace{1cm} (3)

$$f_2(t_1, t_2) = \begin{cases} \sqrt{\frac{\sum_{i=t_1}^{t_2} (v_i-f_1(t_1,t_2))^2}{t_2-t_1}} & t_2 > t_1 \\ 0 & t_2 = t_1 \end{cases}$$  \hspace{1cm} (4)

$$f_3(t_1, t_2) = \begin{cases} \hat{\beta} & t_2 > t_1 \\ 0 & t_2 = t_1 \end{cases}$$  \hspace{1cm} (5)

where $\hat{\beta}$ is the least squares regression line (Deng et al., 2013). Furthermore, the TSF model’s inherent interval-based capabilities provide robustness because the features that are computed from the interval enable capturing and profiling the sequential characteristics of an application.
Since the custom-built HPCs are able to profile a device's firmware, as explained in Section 2.1, any malevolent attempt to obfuscate the firmware, e.g., by inserting instructions at the code level, would result in firmware digression from its "golden trace". HPCs have been proven an effective indicator of compromised firmware, however, we aim to enhance malware detection accuracy leveraging TSC analysis (SANS Institute, 2022; Kuruvila et al., 2021b). The TSF respects the sequential order of the custom-built HPC data by extracting the mean, standard deviation, and slope at different intervals as explained in Section 3.3. Therefore, during the TSF prediction, the interval features extracted from the manipulated data would differ from the expected values. Consequently, our proposed methodology is robust against attackers because the custom-built HPCs provide the granularity needed to profile the firmware. Subsequently, the TSCs can extract the sequential attributes from custom-built HPC data during training to furnish more accurate predictions than traditional models. We note that in our proposed methodology, the custom-built HPCs are hard-coded into the hardware for collection and do not require any upgrades. However, the selection of which custom-built HPCs are employed can be modified. Specifically, at any time, a subset of these 30 custom-built HPCs can be dynamically selected for utilization. The user is free to select only a couple or even all custom-built HPCs for employment.

To demonstrate the capability of our proposed methodology, in this paper, we utilize a Piccolo TMS320F28035 Isolated ControlCARD, which is part of TI’s C2000 ware microcontrollers, for controlling a commercially available solar microinverter (Texas Instruments, 2020). We note that our presented technique is not limited to electric grid controllers, as it is applicable for other embedded components. Since we are profiling the assembly instructions, our method is agnostic of the device being secured. We do not incorporate any measurements from the grid regarding power, current, or frequency. Therefore, if the assembly code of the firmware is procurable through a disassembler, TSC can be applied to the generated custom-built HPCs. However, the number of custom-built HPCs that can be profiled may be limited based on the assembly instructions present in the disassembly.

3.4. Combining custom-built HPCs

While time series classification provides many beneficial properties by accounting for and respecting the sequential attributes of the custom-built HPC order, it is not ideal to utilize all 30 features concurrently. As the number of employed features is increased, additional overhead is incurred to support monitoring all of these different events. Since these micro-controllers, such as TI’s C2000 ware series, consists of limited HPCs, it is ideal to add minimal primitives, to reduce the hardware overhead. While our feature vector consists of a plethora of different custom-built HPCs, we should ideally utilize the smallest possible amount (i.e., one to two HPCs) that enables detecting the malicious firmware attacks, thereby ensuring the required overhead is constrained. The difficulty with only employing a single custom-built HPC is that the TSF is limited in the information that can be extracted. Although the TSF’s sequential properties enable providing improved classification metrics of malicious firmware, false positives can still occur. Scenarios can arise where the custom-built HPC being used has synonymous instruction counts between the benign and malicious firmware. Therefore, while employing a single custom-built HPC furnishes a low overhead solution, additional features should be considered for ensuring robustness.

To address the aforementioned issue, we consider evaluating multi-feature utilization in conjunction with TSCs. Specifically, employing more than one custom-built HPC can bolster the classification metrics of the TSC because there is more information present to ensure robustness. Moreover, when we examine the quantity of the number of sequential series interval segments that are present within a custom-built HPC trace of a program, this quantity will always be smaller when compared to the count of traditional ML samples. Since regular ML models scramble the data, each unit is analyzed individually instead of jointly within a single series. Consequently, the amount of information each sample inherently contains is limited in traditional ML models. However, time series segment samples exhibit more beneficial classification information, since the longer trace allows for extracting more relevant sequential characteristics. Therefore, it becomes only natural to consider the effects of employing more than one custom-built HPC to bolster our utilized TSC’s accuracy. Multi-feature employment engenders additional time series segments for our TSC to learn and extract patterns from. Since prior work has already shown that HPC data has a positive correlation when jointly utilized, incorporating multiple custom-built HPCs concurrently will ideally provide an increase in detection accuracy (Kuruvila et al., 2021a).

4. Impact analysis and attack detection results

4.1. Impact analysis

In this section, we demonstrate the impacts of different attacks on the DG nodes. We have selected attack scenarios where the attacker aims to affect the grid operation through inverter firmware modifications. For instance, the inverter firmware could be manipulated to (i) arbitrarily switch on/off the DG’s inverter output stage, curtailing the generated power (switching/DoS attacks), or (ii) send malicious inputs to the inverter controller causing abnormal system behavior (control input attacks). The attack impact is first examined from the system operator perspective, by observing the per unit (p.u.) voltage and angle measurement perturbations in the presence of diverse attack classes targeting the inverter-based DGs. Then, we assess the corresponding impact of such attack classes from the DG perspective and provide the voltage, current and total harmonic distortion (THD) measurements of the DG nodes during attacks scenarios including switching, control input tampering, as well as combination of the aforementioned attacks in standalone and cascading scenarios.

4.1.1. DG system model

The MATLAB/Simulink software environment is used for the offline simulation of the Canadian distribution system model consisting of four inverter-based DGs, as described in Fig. 4. More extensive distribution system models, incorporating multiple feeders, node, and DGs, could also be used, without affecting our detection methodology. However, we opted for the Canadian urban distribution model because the impact of switching and control input attacks can be immediately observed (i.e., less interdependencies with neighboring nodes). For our worst-case scenario analysis, MATLAB/Simulink is sufficient to illustrate the adverse attack impacts, although there exist specialized software packages (e.g., ETAP, OpenDSS, GridLAB-D, etc.) that can reproduce and assess the results of firmware modifications on distributed system operations with high fidelity. The utility source point of connection of the Canadian urban model is assumed as an infinite bus for our simulations and operates at 120 kV while being connected to a 120 kV/12.5 kV step-down transformer. The remaining grid assets are coupled to the substation transformer through a circuit breaker, while a 2.75 MVar capacitor bank is also connected to the substation bus. The DG nodes under test operate at the 208 V level and are distributed along the feeder. The DGs are interfaced to the feeder (operating at 12.5 kV level) via 12.5 kV/208 V step-down transformers. Per Fig. 4 description,
the DG-connected loads are configured at 2 MVA with a power factor of 0.95 (Wang et al., 2009).

The DG inverter controllers utilize: (i) a phase-locked loop subsystem to assist the automatic gain control and track the operating frequency, and (ii) a discrete proportional–integral (PI) controller tracking the DG-generated peak voltage used to control the pulse-width-modulation input supplied to the inverter’s gate drivers. Additionally, every DG can trigger its islanding mechanisms to preemptively disconnect from the main MG if abnormal conditions are encountered, e.g., frequency and/or voltage violations.

4.1.2. Attack impact analysis

Following the Canadian distribution system model architecture presented in Fig. 4, we have reported that all four DGs can operate in either a fully connected, where power is shared between the DGs and the main distribution system feeder, or an autonomous mode. The autonomous or islanded mode can be initiated by the system operators if abnormal conditions, e.g., frequency or voltage fluctuations, are encountered to protect the DG assets (i.e., customers and loads) and avoid damage or loss of operation.

Notably, by triggering the DG’s islanding mechanism, the damage arising from either the transmission or the distribution system side can be self-contained. However, once a DG is decoupled from the main system, it should be able to cover its respective power demands arising from the inherent loads, such as industrial and residential customers, or perform load shedding by prioritizing any critical loads whose operation needs to be sustained. In Fig. 5, we present the base case of our system where all DGs are operating nominally; this fact is supported by both the DG and feeder voltages, which immediately stabilize to 1 p.u., as well as the voltage angles which decay to 0 degrees immediately after the system simulation start.

According to our threat model description from Section 3.1, we assume sophisticated attackers, who aim to maximize their attack impact by meticulously orchestrating their attacks. In more detail, we consider attack scenarios which involve: (i) switching/DoS attacks targeting the DG’s inverter operation, (ii) input-tampering attacks directed at the DG’s inverter controller compromising the power conversion process, and (iii) combinations of the two previously mentioned attack types leading to more complex malicious scenarios. In the aforementioned attack scenarios, the severe voltage fluctuations cause the immediate islanding of the DG nodes from the main MG (when the DG voltage drops below 0.98 p.u.). In Fig. 6, we present the voltage and angle measurement behaviors under three different compound attack classes. Significant voltage and angle deviations are shown when compared to the base case scenario (Fig. 5).

Specifically, in Fig. 6, we assume that the attacker is performing a two-step attack which aims to first isolate the DGs from the main feeder (triggering the islanding mechanism) and then spoof the DGs’ inverter controller to cause further damage to the DGs’ contained assets (e.g., consumers and loads). In Fig. 6(a), the switching/DoS attack targeting DG node #1 is initiated at 0.1 s causing the defensive islanding of the targeted DG and immediately after (at 0.2 s) the input-tampering attack is ensued compromising the inverter controller. The impacts of the aforementioned on both the voltage and phasor angles are demonstrated in Fig. 6(a). The results of similar attacks are depicted in Figs. 6(b) and 6(c), where in Fig. 6(b), both DG nodes #1 and #3 are targeted by the adversary, while in Fig. 6(c), all DGs are compromised in a cascading fashion (one after the other).

Apart from the “macroscopic” attack impact of malicious attacks as can be seen by system operators having access to voltage and angle measurements (Fig. 6), it is also worth investigating the impact of such attacks within the DG systems. In Fig. 7, we demonstrate how the three-phase voltage, current, and total...
harmonic distortion (THD) properties of the DGs are impacted during switching control-input tampering attacks. Notably, in Fig. 7(a), we depict the voltage, current, and THD values of any DG operating in the attack-free scenario, while Figs. 7(b), 7(c), 7(d) and 7(e) outline the detailed DG node when subjected to different attack scenarios.

In Figs. 7(b), 7(c), and 7(d), we observe the impact of a combined switching and control input attack targeting DG nodes #1, #2, and #3 correspondingly. All three attacks begin by maliciously manipulating the islanding mechanism of the DGs. As can be seen in the figures, the voltage and current values denoting inverter-based power generation fluctuate (from zero power (i.e., no generation) to some generation) when the DGs are connected and then disconnected from the feeder. Furthermore, apart from the switching attack, a control-input attack is superimposed on these attack use cases adversely affecting DG inverter power conversion profiles. In Fig. 7(b), this control input attack starts at 0.2 s, while the attack on DG nodes #2, and #3 are initiated at 0.25 s, 0.3 s, and 0.35 s respectively.

In Fig. 7 the attack impacts on neighboring DGs cannot be observed since we focus only on the local DG measurements (e.g., voltage, current, THD). However, Fig. 6(c) explicitly demonstrates how attack orchestration can degrade voltage stability as seen from the distribution system operator perspective. In such scenarios, where all DGs have been compromised (Fig. 6(c)), defensive islanding and load curtailment are enforced as remedial actions to secure both the generation resources and their associated consumers and loads. In our case, such remedial actions would cause the brownout of the DGs since they have already been disconnected from the main grid, and they rely on their inverter-based resources to meet their load demands.

The THD values should also be monitored while these attack scenarios are in effect. The demonstrated rapid fluctuations of the THD indicate significant degradation in the power quality of the underlined DG system, and increased harmonic frequencies, which could potentially lead to damaging the contained assets or would require extensive frequency compensation schemes to negate their effects and bring the THD values to acceptable levels.

Contrary to the attacks demonstrated in Figs. 7(b), 7(c), and 7(d), in Fig. 7(e), we illustrate the impact of a ‘solely’ switching attack on DG node #4. Evidently, the power generation profile of DG #4 inverter is affected when the DG node is intermittently connected to the feeder, enabling the bidirectional flow of power. The THD values are also affected by the switching attack, however in this use-case we can observe first that the amplitude of the distortion is significantly smaller when compared to the previous three scenarios and it is closely following the switching pattern introduced by the sectionalizations. Although, the severity of this attack should not be overseen, the corresponding impact on the DG operation is notably less pronounced when compared to the compound attack cases. Thus, the simulation results of the attack examples mentioned earlier in this section validate our assumption that sophisticated adversaries by thoughtfully orchestrating their attacks could cause a significantly greater impact on the overall system, even when parts of it operate in a distributed fashion.

4.2. Attack detection results

The objective of our methodology is to detect malicious firmware versions before they are uploaded to resource-constrained embedded devices (e.g., lacking hardware-based malware detectors). To assess and validate the capabilities of our proposed methodology in defending against the malicious firmware attack ramifications shown in Section 4.1.2, we evaluate our technique using a real-world commercially available firmware for a microinverter system. Specifically, we employ TI’s solar microinverter that utilizes a F2803x microcontroller, which is used for both the testing and evaluation of the attacks (Texas Instruments, 2020). We have generated four malicious firmware versions, in accordance with the attack scenarios described in Section 3.2, by introducing modifications to the solar microinverter code. Namely, the scenarios involve a switching attack disabling a whole DG node, a targeted switching attack on the MPPT functionality of the DG, and two MPPT control input attacks that introduce inefficiencies within the solar power conversion process. The malicious firmware versions materializing the aforementioned attacks were ported to the Piccolo TMS320F28035 Isolated ControlCARD that uses a DIMM100 Docking Station Baseboard for supplying the board-specified power and JTAG capabilities.

To obtain the static disassembly instructions from the flashed firmware, we utilized TI’s dis2000 dissembler. We selected this dissembler as this tool was specifically created by TI for dissembling firmware for their C2000WARE microcontrollers. We
employ the custom-built HPCs presented in Section 2.1 for profiling the flashed benign and malevolent firmwares. To generate these custom-built HPCs, we incorporate a Python script that iteratively parses the assembly instructions and computes the number of each distinct custom-built HPC in the disassembled firmware executable. Since HPCs are counted periodically, i.e., at fixed intervals, our Python script collects the total number of each custom-built HPC every 50 assembly instructions. This sampling rate is utilized as it is large enough to capture relevant information about the structure of the program, but diminutive enough to furnish an adequate amount of samples (Kuruvila et al., 2021b). We balanced the samples in our dataset to remove any unwanted bias from skewing our results. Specifically, we utilized an 80:20 split where our model is trained on 80% of the dataset and tested on the remaining 20%. We aggregate samples from all the switching control input attacks as well as the “golden” base-case firmware. Once the custom-built HPCs have been generated, we subsequently train the TSF classifier that was developed in Python using the sktime library (Löning et al., 2019). For the TSF model, we employed a time series interval length of five to ensure the model had a reasonable amount of samples to extract sequential attributes from. Other time series lengths could have been employed, albeit with a difference in model performance.

4.3. Timeseries-based detection with custom-built HPCs

To evaluate the effectiveness of utilizing TSCs in conjunction with custom-built HPCs for detecting malicious firmware attacks, we analyzed the classification metrics furnished when utilizing each unique assembly instruction. Since we want to minimize the amount of features that are employed, analyzing the efficacy of our proposed technique when utilizing a single feature is important. In Fig. 8(a), we present the results when utilizing different traditional ML models including Random Forest (RF), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Neural Network (NN) on custom-built HPC data for detecting the attacks from Sections 3.2.1 and 3.2.2. We determined the accuracy when utilizing each of the main assembly instructions, arithmetic $a$, branch/jump $b$, store $s$, load $l$, and boolean $n$, as explained in Section 2.1. From our results, the RF model furnished the highest accuracy of all classifiers for each employed custom-built HPC with a total average of 52.89%. In Fig. 8(b), we present the
accurate, precision, and recall when the TSF is trained on these
individual assembly instructions. The average furnished accuracy
was 91.97%, which is an increase of 39.08% when compared to the
traditional results of the RF. Our best results were furnished when
the TSF was trained using custom-built HPC “s”. Specifically, we
obtained an accuracy of 97.22% and recall of 93.33%. Moreover,
the precision was 100% which is imperative as we did not incur
any false positives and demonstrates the benefits of utilizing
sequential attributes when employing a single feature.

4.4. Robustnesstoimbalanceddata

Our results in Section 4.3 utilized a balanced dataset to ensure
a fair evaluation of the performance metrics. Specifically, we
were primarily focused on ensuring the false positive rate was
minimized. An imbalanced dataset can lead to a biased ratio of
false positives and false negatives. Consequently, to demonstrate
the robustness of time series classification, we utilize our dataset
without balancing the samples and retrained the TSF. There-
fore, the employed dataset will possess more malicious samples,
because we designed and collected custom-built HPCs for four
attacks while having one golden benign firmware collection. Our
results are shown in Fig. 9. Even with the biased dataset, the TSF
still furnished an average accuracy of 84.54%, which is only 7.43%
less than the results in Fig. 8(b). For the average precision, we ob-
tained 89.41% which is only 5.15% less than the 94.56% furnished
from the balanced dataset. This is noteworthy, as it is critical to
minimize the false positive rate. Consequently, our experimental
results verify the strength of time series classification, as it was
robust to the type of data employed.

4.5. Custom-built combinations with time series classification

In Fig. 10, we evaluate a traditional RF model and the TSF
when utilizing different custom-built HPC combinations. We only
assess the RF model because our results from Fig. 8(a) demon-
strate that the RF had the highest furnished accuracy among all
traditional models. Fig. 10(a) shows our results for all possible
assembly pairs when using the RF model. The average furnished
accuracy, precision, and recall was 64.85%, 58.47%, and 65.81%,
respectively. Our best results were obtained when using custom-
built HPC “b” and “n”, where the model furnished 71.58% accuracy,
which is an 18.69% increase over the average accuracy from
Fig. 8(a). However, while utilizing additional custom-built HPCs
Fig. 9. Time series classification on unbalanced dataset.

Fig. 10. Comparison between traditional and time series classification utilizing custom-built HPC combinations.
improved the RF’s classification metrics, they are still insufficient for incorporating in protection schemes. There would still be a plethora of erroneous furnished predictions, i.e., the false positive rate is major detriment. In Fig. 10(b), we provide our results when evaluating the effectiveness of utilizing different custom-built HPCs together in conjunction with the TSF. The average furnished accuracy was 96.75%, which is a 25.17% increase over the best result of the traditional RF model. Moreover, when employing custom-built HPCs “b” and “l” together, our TSF furnished 100% for all three performance metrics. This attests to the efficacy and benefits of time series classification that is able to extract more detailed characteristics than traditional HPC-based techniques. Consequently, these results indicate that respecting the sequential order of the data ensures that the TSF model can accurately distinguish between the attacks and benign behavior when using custom-built HPCs. Furthermore, they corroborate that only a limited number of custom-built HPCs are necessary for effective malicious firmware detection. While our feature vector consists of 30 custom-built HPCs, implementing an effective detection scheme would only incur overhead for monitoring two of these features.

4.6. Detection of unknown attacks

Our previous results demonstrated the beneficial advantages of time series detection over traditional ML models when using a single and multiple HPCs. To validate that the model was not overfitted, we now evaluate the efficacy of the TSF trained on custom-built HPCs for classifying unknown attacks. As explained in Section 4.2, we developed four different malicious firmware modifications involving two attacks that utilize DoS properties and two modifications that induce erroneous MPPT input. Our previous experiments trained the TSF on all of these attacks using 80% of the data for training and 20% for testing. In this experiment, we purposefully omit one attack from the training data and utilize the corresponding model to predict this unseen attack.

Our experimental results are presented in Fig. 11. Similar to our previous experiments, we evaluate the accuracy when using just a single custom-built HPC, i.e., the basic assembly instructions. When analyzing our results, we observe that, for all the omitted attacks, there was at least one custom-built HPC that furnished a minimum of 86.36% accuracy. Furthermore, when employing custom-built HPC “n”, we achieved over 90% accuracy for three of the four attacks. As previously explained in Section 3.3, any pernicious attempts to modify the firmware will result in different custom-built HPC counts, thereby deviating from the “golden trace”. Furthermore, when the TSF evaluates the data, the extracted interval features from the compromised data will also differ. Therefore, utilizing the TSF in conjunction with custom-built HPCs ensures a robust solution that can detect novel threats. Consequently, these results further bolster our conjecture of the advantages and benefits of time series detection.

5. Concluding remarks and future challenges

This paper analyzes firmware modification attacks targeting MG system with inverter-based DG. We illustrate the comprehensive impact of DoS and control input modification attacks leveraging the Canadian distribution feeder model. Furthermore, we utilize custom-built HPCs and time series analysis to identify malicious instructions and counterfeit firmware within the inverter controller. Our methodology can be implemented in zero-trust applications to support and safeguard devices with constrained computational resources lacking sophisticated security mechanisms (HPC support, secure boot, firmware signatures, etc.). Our proof-of-concept HPC-based malware detection method can be used to distinguish benign and malicious firmware before they can be uploaded to embedded controllers, mitigating potential unexpected consequences. Specifically, we port our tailor-made security primitive to TI’s F2803x microcontroller which does not natively support HPCs. Our experimental results indicate that (i) stealthy adversaries, by strategically constructing their attacks, can maximize their impact on the MG, leading to defensive islanding, load-sheddings, and power quality degradation jeopardizing the DG assets, and (ii) time series analysis on custom-built HPC data can efficiently classify firmware-based attacks within the inverter firmware with limited resources, even when dealing with unbalanced datasets.

Our future work will focus on deploying the proposed time series enabled classification framework leveraging HPCs in diverse embedded architectures – encountered in smart DG deployments – to circumvent adversarial behavior. We will also evaluate the real-time performance of our detection framework leveraging CPS testbeds and hardware-in-the-loop experiments, where commercial inverter systems will be interfaced with emulated large-scale DG power system topologies. In such experimental setups, we will assess the impact of diverse cyberattacks with high fidelity as well as implement and examine automated response and self-healing resilience mechanisms.
Although the presented HPC-based detection framework can effectively detect malicious firmware with minimal performance overheads, it should not be considered a “be-all and end-all” solution. Challenges exist on the device, vendor, and policy levels. On the device-level, for instance, lack of HPC support is an obvious disadvantage that we overcome with our custom-HPC solution. However, it still inhibits the online analysis and end-point detection of malicious firmware binaries within embedded controllers. Vendors, on the other hand, might not prioritize the security of their offered devices over their performance (security-performance trade-off). The procurement of embedded devices with more computational resources (e.g., HPC support), in addition to increasing manufacturing expenses, will also introduce implementation and unit testing costs. Even if some vendors are willing to invest their resources towards securing their devices, legacy devices as well as the heterogeneity of already deployed devices in power systems, e.g., different vendors, instruction set architectures, etc., still creates a big treat surface. Furthermore, governments and legislative organizations should provide security conformance standards to enforce stringent computational and security requirements (device future-proofing) given the influx of connected devices expected to be deployed in power systems (inverter-based resources, electric vehicles, battery storage, etc.). Policies and best practices should also be implemented to harden devices that cannot support contemporary security schemes or operate in untrusted environments (e.g., residential PV inverter controllers).

CRediT authorship contribution statement

Ioannis Zografopoulos: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. Abraham Peedikayil Kuruvilla: Conception and design of study, Acquisition of data. Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. Kanad Basu: Writing – original draft, Writing – review & editing. Charalambos Konstantinou: Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

Approval of the version of the manuscript to be published.

References

Adib, A., Mirafzal, B., Wang, X., Blaabjerg, F., 2018. On stability of voltage source inverters in weak grids. IEEE Access 6, 4427–4439.

A lam, M., Sinha, S., Bhattacharya, S., Dutta, S., Mukhopadhyay, D., Chattopadhyay, A., 2020. Rapper: Ransomware prevention via performance counters. arXiv preprint arXiv:2004.01712.

Amidon, A., 2020. A brief survey of time series classification algorithms. https://towardsdatascience.com/a-brief-introduction-to-time-series-classification-algorithms-7b4284631b97.

Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E., 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Discov. 31 (3), 606–660.

Bahaa-Eldin, A.M., 2011. Time series analysis based models for network abnormal traffic detection. In: The 2011 International Conference on Computer Engineering & Systems. IEEE, pp. 64–70.

Bettayeb, M., Nasir, Q., Talib, M.A., 2019. Firmware update attacks and security for IoT devices: Survey. In: Proceedings of the ArabWIC 6th Annual International Conference Research Track. In: ArabWIC 2019, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/3341305.3333169.

Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., Sto folks, S., 2013. On the feasibility of online malware detection with performance counters. ACM SIGARCH Comput. Archit. News 41 (3), 559–570.

Deng, H., Runger, G., Tuv, E., Vladimir, M., 2013. A time series forest for classification and feature extraction. Inform. Sci. 239, 142–153.

Intel, 2015. Connecting legacy devices to the internet of things (IoT). https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/connecting-legacy-devices-brief.pdf.

Johnson, J., 2017. Roadmap for Photovoltaic Cyber Security. Kim, K.-H., Choi, M.-J., 2015. Android malware detection using multivariate time-series technique. In: 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, pp. 198–202.

Konstantinou, C., Wang, X., Krishnamurthy, P., Khorrami, F., Maniatakos, M., Karri, R., 2022. HPC-based malware detectors actually work: Transition to practice after a decade of research. IEEE Design Test 1. http://dx.doi.org/10.1109/MDAT.2022.3143438.

Kuruvilla, A.P., Karmakar, S., Basu, K., 2021a. Time series-based malware detection using hardware performance counters. In: 2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, pp. 102–112.

Kuruvilla, A.P., Zografopoulos, I., Basu, K., Konstantinou, C., 2021b. Hardware-assisted detection of firmware attacks in inverter-based cyberphysical micro grids. Int. J. Electr. Power Energy Syst. 132, 107150.

Li, Y., McCune, J., Perrig, A., 2011. VIPER: verifying the integrity of PERipherals’ firmware. In: Proceedings of the 18th ACM Conference on Computer and Communications Security. pp. 3–16.

Löning, M., Bagnall, A., Ganeit, S., Kazakov, V., Lines, J., Kiräly, F.J., 2019. skitme: A unified interface for machine learning with time series. In: NeurIPS Workshop on Systems for Machine Learning. NeurIPS, pp. 1–9.

Malinowski, A., Yu, H., 2011. Comparison of embedded system design for industrial applications. IEEE Trans. Ind. Inf. 7 (2), 244–254.

Malone, C., Zahr, M., Karri, R., 2011. Ase hardware performance counters a cost-effective way for integrity checking of programs. In: Proceedings of the Sixth ACM Workshop on Scalable Trusted Computing. ACM, Chicago Illinois USA, pp. 71–76.

Markovic, U., Stanojev, O., Aristidou, P., Vretös, E., Callaway, D., Hug, G., 2021. Understanding small-signal stability of low-inertia systems. IEEE Trans. Power Syst. 36 (5), 3997–4017. http://dx.doi.org/10.1109/TPWRS.2021.3061434.

Maskiewicz, J., Ellis, B., Mouradian, J., Shacham, H., 2014. Mouse trap: Exploiting firmware updates in (USB) peripherals. In: 8th (USENIX) Workshop on Offensive Technologies (WOOT) 14.

MITRE, 2020a. MITRE ATT&CK for industrial control systems: Design and philosophy . https://collaborate.mitre.org/attackics/img_auth.php/3/37/ATT%26CK_for_ICS_-Philosophy_Paper.pdf.

MITRE, 2020b. MITRE ATT&CK for industrial control systems techniques: Module firmware. https://attack.mitre.org/techniques/T0839/.

MITRE, 2020c. MITRE ATT&CK for industrial control systems techniques: Modify controller tasking. https://attack.mitre.org/techniques/T0821/.

MITRE, 2020d. MITRE ATT&CK for industrial control systems techniques: Modify parameter. https://attack.mitre.org/techniques/T0836/.

MITRE, 2020e. MITRE ATT&CK for industrial control systems techniques: Modify program. https://attack.mitre.org/techniques/T0899/.

Ozsoy, M., Donovic, C., Gorlik, I., Abu-Ghazaleh, N., Ponomarev, D., 2015. Malware-aware processors: A framework for efficient online malware detection. In: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). IEEE, Burlington, CA, USA, pp. 651–661.

Ozsoy, M., Khasawneh, K.N., Donovic, C., Gorlik, I., Abu-Ghazaleh, N., Ponomarev, D., 2016. Hardware-based malware detection using low-level architectural features. IEEE Trans. Comput. 65 (11), 3332–3344.

Pan, Z., Sheldon, J., Mishra, P., 2020. Hardware-assisted malware detection using explainable machine learning. In: 2020 IEEE 38th International Conference on Computer Design (ICCD). IEEE, pp. 663–666.

Patri, O., Wojnowicz, M., Wolff, M., 2017. Discovering malware with time series shapelets. In: Hawaii International Conference on System Sciences. IEEE, pp. 1–10.

Rohan, A., Basu, K., Karri, R., 2019. Can monitoring system state+ counting custom instruction sequences aid malware detection? In: 2019 IEEE 28th Asian Test Symposium (ATS). IEEE, Kolkata, India, pp. 61–615.

SANS Institute, 2022. Using IOC (indicators of compromise) in malware forensics. https://www.sans.org/white-papers/34200/.
Sapir, M., Katz, U., Moshe, N., Brizinov, S., Preminger, A., 2022. Evil PLC attack: Weaponizing PLCs. https://claroty-statamic-assets.nyc3.digitaloceanspaces.com/resource-downloads/team82-evil-plc-attack-research-paper.pdf.

Sayadi, H., Makrani, H.M., Dinakarrao, S.M.P., Mohsenin, T., Sasian, A., Rafatirad, S., Homayoun, H., 2019. 2SMaRT: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection. In: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, Florence, Italy, Italy, pp. 728–733.

Sayadi, H., Patel, N., PD, S.M., Sasian, A., Rafatirad, S., Homayoun, H., 2018. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In: 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC). IEEE, pp. 1–6.

Sebastian, D.J., Agrawal, U., Tanimi, A., Hahn, A., 2019. DER-TEE: Secure distributed energy resource operations through trusted execution environments. IEEE Internet Things J. 6 (4), 6476–6486. http://dx.doi.org/10.1109/JIOT.2019.2909768.

Shila, D.M., Geng, P., Lovett, T., 2016. I can detect you: Using intrusion checkers to resist malicious firmware attacks. In: 2016 IEEE Symposium on Technologies for Homeland Security (HST). IEEE, pp. 1–6.

Singh, B., Evtyushkin, D., Elwell, J., Riley, R., Cervesato, I., 2017. On the detection of kernel-level rootkits using hardware performance counters. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. ACM, New York, NY, United States, pp. 483–493.

Texas Instruments, 2020. TMDSSOLARUINVKIT solar micro inverter development kit | Tlcom. http://www.ti.com/tool/TMDSSOLARUINVKIT.

Tian, D.J., Bates, A., Butler, K., 2015. Defending against malicious USB firmware with GoodUSB. In: Proceedings of the 31st Annual Computer Security Applications Conference. pp. 261–270.

Wang, X., Freitas, W., Dinavahi, V., Xu, W., 2009. Investigation of positive feedback anti-islanding control for multiple inverter-based distributed generators. IEEE Trans. Power Syst. 24 (2), 785–795. http://dx.doi.org/10.1109/TPWRS.2008.2007002.

Wang, X., Karri, R., 2015. Reusing hardware performance counters to detect and identify kernel control-flow modifying rootkits. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 35 (3), 485–498.

Wang, X., Konstantinou, C., Maniatakos, M., Karri, R., Lee, S., Robison, P., Stergiou, P., Kim, S., 2016. Malicious firmware detection with hardware performance counters. IEEE Trans. Multi-Scale Comput. Syst. 2 (3), 160–173.

Xenofontos, C., Zografopoulos, I., Konstantinou, C., Jolfaei, A., Khan, M.K., Choo, K.-K.R., 2021. Consumer, commercial, and industrial iot (in) security: Attack taxonomy and case studies. IEEE Internet Things J. 9 (1), 199–221.

Ye, J., Giani, A., Elasser, A., Mazumder, S.K., Farnell, C., Mantoost, H.A., Kim, T., Liu, J., Chen, B., S. Eo, G., Song, W., Roig Greidanus, M.D., Sahoo, S., Blaabjerg, F., Zhang, J., Guo, L., Ahn, B., Shadmand, M.B., Gajanjur, N.R., Abbasszada, M.A., 2021. A review of cyber-physical security for photovoltaic systems. IEEE J. Emerg. Sel. Top. Power Electron. 1. http://dx.doi.org/10.1109/JESTPE.2021.3111728.

Youssef, K.H., 2021. Optimal routing and design of radial medium voltage power distribution networks considering cables short circuit withstand-capacity. IEEE Trans. Power Deliv. 1. http://dx.doi.org/10.1109/TPWRD.2021.3110817.

Zografopoulos, I., Konstantinou, C., 2021. Detection of malicious attacks in autonomous cyber-physical inverter-based microgrids. IEEE Trans. Ind. Inf.

Zografopoulos, I., Konstantinou, C., Hatziargyriou, N.D., 2022. Distributed energy resources cybersecurity outlook: Vulnerabilities, attacks, impacts, and mitigations. http://dx.doi.org/10.48550/ARXIV.2205.11171, URL https://arxiv.org/abs/2205.11171.

Zografopoulos, I., Konstantinou, C., Tsoutsos, N.G., Zhu, D., Broadwater, R., 2021a. Security assessment and impact analysis of cyberattacks in integrated t&d power systems. arXiv:2102.03215.

Zografopoulos, I., Ospina, J., Liu, X., Konstantinou, C., 2021b. Cyber-physical energy systems security: Threat modeling, risk assessment, resources, metrics, and case studies. IEEE Access 9, 29775–29818. http://dx.doi.org/10.1109/ACCESS.2021.3058403.