2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)

2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID) | 979-8-3503-8440-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VLSID60093.2024.00037

Hardware-based Detection of Malicious Firmware Modification in Microgrids

Amisha Srivastava*, Sneha Thakur†, Abraham Peedikayil Kuruvila‡, Poras T. Balsara*, Kanad Basu*

*University of Texas at Dallas, USA
† IEEE Member
‡Samsung Electronics America, USA

***Abstract*—Microgrids play a pivotal role in shaping the future of sustainable and resilient energy solutions. However, their remote accessibility and control functionalities make them susceptible to cybersecurity threats. In particular, components such as Digital Signal Processing (DSP) Boards deployed in Microgrids are prime targets for cyber adversaries seeking to compromise the integrity and functionality of power systems. To address this threat, we propose a comprehensive methodology that integrates custom-built Hardware Performance Counters (HPCs) with Time Series Classifiers (TSCs) to efficiently detect malicious firmware in critical components operating within a Microgrid setup. Our experimental results demonstrate the effectiveness of the proposed approach, achieving up to 100% accuracy in detecting firmware modification attacks. This method represents a significant stride in the Design-for-Security (DfS) paradigm, bolstering the resilience of microgrids against cyber threats and safeguarding critical infrastructure.**

***Index Terms*—Hardware Performance Counters, Microgrids, Time Series Classification, Digital Signal Processing**

I. Introduction

The power system domain is undergoing a significant paradigm shift, transitioning from a conventional top-down model to a more decentralized, flexible, and smart-grid system. At the forefront of this revolution are Microgrids (MGs), which are the integration of Renewable Energy Sources (RES), Photo-Voltaic (PV) devices, wind, fuel cells into the existing power grid [1]. Microgrids are self-contained energy systems that encapsulate a plethora of resources such as wind turbines, solar PVs, gas generators, and Energy Storage Systems (ESS). Microgrids have created a need for an advanced and intelligent power system, known as smart-grid. A smart-grid integrates advanced sensing technologies, control methods, and integrated communications into the current electricity grid.

Smart-grids represent a transformative evolution in the energy sector, integrating advanced technologies and a wide array of interconnected devices to create an intelligent and dynamic energy management system [2]. Within these smart-grids, various interconnected devices play crucial roles in ensuring efficient and reliable energy management. One pivotal component in these systems is the Digital Signal Processing (DSP) Board. These DSP boards contribute to the grid's intelligent characteristics by enabling remote accessibility, control functionalities, and fast data sampling, which are essential for grid management and optimization. However, while the advantages of smart-grids facilitated by these components are significant, they also open doors to potential security vulnerabilities. The remote accessibility and control functionalities of DSP boards make them enticing targets for attackers. The allure of accessing and manipulating critical components remotely increases the risk of cyberattacks on smart-grids. Furthermore, the frequent installation of MGs in unsecured or remotely monitored environments, coupled with their initial design that may not prioritize strong security measures, further exacerbates their susceptibility to cyber threats. The combination of these factors creates potential entry points for attackers seeking to compromise the grid's integrity and disrupt its operations.

To safeguard critical infrastructure and address vulnerabilities, effective detection mechanisms are crucial. Anti-virus software (AVS) has traditionally been used, but it faces limitations like high computational overhead and lack of robustness [3]. AVS runs malware in a virtual machine to monitor behavior and API usage, leading to an ongoing challenge with attackers trying to evade detection. This requires significant computational resources for continuous defense improvements. One promising substitute for AVS is the employment of Hardware-assisted Malware Detection (HMD), an approach that leverages the physical hardware of a system as a defensive tool against potential cyber threats [4]. HMD works on the principle of analyzing hardware-level indicators, such as the number of branch-misses, CPU cycles, instructions, etc. in order to detect signs of malicious activity.

A key instrument in HMD are Hardware Performance Counters (HPCs), a type of embedded hardware resource in modern microprocessors [5]. HPCs allow for the monitoring of low-level microarchitectural events, including clock cycles, cache misses, and instructions retired. By assessing these metrics, HPCs can detect anomalous patterns in system performance that might indicate the presence of malware or a cyberattack because of the hardware footprint obtained through their utilization. However, the direct application of HPC-based HMD techniques to microgrid systems can be challenging. More specifically, some embedded legacy controllers that are integral parts of the microgrid infrastructure lack native HPC support. Optimized for specific tasks, these controllers may not inherently include built-in HPCs to facilitate the measurement of hardware performance metrics. As a result, traditional HPC-based HMD techniques that rely on accessing and utilizing hardware counters may not be directly applicable to such legacy controllers. Without this support, the utilization of HPCs to track and analyze system performance in these controllers becomes unfeasible.

To circumvent this limitation, we propose the use of custom-designed HPCs as Design-for-Security (DfS) primitives to enhance the security of firmware-based components in MGs. These custom HPCs are engineered to monitor the sequence of instructions within the microgrids's firmware, helping identify anomalous behaviors indicative of a potential attack. We leverage these custom-built HPCs in both traditional and time series-based machine learning classifiers to distinguish between benign and malicious firmware modifications. The advent of such strategies is anticipated to significantly bolster the cybersecurity of MGs, enabling their safe and secure operation in the smart-grid environment. Specifically, our contributions are summarized as follows:

- We design firmware modification attacks for crucial components in MGs including DSP boards considering their characteristics and how they operate in a MG setup,
- We assess the impact of these attacks targeting DSP boards on a simulated MG architecture,
- We leverage time series classification in conjunction with custom-built HPCs for the detection of malicious firmware in components of microgrids that lack HPC support,
- Demonstrate that the proposed approach, when evaluated on the modified firmware using the time series-based classifier, furnishes up to 100% accuracy, 100% precision, and 100% recall, respectively.

The rest of the paper is organized as follows. Section II provides the background on power system preliminaries, hardware performance counters and machine learning. Section III shows the prior related

979-8-3503-8440-6/24/$31.00 ©2024 IEEE
DOI 10.1109/VLSID60093.2024.00037

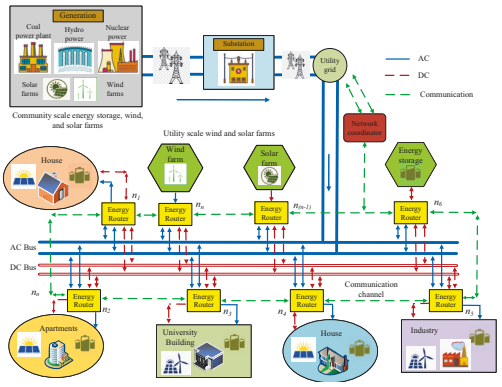


Fig. 1: Smart grid architecture, with ER nodes

work. Section IV describes the proposed methodology. Section V evaluates our proposed approach. Lastly, the paper is concluded in Section VI.

II. Background

A. Power System Preliminaries

Figure 1 depicts a microgrid with a smart ac-dc interface for any utility such as, home, industry or domestic/commercial network [6]. The utility interfaced to the grid with Energy Router (ER) forms an interactive node $(n_1, n_2, ...n_n)$. These ER nodes interact with the DC/AC bus in the grid and among each other using a communication channel. ER is a compact device which consists of power-electronic converters for interfacing an RES, ES, and DC/AC loads with the grid, and includes smart-control and sensing devices with a communication module for interacting with other ER's in the microgrid network. ER architecture and hardware is discussed in [1] and shown in Figure 2. It can be observed that the ER hardware consists of the signal processing board which comprises of the DSP chip and other components required to perform signal processing. The DSP board consists of the circuits including op-amps to measure the voltage and current on the ER. These signals are sensed by DSP TMS320F28335 at its Analog to Digital Converter (ADC) pins. The DSP processes and evaluates these signals, and does the required calculations per the control to generate the required ePWM signals for the converters in ER.

B. Hardware Performance Counters

Hardware Performance Counters (HPCs) are specialized registers designed to monitor microarchitectural events at a low-level, such as branch instructions or cache misses [5]. They are increasingly used in cyber-physical systems for performance tuning and software optimization. The number of available HPCs depends on the processor architecture, typically ranging from four to six concurrently. Linux

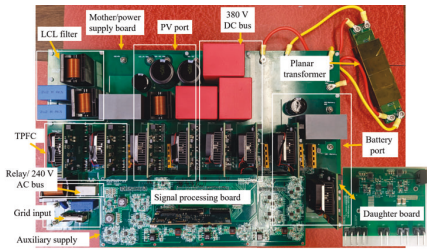


Fig. 2: ER complete hardware prototype

users can access HPC data through the Perf command provided by the linuxtools-common package [7]. Perf allows extracting hardware and software event information for a specific program. The number of obtainable HPCs varies across processors and can be listed using the perf list command. By using the perf stat command with suitable parameters, users can obtain HPC values corresponding to a specific program, enabling in-depth analysis of hardware and software performance metrics for system optimization and troubleshooting.

C. Machine Learning Preliminaries

1) Machine Learning Classifiers: In this paper, we utilize three essential Machine Learning (ML) classifiers for data analytics and predictive modeling. The Neural Network (NN) is adept at identifying complex patterns and is commonly employed in deep learning. The Decision Tree (DT) offers simplicity and interpretability, handling both categorical and numerical data. Lastly, the Random Forest (RF) reduces overfitting and improves results through its ensemble learning approach, combining multiple decision trees to make predictions.

2) Principal Component Analysis: Principal Component Analysis (PCA) is a data reduction method allowing users to define the number of principal components in transformed data [8]. It computes the covariance matrix from initial features, capturing variance relationships. Singular value decomposition extracts data in directions of highest variances. Selected based on largest values, matrix values correspond to original dataset features.

III. Related Work

The concept of HPCs for malware detection was initially introduced by [9]. It was later enhanced by the integration of machine learning algorithms for distinguishing between malware and benign applications [10]. Further advancements included the monitoring of system call events and control flows and real-time supervision of software running on embedded processors using HPC values [11] In the field of time-series-based malware detection, various methods have been explored for diverse tasks such as identifying malicious firmware in controllers [12]. The use of custom-built HPCs for counting assembly-level instructions has been leveraged to train machine learning classifiers for malware detection, and expanded to encompass additional assembly instructions to detect malicious firmware attacks in MGs [13], [14]. In this paper, we highlight the importance of acknowledging the sequential nature of custom-built HPC data, aiding ML models in distinguishing between benign and malicious firmware.

IV. Methodology

In this section, we present our security-centric methodology, designed to bolster the internal components incorporated in microgrids. The existing profiling capabilities of microgrid architectures are significantly limited, primarily restricted to counting clock cycles. This limitation hinders the detection of malicious firmware modifications, as more detailed structural information about the executed program is required. Hence, we incorporate custom-built HPCs into our DfS approach to enhance the security layers within the microgrid ecosystem [14]. Subsequently, we employ a time series attack classification methodology to identify various types of firmware attacks on the microgrid.

A. Threat Model

Our threat model focuses on an adversary seeking to compromise the DSP boards in power microgrids. These boards are vital components responsible for microgrid operation. The adversary's strategy involves creating fraudulent firmware versions, which they aim to introduce into the targeted DSP boards. The attack primarily operates at the application level, utilizing static firmware alterations to initiate abnormal functions. Access can be obtained physically by directly uploading malicious firmware or remotely through vulnerabilities in

firmware update mechanisms. Once successful, the attack exploits the DSP board functionalities to inject malicious code capable of manipulating operation setpoints and sensor data received by the control unit. These process-aware attacks specifically target embedded devices and programmable logic controllers, posing challenges in terms of detection. The consequences of these attacks can range from equipment failures to complete cyberphysical system collapse. Next, we describe each of these attacks in detail and their effect on the microgrid system.

B. DSP Firmware Attacks

Firmware attacks aim to disrupt the functionality of the firmware-regulated device or manipulate it to produce incorrect system output. In this section, we highlight two types of attacks that adversaries could exploit in the DSP board: Denial of Service (DoS) attacks and attacks compromising the signal sensing functionality of the DSP board.

Interrupt Operator Attack: A DoS attack renders the system inoperative and dysfunctional. This is a form of DOS attack where the adversary systematically disrupts the operation of the DSP board by repeatedly toggling it on and off. The attacker manipulates the board's functions by triggering specific system operations which cause interruptions. The duration of each interruption is contingent on certain system parameters, resulting in an unpredictable and erratic operation of the DSP board. By exploiting the system timer, this attack disrupts the availability of certain circuit components, causing intermittent disruptions and system unavailability.

Signal Sensing Attacks: Since the DSP board utilizes Analog to Digital Converters (ADC) to interpret the input signals and respond appropriately, we have developed several attacks compromising this signal sensing functionality.

- **Loop Attack**: This attack results in the creation of an infinite loop due to the alteration of the logical loop condition. It is designed to make the DSP board enter a perpetual loop, effectively rendering the subsequent operations useless and preventing the board from executing its intended tasks. This attack could lead to a freeze in system operations, potentially causing a shutdown.
- **ADC Reading Change Attack**: In this attack, the adversary swaps the ADC function parameters with incorrect values, leading to erroneous readings. The ADC plays a crucial role in the DSP board operation as it interprets incoming signals. Any manipulation of this function can significantly affect the board's performance, leading to system errors and unpredictable behavior.
- **ADC Window Attack**: Here, the attacker modifies the ADC resolution window, which is a key parameter for the ADC's operation. By changing this, the attacker can disrupt the normal operation of the DSP board. The ADC resolution window determines the precision and accuracy of the data conversion process. Altering it can lead to significant discrepancies between the actual and sensed data, leading to incorrect operations and potential system malfunctions.

The highlighted DSP firmware attacks underscore the vulnerabilities inherent in power microgrids, particularly through DoS attacks and compromised signal sensing functionality. Recognizing and comprehending these attack vectors is imperative in order to formulate effective security measures. We aim to enhance the security of microgrid operations and utilize the microgrids using the Digital Signal Processor (DSP) TMS320F28379D model from Texas Instruments (TI). These microgrids are based on the F28x architecture, which has limited profiling capabilities, allowing only clock cycle counting. To address this limitation and strengthen microgrid security, we integrate custom-built HPCs into our approach. These HPCs provide more comprehensive performance profiling and reinforce the security layers within the microgrid ecosystem.

C. Custom-built HPC Design and Collection

To protect the microgrids from the aforementioned firmware attacks, we propose leveraging a custom-built HPCs approach [14]. This technique enables us to monitor the sequence of specific assembly instructions embedded within binary executables. These counters span a broad spectrum of instructions: arithmetic (*a*), boolean (*n*), store (*s*), load (*l*), and branch/jump (*b*). What sets these custom HPCs apart is their capacity to register not just the frequency of each individual instruction type but also the sequences these instructions form. The HPC-based feature vector comprises of 30 individual HPCs, considering the five individual instructions, *a*, *b*, *l*, *n*, and *s*, along with the 25 possible assembly instruction combinations (*aa, ab, al, an, as, bb, ba, bl, bn, bs, ll, la, lb, ln, ls, nn, na, nb, nl, ns, ss, sa, sb, sl, sn*). The initial five HPCs are dedicated to recording the occurrences of arithmetic, store, load, boolean, and branch instructions respectively. Subsequent HPCs in the feature vector, which follow an *XY* format, measure the instances where an *X* instruction is immediately followed by a *Y* instruction. There are also HPCs like '*bb*', which register occurrences of back-to-back branch instructions. Therefore, we not only monitor an instruction succeeded by another type but also situations where an instruction is succeeded by an identical one. This robust approach facilitates a comprehensive view of the assembly instruction landscape, enhancing our ability to detect patterns and anomalies.

D. Time Series-based Attack Detection

Time Series Classification (TSC) is a technique used to classify and analyze time series data, where data points are collected over time [15]. In the context of microgrid security, TSC can be employed to monitor the behavior of various system components and detect any anomalies in the time series data. By combining TSC with custom-built HPCs, the microgrid system gains the ability to effectively detect and respond to suspicious activities, providing an added layer of security against attacks that may attempt to compromise the firmware of critical microgrid components. The HPCs are integrated into the microgrid system to enable more detailed and granular profiling of the system's performance metrics. Traditional ML techniques often neglect the inherent temporal order of HPC data, resulting in inaccuracies. To address this limitation, TSC methods are specifically tailored to handle sequential data, effectively capturing and incorporating temporal dependencies present in the time series. In Section V, we exclusively utilize the Time Series Forest (TSF) algorithm for all of our time series classification experiments. TSF is an ensemble-based approach that utilizes random forest techniques on subsequences of the time series [16]. Our decision to adopt the TSF classifier is primarily motivated by its ability to retain the sequential understanding of data, which is vital for recognizing complex sequential patterns and trends within time series data of the custom-built HPCs. TSF captures and integrates information from different time intervals, preserving the sequential nature of the data. This enables effective profiling of firmware behavior and detection of deviations caused by potential attacks, improving the overall efficacy of our methodology. We extract significant features within each interval to accurately represent the data characteristics. We propose the utilization of statistical features, namely, mean, standard deviation and slope to represent the time series data [16]. These statistical features are simple yet effective representations of data characteristics and offer ease of implementation.

The inputs to Algorithm 1 are the collected custom-built HPCs, represented by the variable *HPC*. The output of the algorithm is the detected malicious firmware indicated by the variable *detected_malware*. The algorithm begins by defining the function *TSF_Classifier(HPC)*, which takes the input *HPC*, representing the collected custom-built HPC data as shown in lines 1-5. The next step is to split the input HPC data into intervals to ensure the preservation of the temporal order. This is achieved through the

Algorithm 1 Time Series Detection

Input: *HPC*
Output: detected_malware

```
1: function TSF_Classifier(HPC)
2:   intervals = split_into_intervals(HPC)
3:   features = extract_features(intervals)
4:   TSF_model = train_TSF(features)
5:   return TSF_model
6: function split_into_intervals(HPC)
7:   intervals ← [ ]
8:   interval_size = user_defined
9:   for i in step_interval_size do
10:    interval = HPC[i : i + interval_size]
11:    intervals ← interval
12:  return intervals
13: function extract_features(intervals)
14:  features ← [ ]
15:  for i in intervals do
16:    mean = calculate_mean(interval)
17:    std_dev = calculate_std_dev(interval)
18:    slope = calculate_slope(interval)
19:    feature_vector = [mean, std_dev, slope]
20:    features ← feature_vector
21:  return features
22: TSF_model = TSF_Classifier(HPC)
23: return detected_malware
```

function *split_into_intervals(HPC)*, which returns the intervals as a list, as represented by lines 6-12. The extracted intervals are then utilized to calculate significant features for each interval. The function *extract_features(intervals)* computes the mean, standard deviation, and slope of each interval and stores these features in a list, as indicated by lines 13-21. With the extracted features, the TSF classifier is trained using the function *train_TSF(features)*, where the input is the list of features calculated in the previous step. The output is the trained TSF model. After the TSF model is trained, it is returned from the function *TSF_Classifier(HPC)*, as shown in line 22. The final step of the algorithm is to utilize the trained TSF model to detect malicious firmware. The algorithm concludes by returning the detected malicious firmware through the variable *detected_malware*.

Our experimental evaluation in Section V demonstrates the advantage of the TSF classifier over traditional ML models, showcasing improved classification metrics in terms of accuracy, precision, and recall. Working in conjuction with TSF, the custom-designed HPCs are capable of profiling a device's firmware. Any malicious attempt to obfuscate the firmware would result in deviations from its normal behaviour a standard profile established for comparison. By applying the TSF to HPC data, we are able to retain and analyze these deviations in the context of their sequential order. This sequential understanding coupled with the TSF's interval-based feature extraction ensures a more accurate and robust malware detection process than traditional models, making our proposed methodology highly effective against attackers.

Our DfS methodology, as depicted in Algorithm 2, takes two inputs, namely the *Firmware_Code* representing the microgrid's firmware, that may contain unauthorized modifications, and the *Sampling_Interval* defining the time interval for data collection. The algorithm begins by defining the function *Disassemble*, which takes a single input parameter '*line*', representing a line of code from the microgrid's firmware. It initializes three variables, '*instruction*', '*op_code*', and '*operand*', as empty strings to store the components of the disassembled low-level assembly instruction, as indicated by 2. The *split_line* function extracts the operation code and operand from the code line and stores them in *op_code* and *operand*, as shown in line 3. The instruction is formed by combining the opcode

Algorithm 2 Attack Detection Framework

Input: *Firmware_Code, Sampling_Interval*
Output: *Malware_DetectionModel*

```
1: function Disassemble(line)
2:   instruction, op_code, operand ← empty_string
3:   op_code, operand ← split_line(line)
4:   instruction ← op_code + operand
5: return instruction
6: AssemblyCode ← Disassemble(Firmware_Code)
7: hpc ← [ ]
8: for instruction in AssemblyCode do
9:     While Sampling_Interval do
10:      hpc ← Current_hpc_val
11:      counter ← 0
12: end for
13: Malware_DetectionModel ← TSF_Classifier(hpc)
14: Return Malware_DetectionModel
```

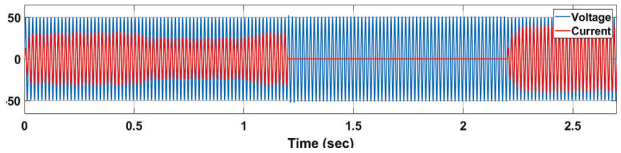
and operand, as represented by line 4. The *Firmware_Code* is disassembled into assembly code using the *Disassemble(Firmware_Code)* function, which translates high-level code into low-level instructions that the custom-built HPCs can monitor, as shown in line 6. During the execution of the loop, the custom-built HPCs continuously monitor and record their values at regular intervals defined by the *Sampling_Interval* and the recorded HPC data is stored in the HPC list (lines 8-12). The time series representation is constructed from the collected HPC data, capturing the temporal patterns and trends of the hardware performance over time by utilizing the function *TSF_Classifier*. The temporal features are then used as input to a time series classifier to build the *Malware_DetectionModel*, as shown in line 13. By learning from temporal patterns in the custom-built HPC data and extracting essential features, the trained *Malware_DetectionModel* becomes capable of detecting potential malicious firmware modifications.

V. Results

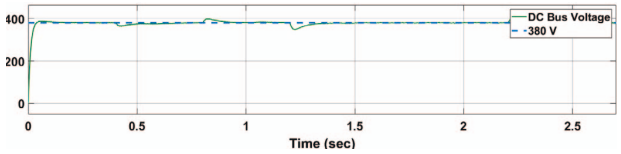
In this section, we present the effectiveness of our hardware-aided method that employs HPCs in identifying alterations in firmware with malicious intent. The detection accuracy of of our proposed model makes it a feasible candidate for practical implementation in power grid systems. To substantiate the proficiency of our approach, we conducted our experiments using a TI TMS320F28379D board for running the firmware responsible for voltage sensing. We curated attacks that incorporated DoS and the introduction of incorrect inputs aimed at instigating aberrant shifts in the normal behavior.

A. Attack Impact Analysis

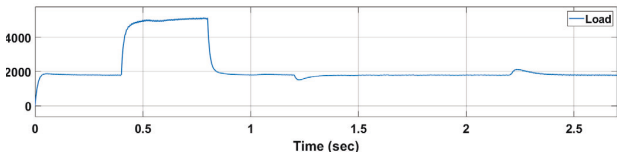
Normal operation of the ER microgrid is depicted in Figure 3, where Figure 3a depicts the grid operation (voltage and current scaled down for representation) at 60Hz, Figure 3b depicts the DC bus maintained constant at 380V by the ER and Figure 3c shows the ER responding to changes in the load in the system. However, when the grid is under attack (Interrupt Operator or Signal Sensing), the normal operation of the the ADC module is interrupted. Therefore, the PWM generation is either stopped completely or intermittently leading to abnormal operation shown in Figure 4a. The interruption occurs at 0.2s which disrupts normal PWM generation and hence, the grid operation. Figure 4b depicts the grid completely out of sync with the DC bus being unsteady and not operating at 60 Hz. The system never recovers from this attack and the grid remains out of sync. Similar behaviour is expected in ADC window attack where the incorrect precision will cause abrupt PWM signals. In Loop attack, the system gets stuck in one operational mode i.e. meeting one load requirement and not modifying its PWM for the load change.



(a) Grid Behaviour vs Time

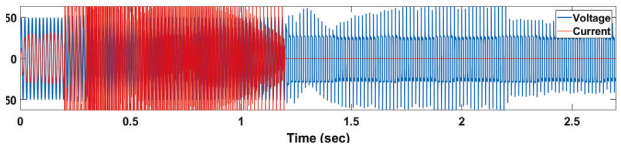


(b) DC Bus Voltage vs Time

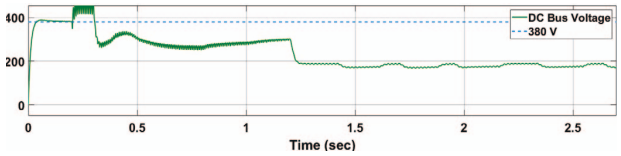


(c) Total Load vs Time

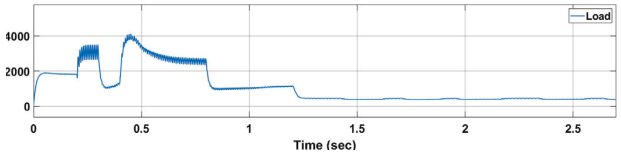
Fig. 3: Microgrid Under Normal Operating Conditions.



(a) Grid Behaviour vs Time



(b) DC Bus Voltage vs Time



(c) Total Load vs Time

Fig. 4: Microgrid Under Attack.

B. Experimental Setup

In order to generate the signatures based on HPCs, we employed a Python script designed to quantify the occurrence of our custom-built HPCs in the disassembled firmware executable. To convert the binary executable into assembly code, the TI dis2000 disassembler was utilized. The matching of HPCs occurs at periodic intervals, hence our script is programmed to accumulate the total count for each designated HPC at the end of every 50 assembly instructions. We settled on this sampling rate as it strikes a balance between providing an adequate volume of samples and effectively capturing key details about the structural layout of the program. As explained in Section IV-B, we developed four different malicious firmware modifications involving one attacks that utilize DoS properties and two modifications that compromise the signal sensing functionality of the DSP board. The process of data collection incorporated samples from all forms of attacks, in addition to the base code. We utilized 70:30 split of this dataset. In this arrangement, 70% of the dataset was allocated for training purposes, while the residual 30% was reserved for testing. This split ensures a robust training phase while still retaining a significant portion of data for validating the effectiveness of the models. Furthermore, all ML models were implemented using the Sklearn library in Python [17]. Additionally, the TSF classifier was developed using the Sktime library [18].

C. Attack Detection with Traditional ML Classifiers

Three machine learning classifiers were built: Decision Tree (DT), Neural Network (NN), and Random Forest (RF). These classifiers were selected for their effectiveness with discrete features and pattern recognition, these models collectively assess HPC-based signatures for detecting malicious firmware changes. All 30 custom HPCs are employed to construct the classifiers, detecting simulated attacks on the microgrid model. Although processors can include multiple HPCs, only few of them can be monitored simultaneously. In order to reduce the set of HPCs (being monitored), we use PCA, a feature selection technique managing high dimensionality data with multiple variables. Since most modern processors are only capable of tracking a couple of HPCs at a time, we used PCA to determine the top three best features: '*l*', '*al*', and "*s*". These features are robust for malicious firmware detection, as they can adequately capture many of the malignant tendencies found in pernicious firmware. Compromised firmware will attempt to deviate from the normal control flow of the system. Consequently, this will increment the number of branch instructions and engenders other malicious activity, such as manipulating outputs or internal system functions which increments boolean and arithmetic instructions. Moreover, reducing the number of custom-built HPCs aid in decreasing the hardware overhead of the DfS architecture. These three dominant features were used to train a new set of models. Figure 5 shows the performance metrics for each classifier after applying PCA. When evaluating our results, all the incorporated models had a minimum of 74.13% accuracy, which indicates that our proposed custom-built HPCs are generalizable.

The DT and RF classifiers have a peak accuracy of 74.13% and 75.86%, respectively, and peak precision of 89.58% and 100%, respectively. While all models could detect the firmware attacks, the NN is able to detect, with high accuracy and precision, all the malicious firmware samples. Our best result was from the NN that furnished a 77.58% accuracy, 100% precision, and 77.6% recall, as seen in Figure 5. Since false positives are a major issue in traditional HPC-based detection, our high precision addresses this issue. Moreover, this demonstrates the benefit of PCA feature selection utilized in our models. When utilizing 30 HPCs, the model becomes overfitted, but employing PCA enabled us to attain the optimal set of HPCs required to construct an effective model. Only three custom HPCs are sufficient to provide a robust defense against malicious firmware modifications, thus significantly reducing the DfS hardware overhead.

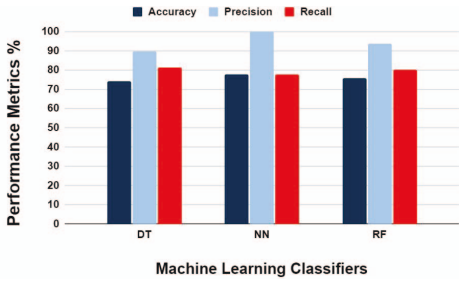


Fig. 5: Performance of ML classifiers with Top-3 Features.

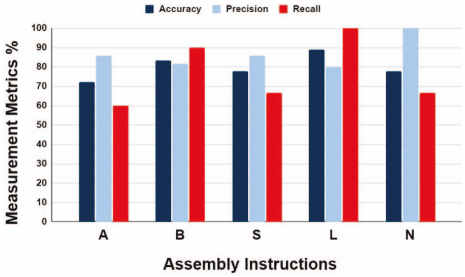


Fig. 6: Individual Assembly Instructions

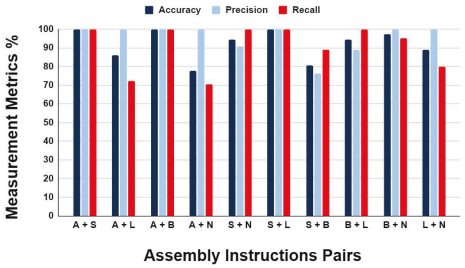


Fig. 7: Pairwise Combinations of Assembly Instructions

D. Attack Detection with Time Series Classifier

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. To evaluate the effectiveness of utilizing 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 Figure 6, we present the accuracy, precision, and recall when the model is trained on these individual assembly instructions. We determined the accuracy when utilizing each of the main assembly instructions, arithmetic *a*, branch/jump *b*, store *s*, load *l*, and boolean *n*. From our results, the Neural Network (NN) model furnished the highest accuracy of all classifiers for each employed custom-built HPC. The average furnished accuracy was 80.7%, which is an increase of 4.2% when compared to the traditional results of the NN. Our best results were furnished when the model was trained using custom-built HPC '*l*'. Specifically, we obtained an accuracy of 88.88% and recall of 100%. Moreover, the precision was 80% which is imperative as we did not incur many false positives and demonstrates the benefits of utilizing sequential attributes when employing a single feature. We only assess the NN model because our results from Figure 5 demonstrate that the NN had the highest furnished accuracy among all traditional models. This shows that time series-based classification provides better results by utilizing just the main assembly instructions as compared to the traditional ML models which had been optimized by PCA.

Figure 7 shows our results for all possible assembly pairs utilized as features for the TSF. The average furnished accuracy, precision, and recall was 91.94%, 95.6%, and 90.64%, respectively. Our best results were obtained when using custom built HPCs *a* and *s* where the model furnished 100% accuracy, which is an 23.91% increase over the average accuracy from Figure 5. Moreover, when employing custom-built HPCs, *a* and *s*, *a* and *b*, *s* and *l* together pairwise, our model 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 model can accurately distinguish between the attacks and benign behavior when using custom-built HPCs in conjuction with TSF. 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. Regardless of whether we utilized a single HPC or multiple HPCs, our results have demonstrated the beneficial advantages of time series detection over traditional ML models in order to predict these malicious attacks.

VI. Conclusion

In this paper, we demonstrated the significant impact of firmware modification attacks on DSP boards within microgrid configurations. In order to mitigate these attacks, we propose a design-for-security strategy using periodic firmware instruction sampling using time series classification. Our approach demonstrates that even devices without native HPC support can be effectively strengthened against adversaries through DfS principles. More specifically, we focused on enhancing the security measures of microgrids through the deployment of custom-built HPCs used as features for time series classification. Our experimental results validate the effectiveness of our methodology, with machine learning models trained on time series data achieving 100% accuracy and precision in detecting firmware modification attacks. In the future, we intend to explore the integration of automated feature selection methods to further optimize and enhance the effectiveness of the proposed methodology.

VII. Acknowledgment

This research is supported by NSF grant #2223046.

References

[1] S. Thakur *et al.*, "Grid forming energy router: A utility interface for renewable energy sources and energy storage grid integration applications," in *IEEE APEC*, 2021.
[2] A. Cagnano *et al.*, "Microgrids: Overview and guidelines for practical implementations and operation," *Applied Energy*, 2020.
[3] V. Lou, "Application behavior based malware detection," 2010.
[4] M. Ozsoy *et al.*, "Hardware-based malware detection using low-level architectural features," *IEEE Transactions on Computers*, 2016.
[5] L. Uhsadel *et al.*, "Exploiting hardware performance counters," in *2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2008.
[6] S. Thakur *et al.*, "Grid forming energy router: Investigation of load control and stability response," in *PEDG, 2020 IEEE*. IEEE, 2020.
[7] A. C. De Melo, "The new linux'perf'tools," in *Slides from Linux Kongress*, 2010.
[8] R. Bro *et al.*, "Principal component analysis," *Analytical methods*, 2014.
[9] C. Malone *et al.*, "Are hardware performance counters a cost effective way for integrity checking of programs," in *Sixth ACM workshop on Scalable trusted computing*, 2011.
[10] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH computer architecture news*, 2013.
[11] X. Wang *et al.*, "Reusing hardware performance counters to detect and identify kernel control-flow modifying rootkits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015.
[12] I. Zografopoulos *et al.*, "Time series-based detection and impact analysis of firmware attacks in microgrids," *Energy Reports*, vol. 8, 2022.
[13] A. Rohan *et al.*, "Can monitoring system state + counting custom instruction sequences aid malware detection?" in *IEEE ATS*, 2019.
[14] A. P. Kuruvila *et al.*, "Hardware-assisted detection of firmware attacks in inverter-based cyberphysical microgrids," *International Journal of Electrical Power & Energy Systems*, 2021.
[15] H. Ismail Fawaz *et al.*, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, 2019.
[16] H. Deng *et al.*, "A time series forest for classification and feature extraction," *Information Sciences*, 2013.
[17] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, 2011.
[18] Löning *et al.*, "sktime: A unified interface for machine learning with time series," *arXiv preprint arXiv:1909.07872*, 2019.