Shells Bells: Cyber-Physical Anomaly Detection in Data Centers

Lars Wüstrich*, Sebastian Gallenmüller*, Stephan Günther*, Georg Carle*, Marc-Oliver Pahl†
Technical University of Munich*, IMT Atlantique†
wuestrich,gallenmu,guenther,carle@net.in.tum.de*, marc-oliver.pahl@imt-atlantique.fr†

***Abstract*—Monitoring the side-channel sound can improve anomaly detection (AD) in data centers (DCs). However, a DC's dense setup results in a composite soundscape which makes it difficult to attribute sounds to individual devices.**
We propose a novel cyber-physical AD approach that validates device activity in realistic composite audio signals. By leveraging information from management network traffic, we predict changes in the DC soundscape. We use a convolutional neural network to compare our predictions with real observations to validate correct device activity and identify anomalies. Our evaluation using data from a real DC environment identifies spoofed and masqueraded activity with an accuracy of 98.62 %.

I. Introduction

Data Centers (DCs) provide essential services that enable our daily lives [4]. The hosted devices must work as intended for an optimal DC operation. Thus, monitoring is crucial for DC management. Operators combine various techniques to monitor and manage DC devices [51]. Tools like the Intelligent Platform Management Interface (IPMI) [27] allow operators to remotely interact with devices without being physically present. This enables the orchestration of critical operations involving multiple devices such as booting device clusters. While remote capabilities simplify DC management, they introduce new problems. Due to faults and attacks [10] devices may report a wrong state. This includes reporting of non-existent (spoofing) or masquerading of existing events [5], falsifying information in the monitoring. The result is a loss of view in the monitoring and sub-optimal management, potentially leading to outages.

Monitoring that tracks DC device status needs to identify when devices report wrong states. In addition to the digital device state, DC monitoring includes physical properties like the power usage of the hosted devices [11], [32], [51]. However, DCs are dense and interconnected setups [51] where multiple devices simultaneously impact physical measurements. This results in composite signals and makes attributing physical changes to individual devices difficult. Thus, DC environments render the existing approaches that monitor physical properties of isolated devices infeasible. This raises the question: *How can we validate device activity in composite DC signals for anomaly detection (AD)?*

Remote management also provides an opportunity to enable AD on composite DC signals such as their soundscape. Control traffic for device management is an additional source of information that indicates future device behavior.

Our paper has three main contributions. (1) We present a novel cyber-physical approach for AD in DC environments. We correlate IPMI requests and changes in a composite audio signal to validate individual device activity. Sound is less popular for monitoring DCs than other side-channels (SCs), such as power consumption or temperature [51]. It is challenging since it is affected by environmental reflections and is easily influenced by multiple sources, leading to a composite signal. Using information from IPMI helps to predict when and how device activity affects the DC soundscape. Our approach addresses the limitations of other SC monitoring methods which require isolated devices [3], [23]. (2) We evaluate our system in a real DC environment. We show that acoustic side channels provide a an easy-to-implement method with advantages over existing approaches [11], [24]. Our evaluation also shows that it is possible to train a machine-learning model with semi-synthetic data for a real world deployment. (3) We open-source our implementation [50].

Section II compares related work. Section III introduces the properties of DC environments and challenges of acoustic side channels. Section IV presents our approach for cyber-physical AD for DCs, using IPMI and acoustic recordings. Section V shows our evaluation in a real-world setup. Section VI discusses generalizability and limitations.

II. Related Work

Side-Channel Based AD and Verification: The majority of the publications using SC monitoring verify the control flow integrity of code execution. In contrast to monitoring code execution, our approach validates correct physical behavior. Han et al. [23] and Aubel et al. [48] measure electromagnetic (EM) emissions to validate the correct control flow of programs. Boggs et al. [8] and Nazari et al. [39] use an EM SC to detect malicious code execution. Han et al. [22] and Lui et al. [33] analyze power intake to verify the correct execution of software. Dupuis et al. [16] investigate power SCs to identify hardware trojan horses. Bolboacă et al. [9] and Formby et al. [17] measure command execution times to detect anomalous behavior. They exploit the deterministic behavior to implement a timing SC for AD.
Our work monitors device behavior when executing a physical action in a composite signal. Closest to our work is [5], in which Birnbach et al. combine measurements from multiple sensors to verify the execution of reported physical events, such as an opened window, in IoT environments. The authors

combine multiple SCs to detect spoofed events. Our work validates specific device activity.

Multi-Layer AD: Several works show the benefits of using information from multiple layers for AD. Sahu et al. [43] fuse cyber and physical information to identify false command and measurement injections in power systems. Yang et al. [52] analyze the power consumption and perceived state of ICS devices to identify attacks. Carcano et al. [12] model MODBUS traffic and command values to track if command chains cause critical states. Similar to our work, Choi et al. [15] create a physical and a cyber model of robotic vehicles. Mismatches between those models indicate abnormal behavior. Our approach uses network control traffic to dynamically predict expected changes and compare them with real measurements for AD.

Audio-Based AD: Most works using acoustic SCs target isolated device activity. Audio is used in various SC attacks [2], [21]. Applications monitoring audio are often related to manufacturing: Al Faruque et al. [1] and Hojjati et al. [25] reconstruct a product's manufacturing processes and shape from audio recordings, Müller et al. [38] present a method for AD on audio spectrograms via image recognition to identify anomalous device behavior, Bayens et al. [3] and Chhetri et al. [14] use an acoustic SC and control signals to detect attacks on 3D printers. By combining measurements with network traffic, our approach is less intrusive than analyzing local code execution and utilizes the activity of multiple devices at once.

AD in DCs: Most AD approaches for DCs rely on network traffic monitoring or collect information on end hosts. Garg et al. [18] present an ML-based approach to detect anomalies from DC network traffic but they do not consider physical device behavior. Borghesi et al. [11] monitor the power consumption of DC devices. The authors train an autoencoder to learn the relationship between physical and software properties during operation in an IPMI-managed DC to detect anomalous power consumption states. Wüstrich et al. [51] analyze the spectrogram of a DC soundscape to identify single device activity and error codes without access to technologies such as IPMI.

Only a few ADS in DCs use information from multiple layers. Hernandez et al. [24] propose to combine information from SNMP, motherboard sensors, and external power distribution units to detect malware on single devices. Their approach requires high temporal and spatial resolution for reliable malware detection. Our solution exceeds these proposals by combining information from multiple layers and actively predicting changes due to control traffic.

III. Environment Characteristics

DC Environments: DC environments have unique properties. The common remote management [10] leads to rare human interaction. Therefore, all environmental noise and physical changes are due to known device activity. The soundscape of DC environments is mostly constant. This is due to the constant spinning of server fans and running climate control while devices have a constant load. Unusual device activity like power-on operations stand out [51].

A common tool to manage DC devices is IPMI [19], [27]. IPMI is a UDP-based request-response protocol for querying device information or executing commands. Commands include power-on or power-off instructions, or request for device status. An IPMI device returns the requested data or a status code indicating the success or failure of an executed command. IPMI commands have a distinct *netfunction (netfun)*, command code, and command bytes. Depending on the success of the execution, devices send out specific response codes indicating the result [27]. Each communication has two phases. The first phase is a handshake, which uses a pre-shared key (PSK) to derive a shared secret defined by the standard as $K2$ [27]. After the handshake, the devices use $K2$ to encrypt and integrity protect their communication [27] in the second phase. Since operators know the PSK, observing a handshake from a vantage point is sufficient to calculate $K2$. By using $K2$, we can decrypt and parse the IPMI traffic.

Acoustic Side-Channels—Advantages and Challenges: Sound is a byproduct of many physical activities [3] and can propagate over several meters. Events create distinct patterns in spectrograms [41] allowing an identification. Thus, a single microphone can monitor activity of multiple devices. Microphones can be added retrospectively without affecting existing infrastructure and measure the environment non-intrusively. This allows to collect information without influencing the monitored devices. In addition, microphones are more affordable than other measuring equipment, such as heat cameras, electromagnetic emission (EM)-measurement devices, or power meters. Even cheap microphones provide high resolution at a high sampling rate [51].

Using acoustic SCs is also challenging. Reflections, obstacles, and the openness of a space influence the spreading of sound waves. If an audio signal contains sound from multiple nearby devices, isolating the sound of individual devices is challenging. Acoustic channels, therefore, require a thorough pre-processing and often additional information to isolate individual activity. Since sound waves easily mix, a major challenge is noise filtering. Noise can be static or dynamic. Several techniques exist to filter noise from an audio signal [36]. Active noise-canceling techniques aim to remove monotonous noises or are performant in a narrow band of the complete frequency spectrum. The removal of dynamic sound requires complex methods [42]. These require additional information about the nature of a sound and its source, such as information about devices in proximity that perform actions. Unknown noise sources include all sounds for which no such information is available. Knowledge about the scenario helps to address these challenges. Bandpass filters limit the frequency band to the expected spectrum of sound. This can help to improve the performance of advanced filtering methods and focus on a specific activity.

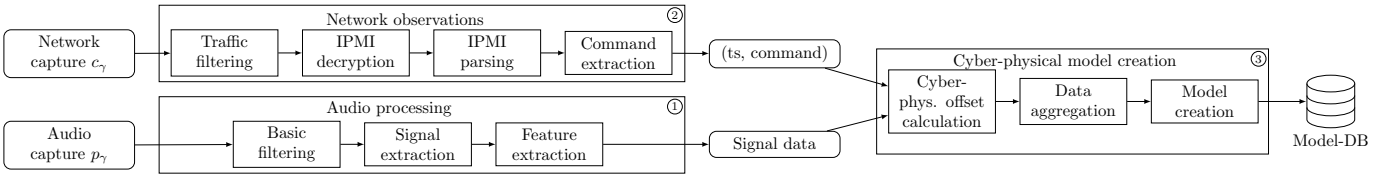

Fig. 1. For creating a sub-model, we first extract layer-specific features from c_γ and p_γ. Then, we fuse the information into a cyber-physical model.

IV. Sound-Based AD for DCs

This section presents our approach for sound-based AD for DC environments. We combine IPMI traffic and audio analysis for AD. When an operator issues a command, IPMI delivers it to its destination via a network. The receiver then executes the command as an action. This causes physical changes on devices, such as the sound of fans spinning up of booting servers (cf. Fig. 2). Thus, IPMI traffic can indicate future changes in the DC soundscape.

Due to their unique properties, DC environments are suitable for this approach. In particular, it allows to use audio signals to identify device activity [51]. Sound can indicate issues such as faulty fans early before they are detected by other means. All intended changes in the environment are due to known device activity. Actions such as power-on of servers, create specific patterns in the spectrogram of the soundscape [41], setting them apart from constant noise.

Similar to other AD systems [13], our approach consists of a model generation and AD phase. The model generation phase builds a global reference model. It correlates changes in the soundscape and IPMI commands, characterizing expected behavior. During the AD phase, we use the global reference model to predict and validate changes in the DC soundscape. If the prediction and the observations mismatch, DC devices behave differently than expected. In addition to validating "normal" activity, this allows to highlight unexpected behavior in the monitoring. This includes the non-execution of instructed and reported activity (spoofing) or the execution of uninstructed activity (masquerading) [5]. Both result in wrong information in the monitoring. Abnormal execution times can also indicate issues [17]. Thus, we consider early or delayed execution of instructions.

A. Reference Model Generation

The global reference model consists of multiple sub-models. We refer to the global reference model as model-DB, storing all sub-models. Modularization has the advantage that changes to the system do not require retraining the global model. Each sub-model maps an IPMI command to audio, resulting from the corresponding action. We identify an IPMI command by its netfun, command code, command bytes, and the carrying packet's L3 information. The inputs for the model generation are an audio recording and a network capture. In the beginning, the ADS processes the inputs separately. The following describes the process for creating one sub-model. Figure 1 visualizes the procedure.

Audio Processing. In an optimal case, the recording only contains the sound of the action without noise. While this is, in many cases, not practical, it is possible to remove static noise from audio. Due to the static DC soundscape, activity of single devices stands out [51]. Thus recording actions without external changes to the soundscape is sufficient.

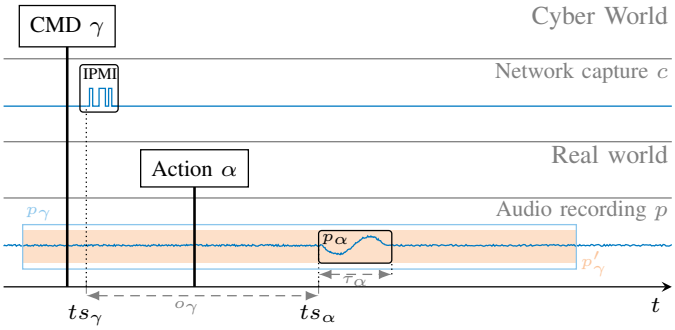

Fig. 2. Schematic relation between IPMI and changes in the soundscape

Since physical measurements are noisy, we need to reduce background noise. This is necessary to extract accurate information about the change due to device activity. Nearby device fans and climate control cause the majority of static background noise [51]. Additionally, device activity does not affect the whole frequency band at once [3]. To reduce the effects of background noise, we limit the considered frequency band of the audio signal to 400-900 Hz, which the DC devices affect via low- and high-pass filters.

We then use the *Root Mean Square Energy (RMS Energy)* as an indicator to automatically identify the time frame of device activity [51]. We use this method to extract the timestamp ts_α and duration τ_α of the activity.

We further calculate a noise profile characterizing the static noise in the recording before ts_α. Using the noise profile, we remove static noise from the recording. These steps result in a noise-free recording. The final step extracts the audio between ts_α and τ_α from the noise-free recording. The extracted portion characterizes the audio of the activity.

Command Extraction. The model generation simultaneously analyzes the network capture to extract the IPMI command. After analyzing the IPMI handshake between source and destination, we derive the shared secret $K2$. We extract netfun, command code, and command bytes from the following IPMI request. We also collect information about the receiver via the L3 destination address. The timestamp ts_γ specifies the time of the IPMI command.

Event Correlation. We now need to correlate the events from the two inputs. We use temporal information for this task. We calculate the offset o_γ between the IPMI command and change in the soundscape to create the corresponding mapping.

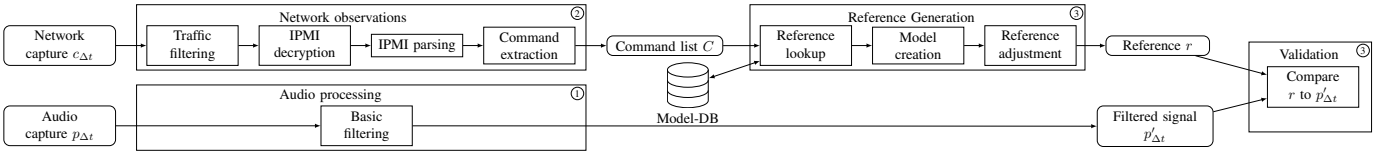

Fig. 3. The AD phase creates a dynamic reference from control traffic using the model-DB to construct a reference for comparison with real observations.

Figure 2 also visualizes this offset. We calculate the offset as $o_\gamma = ts_\alpha - ts_\gamma$. Capturing all data on the same device can prevent synchronization issues [43]. We address this effect in the validation step (cf. Section IV-B).

DCs often physically and logically separate management traffic and have a static setup. The DC infrastructure over-provisions network resources for management traffic to ensure a fast and reliable method for operator device access. Thus, the network typically routes IPMI packets via the same hops. This keeps delivery times of IPMI commands constant. Therefore, the network-induced delay in this scenario is negligible, similar to Industrial Control System (ICS) scenarios [17]. This makes the physical behavior of DC devices predictable. However, the time between receiving and executing commands can have minor variations that need consideration during AD. A sub-model comprises all information from audio processing, IPMI command, and correlation.

B. Anomaly Detection

In the AD phase, our system validates that recorded audio is consistent with the observed IPMI traffic. Therefore, we first dynamically construct a reference signal containing all expected audio changes by using the information in the model-DB. We then compare the real recordings with the reference signal. If the recordings match, we validate the activity as normal. If the observation is inconsistent, it classifies it into our four anomaly types: spoofing, masquerading, early, or delayed execution. In the following, we describe the validation of a time frame Δt. Figure 3 shows the procedure. During the AD phase, the time frame moves as a sliding window for continuous monitoring.

To validate Δt, we first construct a reference signal r. Thus, we analyze the IPMI traffic during Δt. By using the PSK and observed handshakes, we decrypt and parse the IPMI control traffic. We extract all IPMI requests γ and their timestamp ts_γ and aggregate them in a list C.

We use the list to dynamically construct r. This distinguishes our approach from others which compare observations to fixed prerecorded traces. We initialize a silent r and synchronize the start and end of r with Δt. Then, we look up each IPMI request γ in C in the model-DB. The model-DB contains all γ that affect the soundscape. If the model-DB does not contain γ, we assume it does not affect the audio signal. Thus, we ignore all γ that are in C, but have no reference in the model-DB. If γ exists in the model-DB, we know its effects on the soundscape. We know ts_γ of the command from C. By using the offset o_γ and reference from the model-DB, we can predict the sounds due to γ in r. Thus, we add the expected audio of γ from the model-DB to r at $ts_\gamma + o_\gamma$. By repeating this process for all $\gamma \in C$, we construct r. If there is no γ in C, the reference r is silent since we do not expect any changes. Figure 5 shows an example of a constructed audio reference with a predicted server power-on compared to an actual recording.

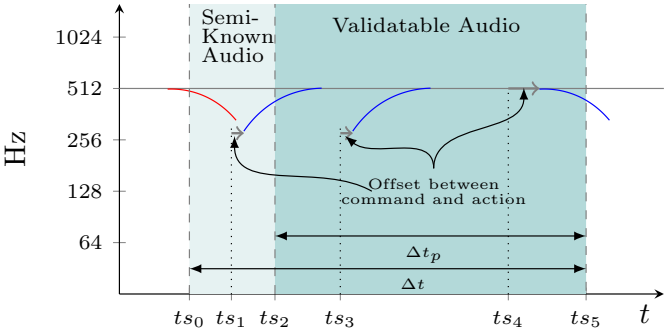

Fig. 4. The time frames Δt, Δt_p, and events in an example recording

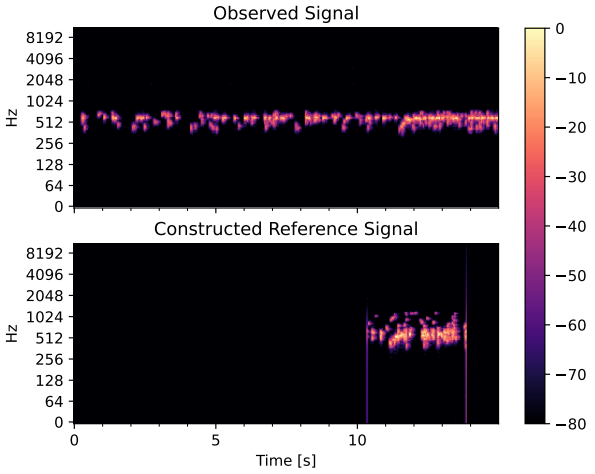

Fig. 5. A real signal and its reference with a server turning on at 11 s

We can only validate a part of r during Δt. This is because the execution of an action, e. g. booting a machine takes time. Thus, the recording during Δt can contain partial sounds due to actions that started before its beginning and have not ended yet. We refer to the validatable part as Δt_p. The validatable part Δt_p starts after an offset to the start of Δt and ends simultaneously as Δt. If an α started before the beginning of $p_{\Delta t_p}$ but has not yet finished, $p_{\Delta t_p}$ includes a part of α at the beginning. The offset has to be long enough such that a generated reference includes all known sources that could influence $p_{\Delta t_p}$. The time-frame during the offset is semi-known since it can contain known and unknown activity. Thus, we keep only the validatable part Δt_p for the AD. Figure 4 visualizes the time frames.

We finally compare r to the recording. The example in

TABLE I: SIMILARITY CALCULATION METHODS FOR AUDIO SIGNALS

Method	Noise	Temporal	Comparison	Reasoning
Fingerprinting	✓	✗	✓	% matching hashes
DTW	✗	✓	✓	distance
RMS Energy	✗	✓	✓	distance
Image recognition	✓	✓	✗	events
ML	✓	✓	✓	events/classes

Figure 5 shows that r and the recording are similar, but not identical. The recording contains remaining noise, and the power-on trace is later than predicted. This emphasizes the need for a comparison method that is robust to noise and temporal shifts. Temporal shifts occur due to the lack of time guarantees in DCs. Thus, it is only possible to estimate when a device executes an action. The comparison method should validate a time frame even if minor shifts occur. While our method applies various noise filters throughout the process, some noise can remain in the recording. Therefore, the chosen comparison method must recognize similarity despite remaining noise. There are a variety of methods to compare audio snippets. Audio fingerprinting [49], dynamic time warping (DTW) [37], image recognition [7], and machine learning (ML) [29] are common techniques. We compare and motivate our chosen comparison and validation method in the following.

Audio fingerprinting offers noise robustness [49]. However, temporal shifts lead to low similarity of the fingerprint. This effect is measurable for shifts by $\geq 10\,\mathrm{ms}$ to its reference counterpart. This makes audio fingerprinting only suitable for deterministic environments in which it is possible to accurately predict device activity or to compare fixed snippets.

DTW calculates a distance between signals [37]. It is robust to temporal shifts and speed variations. However, noise can result in a high distance calculation of DTW, even if most of the signal is similar [44], affecting its performance.

It is also possible to compare aggregated signal information, e. g., the **RMS Energy** [51]. Noise affects RMS-Energy-based comparisons similar to DTW. Since the RMS-Energy is an aggregated value, it is useful to identify time frames when changes occur. Aggregating information into a single value makes it impossible to identify which frequencies changed. This makes it challenging to distinguish activities that cause similar changes to the aggregated value.

Image recognition like [7] identifies events that show up as geometric patterns [41] in spectrograms. These patterns must stand out from the background in a spectrogram. Noisy environments, such as DCs, alleviate the contrast of patterns to the background and reduce the efficiency of such approaches. While image recognition can identify events in spectrograms, it does not aim to compare two complete spectrograms.

ML methods can be robust to noise and temporal shifts and can even identify melody variations [29]. Depending on the ML method, it is possible to classify the result. A drawback of this approach is that the acceptable delay needs to be specified at training time.

Table I presents our comparison. Due to the robustness, our validation relies on an ML-based approach. However, the high resolution and audio sampling rate lead to high dimensional input data for ML. Processing such information with ML requires data preprocessing and preparation.

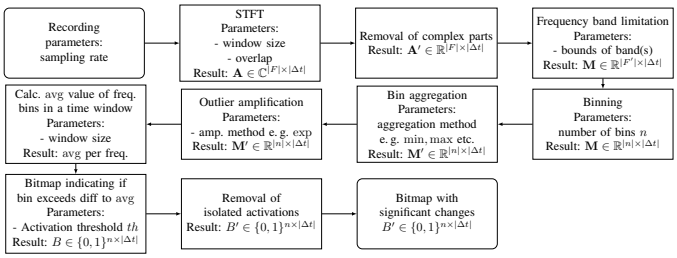

Fig. 6. Preprocessing chain of audio input for the ML validation

Our approach applies a procedure to reduce the dimensionality of the audio input while retaining important information about significant changes. Figure 6 visualizes the procedure. In general, the procedure reduces the resolution of the frequency band and identifies deviations from the average energy on a frequency. The process initially performs a *Short Time Fourier Transform (STFT)* on the audio signal. Similar to other works [20], [41] we choose a window size of 2048 for the STFT. $|\Delta t|$ denotes the number of STFT time frames of the audio during Δt and $|F|$ the frequency band size. The result of the STFT is a matrix $\mathbf{A} \in \mathbb{C}^{|F|\times|\Delta t|}$ holding Fourier coefficients as elements. A $15\,\mathrm{s}$ recording at a sampling rate of $44.1\,\mathrm{kHz}$ leads to $|\Delta t| = 646$, each frame covering $23.21\,\mathrm{ms}$. An $a_{ij} \in \mathbf{A}$ describes the amplitude of frequency $i \in F$ of a frame $j \in |\Delta t|$. Since the elements of $\mathbf{A}$ are complex, ordering them is impossible. To compare the $a_{ij} \in \mathbf{A}$, the next step takes their absolute values $|a_{ij}|$ to create a matrix $\mathbf{A}' \in \mathbb{R}^{|F|\times|\Delta t|}$. Since device activity does not affect all frequencies $f \in F$ [51], the process reduces the frequency band to F'. The reduced frequency band F' is a subset of F, e. g., the lower quarter. This results in $|F'| < |F|$ and a matrix $\mathbf{M} \in \mathbb{R}^{|F'|\times|\Delta t|}$. The first row of Figure 6 shows these steps. To further reduce the dimensionality, we aggregate the frequency band F' of each time step $j \in \Delta t$ into n bins. The number of bins $n \in \{1, ..., |F'|\}$ influences information loss due to value aggregation. When $n = 1$, all frequencies in F' are aggregated into a single bin. If $n = |F'|$, all frequencies in F' are considered individually. This results in a tradeoff between the resolution and dimensionality of the input. After a series of experiments, we choose $n = 128$.

The method calculates a single value for each bin using aggregation methods such as $\max, \min, \mathrm{avg}$, or sum. We calculate the avg of each bin as representative. This step makes the method susceptible to noise causing random spikes in a bin. We address those outliers in a later step. The remaining steps aim at identifying significant changes in the signal. Each representative element of a bin is amplified to separate outliers. The amount of amplification depends on the scenario. While separation is important, our method limits the amplification effect via a scenario-specific ceiling for the later steps.

The following steps consider individual frequency bins b_{fj} of a frequency f during all $j \in |\Delta t|$. To identify outliers on a frequency bin b, the procedure calculates the average

value of each frequency bin over Δt as $a_{b_f} = \sum_{j=0}^{|\Delta t|} \frac{b_{fj}}{|\Delta t|}$. The second row of Figure 6 represents these steps. It then calculates a distance $|b_{fj} - a_{b_f}|$ and compares it to a threshold th. If $|b_{fj} - a_{b_f}| \leq th$, it is marked as 0, otherwise as 1. The sox utility filters noise in a similar way [28]. The choice of th depends on a variety of factors. One factor is how many events stand out from the background noise. A lower difference to the background demands a smaller th as otherwise expected bits do not get activated. The second factor is the number of events in Δt. More events increase the average, reducing the difference between peaks and the average. Therefore, th needs to be set accordingly. The method stores all marks in bitmap $B \in \{0,1\}^{n \times |\Delta t|}$ highlighting the changes.

Random noise spikes can impact the calculation of bin representatives and cause wrongly activated bits in the bitmap. These occur randomly and in isolation. Similar to Pham et al. [41], we found that activity causes patterns, i. e., larger, connected, and activated fields. To remove random noise, we deactivate isolated bits. We define an isolated bit by the maximum number of activated bits in proximity. The choice of this number requires thorough consideration. A small maximum activation number can lead to noise remaining in the bitmap. An activation number that is too high leads to the false removal of action patterns. In particular, small patterns, e. g., caused by a short beep, can be wrongfully filtered out if the activation number is too high. The definition of proximity requires a similar discussion. A too-small proximity distance dismisses valid signals, and high values fail to filter noise. This results in the final bitmap $B' \in \{0,1\}^{n \times |\Delta t|}$.

The procedure highlights changes relative to background noise. This makes it adaptive and robust to changes in the base level of background noise. The preprocessing and signal construction for the ML-based validation creates two bitmaps. Each bitmap highlights the changes in one audio artifact—the reference and the recorded signal. Since bitmaps represent images, we use a convolutional neural network (CNN). Various image processing applications show that CNNs are particularly suited for tasks with image-like input. The structure of our CNN is similar to the LeNet [30] and ConvNet [31]. We use four convolutional layers with max-pooling layers after the second and the fourth layer. There are two more fully connected layers after the second pooling layer for the classification. The input for the CNN is a concatenation of both bitmaps into a validation bitmap. In our setup, the size of the validation bitmap is in $\{0,1\}^{256 \times 646}$. The CNN categorizes the input into "normal" or one of our four anomaly types.

By using this approach, it is possible to validate individual device activity in a composite audio signal.

C. Application-Specific Challenges

Acoustic SCs introduce a variety of challenges. (1) Sound wave dispersion depends on the environment. Therefore, references for the model-DB need to be created within the monitored system. Changes in the environment, e. g., the setup of new structures that affect sound propagation, can require a new training of the model-DB. (2) Audio filtering is complex and requires additional knowledge about an environment. The choice of filters depends on the environment and monitored devices. Filtering introduces information loss. Careful tuning is necessary to avoid filtering out important information. In particular, filtering dynamic sound sources is computationally expensive [7]. Dynamic filtering methods require additional domain knowledge.

V. Evaluation

Our evaluation measures the capabilities of our approach. We first evaluate AD in a controlled environment using synthetic data from a real-world DC. In a second evaluation, we show the functionality in a real setup.

Implementation: We implemented our system [50] in Python, using *librosa* [34] and *(py)sox* [6], [45] for audio processing. The prototype uses *pypacker* [46] to process network traffic. The CNN validation uses *pytorch* [40].

Test Environment: The DC environment consists of 33 servers and 4 actively cooled switches in a single air-conditioned room. We can control all devices via IPMI. There are several types of servers with multiple devices of each type. In the soundscape, the servers mainly differ in the cooling fans, emitting noise on frequency bands [51]. The typical noise level in the DC is around $72.6\,\mathrm{dB(A)}$. The ADS runs on a Linux machine with Debian 11, an Intel Xeon 1265L CPU, and $16\,\mathrm{GB}$ RAM.

Experiment Setup: Our experiments involve three device types: *(i)* a management device issuing commands to servers using *ipmitool* [26], *(ii)* servers that receive IPMI commands and execute corresponding actions, and *(iii)* a monitor that runs our ADS, capturing real-time traffic between the management device and servers. Here, management device and monitor share the same machine. After building the model-DB, our system passively observes the environment. We use a Blue Yeti X [35] microphone in cardioid recording mode. The microphone records the frequency band of $20\,\mathrm{Hz}$ to $20\,\mathrm{kHz}$ at a sampling rate of $44.1\,\mathrm{kHz}$. The microphone is located approximately $0.3\,\mathrm{m}$ in front of the server racks (cf. Figure 7).

Reference Model Generation: To construct the model-DB, we send *shut down* and *power on* IPMI commands. These commands cause server fans to stop or start, changing the DC soundscape. We assume devices execute and operate as intended during the reference model generation. The monitor builds a sub-model for each command (cf. Section IV). The creation takes place *during normal operation*, i. e., without interrupting the general operation. We limit the frequency band to $400\,\mathrm{Hz}$ to $900\,\mathrm{Hz}$, the fan spectrum, to reduce the background noise. Additionally, we apply noise filtering via sox with a strength of 0.05. The model-DB contains 8 references for 4 devices, covering a subset of all devices. We use the same model-DB for the entire evaluation.

Metrics: We evaluate the performance by analyzing commonly used metrics [3], [12]: *true positives (TP)*, *true negatives (TN)*, *false positives (FP)* and *false negatives (FN)*. We derive the *accuracy* $(TP + TN)/(TP + TN + FP + FN)$, the *precision* $TP/(TP + FP)$ and *recall* $TP/(TP + FN)$.

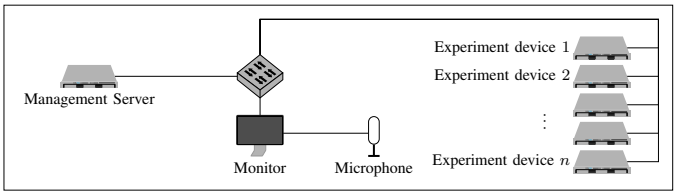

Fig. 7. Schematic experiment setup

TABLE II: VALIDATION RESULTS OF THE CNN MODEL

Class/Classified	Normal	Early	Delay	Spoof.	Masqu.	Recall
Normal (3000)	2947	23	14	8	8	0.9823
Early (1920)	40	1876	0	0	4	0.9771
Delay (1920)	35	0	1884	5	1	0.9813
Spoofing (3840)	27	0	1	3811	1	0.9924
Masquerading (3840)	38	1	0	0	3801	0.9898
Precision	0.9546	0.9874	0.9921	0.9966	0.9963	

A. Synthetic Model Training and Evaluation

Model Training with Generated Data: The generation of sufficient samples to train ML is challenging. If there are multiple actions, the combination of these actions creates an exponentially growing combination space. Recording audio in a real environment requires time, e. g., 1000 recordings with a length of 15 s are equivalent to 4.16 h. Recorded devices are blocked from being used by other users during the data generation. Further, the repeated command execution causes wear and tear. We create a semi-synthetic dataset by rearranging references from the previously generated model-DB. This is possible since we consider deterministic actions. This allows to generate labeled samples from a few recordings for a dataset. Our dataset contains 72600 recordings and references spanning 302.5 h. Samples contain 0 to 2 simultaneous actions. Traces of 1 action show the possibility to identify a single device in a composite signal, and 2 actions show the performance if multiple devices operate concurrently. There are 160 different references for 1 and 80 for 2 actions. This ensures a balance of spoofing and masquerading samples with 1 and 2 actions. We train our CNN on this dataset.

To generate labeled samples, we temporally shift or remove actions. Depending on the change in the recording, we automatically assign a label. "normal" labeled samples contain the same actions in the reference and recording at the same time stamps. The labeling requires a definition of acceptable delay and early execution. In our setup, we consider the execution within a time frame of 0.5 s to the expected execution acceptable and as "normal". If an event shifts 0.5 s to 1.5 s, it has an early or delayed execution label. Temporal shifts of more than 1.5 s are not in the dataset and are considered either spoofing or masquerading. Multiple repetitions of the model-building process show that this is a reasonable assumption. In our experiments, the offset between the IPMI command and change in the soundscape varied in a range of 0.35 s for power-on and 0.6 s for power-off. Samples with references containing more actions than the recording are labeled as "spoofing". Samples with fewer actions in the reference than the recording are labeled "masquerading". The class distribution in the data set is 15000 normal, 9200 samples for each delay and early execution, and 19200 for each masquerading and spoofing. The train-test split of the dataset is 80/20. We train the model for 20 epochs with a learning rate of 0.01. A stagnating loss indicated overfitting after 11 epochs. Our evaluation uses the model trained for 11 epochs.

Results: Table II lists the classification results. The diagonal entries of the table reflect TP, horizontal entries FP, and vertical entries FN classifications. The chosen model has an overall accuracy of 98.62 % on the synthetic test dataset. The model achieves the highest precision for detecting spoofing and masquerading anomalies with a precision of 99.66 % and 99.63 %. The model classifies early and delayed execution with a precision of 98.74 % and 99.21 %. It shows the worst performance at classifying normal conditions with a precision of 95.46 %. The recall for all classes is $\geq$ 97.71 %.
Our evaluation shows a limitation of our approach due to the preprocessing step. Multiple identical operations within the same time frame merge into a single pattern in the bitmaps. Due to this shadowing effect, our CNN fails to distinguish patterns from individual and simultaneous activities,.

B. Real-World Evaluation

Setup: We reuse the setup shown in Figure 7 and simulate the different anomalies. To simulate early and delayed execution of events, we add or subtract 1 s to the offset in the model-DB's sub-models. For the simulation of the event masquerading, we remove the sub-model for a monitored device from the model-DB such that the constructed reference does not contain the expected change. If an IPMI device receives a command to turn into a state it already is in, e. g. power on, it responds with a success message. This way we simulate event spoofing for power-off and power-on.
We create 4 types of time frames: normal with 0 actions, normal with 1 action, spoofing, and masquerading. Due to the missing guarantees for maximum execution times in IPMI, we include "early" and "delayed" action execution in "normal" cases. We assess the CNN classification by visual inspection. We collect 20 normal time frames and for each anomaly. We validate 80 independent 15 s time frames, making up 20 min.

Results: The results in Table III show that the model can accurately classify real-world measurements. The system classifies all 20 of the time frames with 0 actions correctly as "normal". Thus, the model does not classify random noise as spoofed activities. The remaining time frames have at least 1 action in either the reference, recording, or both.
A visual inspection of the 20 time frames confirms 15 of the classifications by the CNN. As expected, the offset between IPMI commands and actions varied. Therefore, the CNN correctly identifies minor deviations from the expectation as either early or delayed execution in 3 out of 3 cases. In 6 other cases, the deviation from the expectation was larger than the acceptable difference of 0.5 s. The CNN correctly classified those time frames as spoofing. In 4 other cases, the CNN falsely classified a time frame as "spoofing".
The model successfully detected 19 of 20 spoofed events. In

TABLE III: DC EVALUATION RESULTS

Class/Classified	Normal	Early	Delay	Spoof.	Masqu.
Normal (40)	25	1	2	12	0
Spoofing (20)	1	0	0	19	0
Masqu. (20)	20 (4)†	0	0	0	0 (16)†

† In combination with RMS Energy.

the misclassified time frame, the model classified the activity once as "normal" due to patterns in the recording of remaining noise. This highlights the importance of filtering and change extraction as input for the validation.

The model did not identify action masquerading. A visual inspection of the bitmaps of the reference and recording shows that the construction method correctly highlights device activity. In 4 cases, the remaining patterns in the recording are indistinguishable from noise patterns, which is why we deem this classification correct. This indicates an issue with the CNN classification. We argue that improved training can address this issue, e. g., if it includes more examples of masquerading. To address this issue, we combine multiple validation methods. By including the RMS-Energy-based approach [51] used in the model building can identify time frames in which activity takes place. If there is no expected activity but the RMS-Energy-based method identifies an action, we correctly classify masquerading in these 16 time-frames.

Table III summarizes the results. In our quantitative evaluation, 57 (73 in combination with RMS-Energy) of the 80 time frames were classified correctly. This shows that the presented approach could successfully validate device activity and identify anomalies in a composite signal.

Sensitivity: To evade detection, anomalies must simultaneously affect the IPMI traffic and soundscape measurements. Action spoofing requires corresponding physical changes. This requires physical access to the environment for active attackers. Action spoofing is possible in combination with action masquerading. By forcing another device to perform an action that causes identical changes, both signals match. There is an analogous approach to evade the detection of action masquerading. One way to address this is to add localization capabilities, e. g. multiple microphones. Another way to avoid detection of action masquerading is to manipulate the execution of actions by limiting physical changes, e. g. lowering the speed of fan deceleration. Since our system filters small changes as noise, an attacker can evade detection. In our setup, we can only detect activities with a signal to noise ratio (SNR) higher than $-4.8\,\mathrm{dB}$. Such low-level manipulations require firmware-level access to the compromised device to alter how a device executes an action. Finally, without localization, it is possible to masquerade actions by shadowing other legitimate actions. Fixed offsets require attackers to accurately predict actions, which is challenging [5].

VI. GENERALIZABILITY AND LIMITATIONS

Generalizability: Our method can be used in other environments with other SCs and protocols. The requirements for an adaption are the observability of commands and predictable environments. It is possible to parse other control traffic, e. g. MODBUS [47]. Similarly, using other SCs, e. g., shared power meters or temperature sensors is possible. While validation methods like audio fingerprinting are SC-specific, ML approaches are flexible and adaptable. Systems that guarantee strict execution times allow stricter validation methods.

As shown in the evaluation, the modular approach allows to create a synthetic dataset that can be used to train a model for ML. Our system can address commands with variations due to additional command arguments, e. g. minor adjustments to fan speeds. Such variations can be identified in the command and can be included as additional sub-models in the model-DB.

Limitations: Our approach requires access to the unencrypted messages to monitor executed commands. The offset between commands and the corresponding activities must be deterministic within a time window. Otherwise, our approach can only reliably predict expected changes. For a successful detection, the monitored actions must stand out compared to noise. Finally, our evaluation shows that the current implementation could be more robust to shadowing. Our solution cannot detect multiple simultaneous executions of the same action. It is also difficult to distinguish actions causing identical spectrogram patterns. This can be addressed by using multiple sensors that enable localization.

VII. CONCLUSION

We presented a novel approach to cyber-physical AD for DCs. Our method leverages IPMI control traffic to validate device activity in a composite signal—the DC soundscape. By utilizing control traffic, we predict changes in the soundscape, making it possible to attribute changes to individual devices. The proposed method is non-intrusive since it does not interact directly with components after generating a reference model. During AD, it rearranges sub-modules of references on demand. Using information of multiple layers for AD makes it easier to identify anomalies. Our evaluation uses real-world data and shows that our system is robust to noise and minor temporal inaccuracies. In experiments with a synthetic dataset we achieve an accuracy of $98.62\,\%$. Experiments in a real DC environment confirm the synthetic results. Our contribution opens up new possibilities for AD in DCs by enabling reasoning about individual devices in composite signals.

ACKNOWLEDGEMENTS

This work was supported by the German Federal Ministry of Education and Research; projects SKINET (16KIS1221), 6G-life (16KISK002), and 6G-ANNA (16KISK107). We received funding from the EU's Horizon 2020 program (SLICES-SC 101008468, SLICES-PP 101079774), the Bavarian Ministry of Economic Affairs, Regional Development and Energy as part of the project 6G Future Lab Bavaria, and from the German Research Foundation (HyperNIC, CA595/13-1). Support was also provided by the industrial chair Cybersecurity for Critical Networked Infrastructures (https://CyberCNI.fr) with support of the FEDER development fund of the Brittany region.

REFERENCES

[1] M. A. Al Faruque, S. R. Chhetri, A. Canedo, and J. Wan, "Acoustic Side-Channel Attacks on Additive Manufacturing Systems," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–10.

[2] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder, "Acoustic side-channel attacks on printers," in *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*. USENIX Association, 2010, pp. 307–322. [Online]. Available: http://www.usenix.org/events/sec10/tech/full_papers/Backes.pdf

[3] C. Bayens, T. Le, L. Garcia, R. Beyah, M. Javanmard, and S. Zonouz, "See No Evil, Hear No Evil, Feel No Evil, Print No Evil? Malicious Fill Patterns Detection in Additive Manufacturing," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1181–1198.

[4] T. Benson, A. Akella, and D. A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010*, M. Allman, Ed. ACM, 2010, pp. 267–280. [Online]. Available: https://doi.org/10.1145/1879141.1879175

[5] S. Birnbach, S. Eberz, and I. Martinovic, "Peeves: Physical Event Verification in Smart Homes," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 1455–1467. [Online]. Available: https://doi.org/10.1145/3319535.3354254

[6] R. Bittner, E. Humphrey, and J. Bello, "Pysox: Leveraging the Audio Signal Processing Power of sox in Python," in *Proceedings of the International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, 2016.

[7] V. Boddapati, A. Petef, J. Rasmusson, and L. Lundberg, "Classifying Environmental Sounds Using Image Recognition Networks," in *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017*, ser. Procedia Computer Science, C. Zanni-Merk, C. S. Frydman, C. Toro, Y. Hicks, R. J. Howlett, and L. C. Jain, Eds., vol. 112. Elsevier, 2017, pp. 2048–2056. [Online]. Available: https://doi.org/10.1016/j.procs.2017.08.250

[8] N. Boggs, J. C. Chau, and A. Cui, "Utilizing Electromagnetic Emanations for Out-of-Band Detection of Unknown Attack Code in a Programmable Logic Controller," in *Cyber Sensing 2018*, vol. 10630. International Society for Optics and Photonics, 2018, p. 106300D.

[9] R. Bolboacă, B. Genge, and P. Haller, "Using Side-Channels to Detect Abnormal Behavior in Industrial Control Systems," in *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2019, pp. 435–441.

[10] A. Bonkoski, R. Bielawski, and J. A. Halderman, "Illuminating the Security Issues Surrounding Lights-Out Server Management," in *7th USENIX Workshop on Offensive Technologies, WOOT '13, Washington, D.C., USA, August 13, 2013*, J. Oberheide and W. K. Robertson, Eds. USENIX Association, 2013. [Online]. Available: https://www.usenix.org/conference/woot13/workshop-program/presentation/bonkoski

[11] A. Borghesi, A. Libri, L. Benini, and A. Bartolini, "Online Anomaly Detection in HPC Systems," in *IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS 2019, Hsinchu, Taiwan, March 18-20, 2019*. IEEE, 2019, pp. 229–233. [Online]. Available: https://doi.org/10.1109/AICAS.2019.8771527

[12] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. N. Fovino, and A. Trombetta, "A Multidimensional Critical State Analysis for Detecting Intrusions in SCADA Systems," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 179–186, 2011.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[14] S. R. Chhetri, A. Canedo, and M. A. Al Faruque, "KCAD: Kinetic Cyber-Attack Detection Method for Cyber-Physical Additive Manufacturing Systems," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2016, pp. 1–8.

[15] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting Attacks Against Robotic Vehicles: A Control Invariant Approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 801–816.

[16] S. Dupuis, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "On the Effectiveness of Hardware Trojan Horse Detection via Side-Channel Analysis," *Information Security Journal: A Global Perspective*, vol. 22, no. 5-6, pp. 226–236, 2013.

[17] D. Formby, P. Srinivasan, A. M. Leonard, J. D. Rogers, and R. A. Beyah, "Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems." in *NDSS*, 2016.

[18] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A Hybrid Deep Learning-Based Model for Anomaly Detection in Cloud Datacenter Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 3, pp. 924–935, 2019. [Online]. Available: https://doi.org/10.1109/TNSM.2019.2927886

[19] O. Gasser, F. Emmert, and G. Carle, "Digging for Dark IPMI Devices: Advancing BMC Detection and Evaluating Operational Security," in *TMA*, 2016.

[20] D. Genkin, N. Nissan, R. Schuster, and E. Tromer, "Lend Me Your Ear: Passive Remote Physical Side Channels on PCs," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, p. 4437–4454.

[21] D. Genkin, A. Shamir, and E. Tromer, "Acoustic Cryptanalysis," *J. Cryptol.*, vol. 30, no. 2, pp. 392–443, 2017. [Online]. Available: https://doi.org/10.1007/s00145-015-9224-2

[22] Y. Han, M. Chan, Z. Aref, N. O. Tippenhauer, and S. A. Zonouz, "Hiding in Plain Sight? On the Efficacy of Power Side Channel-Based Control Flow Monitoring," in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds. USENIX Association, 2022, pp. 661–678. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/han

[23] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch Me, But Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1095–1108.

[24] J. M. Hernández, L. Pouchard, J. T. McDonald, and S. J. Prowell, "Developing a Power Measurement Framework for Cyber Defense," in *Cyber Security and Information Intelligence, CSIIRW '13, Oak Ridge, TN, USA, January 8-10, 2013*, F. T. Sheldon, A. Giani, A. W. Krings, and R. K. Abercrombie, Eds. ACM, 2013, p. 28. [Online]. Available: https://doi.org/10.1145/2459976.2460008

[25] A. Hojjati, A. Adhikari, K. Struckmann, E. Chou, T. N. Tho Nguyen, K. Madan, M. S. Winslett, C. A. Gunter, and W. P. King, "Leave Your Phone at the Door: Side Channels That Reveal Factory Floor Secrets," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 883–894.

[26] IBM. ipmitool. https://github.com/ipmitool/ipmitool. Accessed: 2023-04-10.

[27] *- IPMI - Intelligent Platform Management Interface Specification Second Generation*, Intel, Hewlett-Packard, NEC, Dell, 10 2013, rev. 1.1.

[28] U. Klauer. (2013, 06) Message on sox mailing list: noise reduction algorithm. [Online]. Available: https://sourceforge.net/p/sox/mailman/sox-users/?viewmonth=201306\&viewday=27

[29] M. Krause, M. Müller, and C. Weiß, "Towards Leitmotif Activity Detection in Opera Recordings," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 4, no. 1, pp. 127–140, 2021. [Online]. Available: https://transactions.ismir.net/articles/10.5334/tismir.116/

[30] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for Images, Speech, and Time Series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[31] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[32] M. Levy and J. O. Hallstrom, "A New Approach to Data Center Infrastructure Monitoring and Management (DCIMM)," in *IEEE 7th Annual Computing and Communication Workshop and Conference, CCWC 2017, Las Vegas, NV, USA, January 9-11, 2017*. IEEE, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CCWC.2017.7868412

[33] Y. Liu, L. Wei, Z. Zhou, K. Zhang, W. Xu, and Q. Xu, "On Code Execution Tracking via Power Side-Channel," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 1019–1031. [Online]. Available: https://doi.org/10.1145/2976749.2978299

[34] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th Python in Science Conference*, vol. 8. Citeseer, 2015, pp. 18–25.

[35] B. Microphones. Blue - Yeti X. https://www.bluemic.com/en-us/products/yeti-x/. Accessed: 2023-03-03.

[36] A. A. Milani, I. M. S. Panahi, and P. C. Loizou, "A New Delayless Subband Adaptive Filtering Algorithm for Active Noise Control Systems," *IEEE Trans. Speech Audio Process.*, vol. 17, no. 5, pp. 1038–1045, 2009. [Online]. Available: https://doi.org/10.1109/TASL.2009.2015691

[37] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[38] R. Müller, F. Ritz, S. Illium, and C. Linnhoff-Popien, "Acoustic Anomaly Detection for Machine Sounds based on Image Transfer Learning," in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021, Volume 2, Online Streaming, February 4-6, 2021*, A. P. Rocha, L. Steels, and H. J. van den Herik, Eds. SCITEPRESS, 2021, pp. 49–56. [Online]. Available: https://doi.org/10.5220/0010185800490056

[39] A. Nazari, N. Sehatbakhsh, M. Alam, A. G. Zajic, and M. Prvulovic, "EDDIE: em-based detection of deviations in program execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA 2017, Toronto, ON, Canada, June 24-28, 2017*. ACM, 2017, pp. 333–346. [Online]. Available: https://doi.org/10.1145/3079856.3080223

[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in neural information processing systems*, vol. 32, 2019.

[41] P. Pham, J. Li, J. Szurley, and S. Das, "Eventness: Object detection on spectrograms for temporal localization of audio events," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2491–2495.

[42] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, "An Overview of Lead and Accompaniment Separation in Music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018.

[43] A. Sahu, Z. Mao, P. Wlazlo, H. Huang, K. Davis, A. Goulart, and S. Zonouz, "Multi-Source Multi-Domain Data Fusion for Cyberattack Detection in Power Systems," *IEEE Access*, vol. 9, pp. 119 118–119 138, 2021.

[44] X. Song, Q. Wen, Y. Li, and L. Sun, "Robust time series dissimilarity measure for outlier detection and periodicity detection," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 4510–4514.

[45] SoX. SoX - Sound eXchange. http://sox.sourceforge.net/. Accessed: 2022-07-29.

[46] M. Stahn. pypacker - the fastest and simplest packet manipulation lib for python. https://gitlab.com/mike01/pypacker. Accessed: 2023-03-03.

[47] A. Swales, "Open modbus/tcp specification," *Schneider Electric*, vol. 29, pp. 3–19, 1999.

[48] P. Van Aubel, K. Papagiannopoulos, Ł. Chmielewski, and C. Doerr, "Side-Channel Based Intrusion Detection for Industrial Control Systems," in *International Conference on Critical Information Infrastructures Security*. Springer, 2017, pp. 207–224.

[49] A. Wang, "An Industrial Strength Audio Search Algorithm," in *ISMIR 2003, 4th International Conference on Music Information Retrieval, Baltimore, Maryland, USA, October 27-30, 2003, Proceedings*, 2003.

[50] L. Wüstrich. (Accessed 2024-01-22) Shells Bells: Implementation. [Online]. Available: https://github.com/wuestrich/Shells-Bells

[51] L. Wüstrich, S. Gallenmüller, M.-O. Pahl, and G. Carle, "AC/DCIM: Acoustic Channels for Data Center Infrastructure Monitoring," in *NOMS 2022–2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–5.

[52] A. Yang, X. Wang, Y. Sun, Y. Hu, Z. Shi, and L. Sun, "Multi-Dimensional Data Fusion Intrusion Detection for Stealthy Attacks on Industrial Control Systems," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.