WaveSleuth: Retrospective PLC Memory for Anomaly Detection in Industrial Control Systems

Nehal Ameen*, Ramyapandian Vijayakanthan†, Adeen Ayub*, Aisha Ali-Gombe†, Irfan Ahmed*
* Department of Computer Science, Virginia Commonwealth University, USA
† Department of Computer & Information Sciences, Towson University, USA
‡ Computer Science and Engineering, Louisiana State University, USA
*{ameennm, ayuba2, iahmed3}@vcu.edu, †{rvijay1@students.towson.edu} ‡{aaligombe@lsu.edu}

***Abstract*—This paper presents WaveSleuth for industrial control systems (ICS), a novel intrusion detection system (IDS) that leverages the volatile memory of controller devices, transforming it into audio signals to detect cyberattacks. Attackers target programmable logic controllers (PLCs) in ICS to sabotage underlying physical processes; their attack footprints are often present in PLC memory, which WaveSleuth utilizes to detect a compromised PLC. Since PLCs are proprietary and heterogeneous, analyzing their device memory at a semantic level is challenging and does not scale. WaveSleuth models a PLC device memory as lossless audio signals at the byte level and deploys audio fingerprinting to detect anomalies with high confidence without requiring semantic knowledge of the data structures and firmware in the memory. Specifically, it periodically extracts volatile memory from a PLC device, converts the acquired raw memory data into audio signals, and then extracts a set of features called Mel-frequency cepstral coefficients (MFCCs) to measure the dynamic time warping (DTW) distance between two consecutive memory snapshots. WaveSleuth uses a distance threshold to determine whether the most recently acquired memory snapshot has been altered unexpectedly. We evaluate WaveSleuth on four real-world control logic and device firmware attacks on an actual PLC used in industrial settings to control a laboratory-scale, fully functional four-floor elevator. We also benchmark WaveSleuth with a state-of-the-art solution, PEM, which requires semantic knowledge to analyze PLC memory contents. Our evaluation results show that the attacks modify different PLC memory regions, such as firmware jump table and I/O data; WaveSleuth outperforms PEM and detects all attacks successfully.**

***Index Terms*—Cyber-physical systems, industrial control systems, programmable logic controllers**

I. Introduction

Industrial control systems (ICS) are integral to various critical infrastructures, automating industrial physical processes such as nuclear and power generation plants, oil and gas pipelines, and transportation systems [1], [2]. Physical processes are located at field sites and are directly controlled by programmable logic controllers (PLCs). The control center runs ICS services such as HMI, historian, and engineering workstations that receive and utilize data on physical processes from PLCs for remote monitoring and maintenance. PLCs are embedded devices equipped with control logic programs that define how a physical process is controlled [3].

Attackers typically target a PLC's control logic to sabotage a physical process, using different techniques such as fragmentation and noise padding, return-oriented programming, data execution, and redundant address pins (RAP) attacks, etc [4]–[14]. For example, RAP [10] is used to stealthily attach a small piece of programmable code (PMC) to a PLC control logic; PMC then starts running as part of a control logic scan cycle and is used to obfuscate and transfer control logic code from an attacker machine to a target PLC. These network-based stealthy attacks leave their footprints in a compromised PLC memory, which can be leveraged for detection. The challenge is to model and analyze the PLC memory contents. Limited work in PLC memory analysis mostly requires semantic knowledge of device memory specific to a PLC make and model, making it challenging to scale to heterogeneous proprietary devices. [15]–[22].

In this paper, we present WaveSleuth, a novel intrusion detection system for PLCs that models device volatile memory as audio signals and further employs audio fingerprinting to detect anomalies and raise alerts. By converting memory contents into audio signals, WaveSleuth takes advantage of audio signal analysis, a well-established field of study. WaveSleuth works as follows: It periodically acquires memory contents from a target PLC, transforms them into audio signals at the byte level without requiring semantic knowledge of PLC memory contents, and then extracts features from audio signals since there are many distinctive extractable features; WaveSleuth uses the Mel-frequency cepstrum coefficient (MFCC) [23]. MFCC is one of the most widely used audio signal features in various fields such as acoustic, medical, and industrial analysis to detect changes [24]. Given that MFCCs encapsulate the fundamental structure of the spectral envelope, they serve as an effective means to capture distinctive patterns and inherent attributes within signals, enabling the extraction of integral insights into both the current state and the operational intricacies of the PLC memory. Once WaveSleuth extracts the MFCCs from the audio signals of the memory snapshots, it measures the similarity between the signals by calculating the dynamic time warping (DTW) distance between consecutive memory snapshots [25]. This meticulous analysis determines the integrity of the latest memory snapshot, effectively discerning whether any unauthorized tampering has occurred.

We demonstrate that WaveSleuth works well in practice by subjecting it to a comprehensive evaluation using a fully operational physical process designed to emulate a four-floor elevator system. Throughout this assessment, we execute four distinct attacks tailored to target five different regions

within the volatile memory of the PLC, inducing alterations therein. The attack footprints vary in size, starting at 4 bytes, which is the size of an address pointer, and goes up to 660 bytes, as in the case of altering the entire control logic. We also benchmark closely related work, PEM [17]. Our results indicate that, unlike PEM, WaveSleuth can accurately identify all these attacks on PLCs, even those with minimal footprints. Additionally, we perform rigorous stress tests on memory modifications to illustrate WaveSleuth's detection capability. Our evaluation results show that WaveSleuth is proficient in swiftly discerning unusual deviations between two consecutive memory snapshots.

The contributions of this work are summarized as follows:

1) **Device-Agnostic Anomaly Detection:** Traditional anomaly detection in PLCs often requires deep knowledge of specific device configurations. This work presents a novel, device-agnostic methodology for detecting unauthorized modifications in PLC volatile memory. It leverages zero-semantic knowledge, allowing WaveSleuth to readily deploy across various PLC models without requiring model-specific customization.
2) **Enhanced Detection of Low-Footprint Attacks:** WaveSleuth can uncover evidence of sophisticated attacks that attempt to leave minimal footprints. This high-fidelity attack detection capability significantly improves the IDS's ability to identify and respond to the most subtle intrusion attempts.
3) **Rigorous Stress Testing for Robustness:** To ensure the reliability and effectiveness of WaveSleuth in real-world deployments, we have subjected it to a comprehensive stress testing methodology to manipulate a PLC memory at different scales and detect changes. This rigorous testing process guarantees robustness under various attack footprints in a PLC memory, fostering trust in its ability to perform effectively in practice.

II. Motivation and Problem Statement

Motivation. Critical infrastructure such as power generation, water treatment, and transportation are constantly targeted by cyber-attacks all over the world. In December 2015, the world witnessed the first known power outage caused by a malicious cyber-attack. Three utility companies in Ukraine were hit by BlackEnergy malware, leaving hundreds of thousands of homes without electricity for six hours [26]. The German Federal Office for Information Security reported in 2014, that a German steel mill suffered major damage after a cyber-attack forced the shutdown of a furnace, where the attackers used social engineering techniques to gain control of the blast furnace systems [27].

The legacy of the Colonial Pipeline incident is often called into question over a straightforward distinction: Russian ransomware hackers didn't shut down the pipeline themselves; Colonial did [28]–[32]. They did it as a precautionary measure because they were unsure about the extent of the attack and whether it could spread to their physical and industrial control systems, causing even more damage. This raises the question of whether there is a way to identify the size and impact of an attack without shutting everything down.

Problem Statement. Given an ICS environment, our goal is to identify the compromised PLC devices at any given time. In case of a cyberattack, this will allow asset owners/operators to estimate the scale of the attack by determining the total number of compromised devices, the impacted region (subnets) of ICS, and the exact devices that are compromised and need immediate attention for recovery.

III. Attack Model

Attacker Capabilities. The attacker gains access to the ICS control center through common methods such as social engineering (e.g., phishing emails) or introducing malicious USB drives. This reflects real-world attacks like TRITON [33] and the Ukraine Power Grid attack [34]. The attacker can also target supply chain vulnerabilities to access critical components or software within the ICS environment. Additionally, they can exploit weaknesses in remote access protocols for field devices to establish a foothold within the ICS.

Attacker Knowledge Acquisition. The attacker acquires a replica of the PLC make and model before launching the attack on a real system. This allows for developing and testing exploits tailored to the specific PLC's firmware and communication protocols. If detailed specifications for the ICS protocols are not publicly available, the attacker may employ tools such as PREE [35] to reverse-engineer them. This enables crafting exploits that target vulnerabilities within communication mechanisms. The attacker also knows the memory map of the target PLC to identify code and data sections. This information may be obtained from datasheets/manuals or through techniques such as JTAG-based memory acquisition [36].

Attacker Goals and Techniques. The primary objective of the adversary is to launch a covert attack on ICS physical processes. Stealth is prioritized to evade detection by network-based intrusion detection systems (IDS) monitoring the ICS network traffic. Consequently, the attacker develops exploits for the targeted PLC to compromise control logic. Additionally, the attacker may attempt to disable or bypass the IDS through various means. This might involve data flooding to overwhelm the IDS, exploiting vulnerabilities within the IDS itself, or manipulating sensor data to mask malicious activity.

IV. WaveSleuth System Design

This research proposes an integrated IDS and triaging framework titled *WaveSleuth* that leverages retrospective PLC volatile memory analysis using audio fingerprinting. As shown in Figure 1, WaveSleuth comprises four main modules.

- ***Memory Acquisition:*** This module acquires a snapshot of an active PLC memory, providing critical data for intrusion detection and triaging in an ICS network.
- ***Pre-processing and Feature Extraction:*** This module pre-processes the raw memory (from binary form) into audio signals for efficient and effective processing and then extracts the most distinct and deterministic features

from the audio signals to be used as an input for the anomaly detection algorithm.

- ***Similarity Measuring:*** This is the anomaly detection module that compares features from two audio signals (representing two memory instances) to determine if changes in the PLC runtime state have occurred.
- ***Memory Signal Detection:*** This final module is critical in determining whether a newly acquired memory snapshot is compromised, compared to a historic memory snapshot. This indicates a potential malicious activity or abnormal functionality of the target PLC.

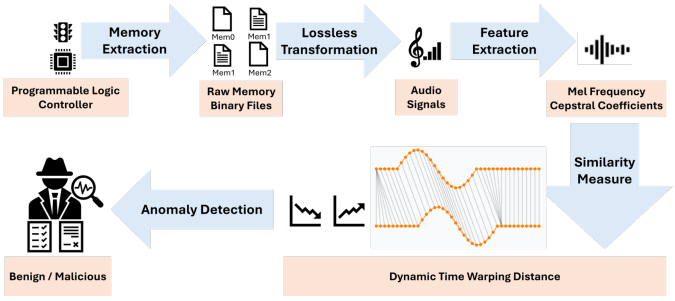


Fig. 1: WaveSleuth Framework

A. Memory Acquisition

A PLC's runtime memory provides a comprehensive snapshot of the system's current state, including control logic, I/O data, firmware data structures, and controller configurations. Leveraging a snapshot of a PLC memory can help detect indicators of compromise (IoCs), such as malicious control logic code, unauthorized firmware patching, and abnormal control behaviors that might not leave traces on a PLC device's volatile memory. In our proposed integrated framework, this real-time memory acquisition enables immediate identification and response to threats, facilitating rapid triage and minimizing the impact of an attack. We focus on runtime context, specifically memory snapshots for anomaly detection rather than network traffic analysis, primarily because:

- Unlike mobile devices or traditional computers, PLCs are task-specific. This uniqueness in functionality means that the same code set will execute continually over time except where an update is made. Thus, it is safe to assume PLCs maintain specific memory patterns over time.
- Network intrusion detection techniques are increasingly insufficient for detecting control and data-oriented attacks and other attacks designed to circumvent network-based IDSs such as CrashOverride and TRISIS [6], [37], [38].
- Modern attacks can leave no evidence on non-volatile storage, making it challenging for non-runtime-based IDSs or triaging tools to detect incidents [39].

From related literature, substantial research has addressed two main challenges in PLC memory acquisition. The first challenge is acquiring the entire PLC memory for comprehensive analysis, which cannot be achieved using only network protocol-based approaches [16], [40], [41]. The second challenge is finding a non-intrusive way to remotely acquire the entire PLC memory instead of using hardware-level debugging ports such as JTAG [36], which require physical proximity for disassembling a suspect PLC or power cycling. To address these challenges, PEM is a state-of-the-art memory acquisition framework to remotely acquire a PLC memory while it controls a physical process [17]. Our memory extractor module can leverage PEM or other PLC memory acquisition methods to enhance memory acquisition capabilities.

A PLC memory typically consists of protocol-mapped, i.e., readable over the network, and non-protocol-mapped memory spaces. Accessing the non-protocol-mapped memory can be challenging as it cannot be acquired directly via ICS protocols. To address this issue, PEM enables access to the non-protocol-mapped memory by appending a harmless duplicator code to the control logic [17]. This code copies the non-protocol-mapped memory space to the protocol-mapped memory space, which can then be accessed using the ICS network protocol. The original control logic of the PLC runs before the appended duplicator; therefore, the PLC can still control its underlying physical process during memory acquisition. Leveraging PEM for the integrated framework, the entire memory of a PLC, including regions such as external RAM, peripheral I/O, and on-chip RAM (in-memory firmware), can be acquired through an ICS network protocol. WaveSleuth incorporates PEM to periodically extract memory snapshots from PLCs to compare their new and historical images.

B. Preprocessing and Feature Extraction

This module consists of two sub-modules to pre-process a memory dump and extract features afterwards.

1) ***Preprocessing****:* The extracted PLC memory images are in a binary unstructured format, making their analysis challenging, often requiring specialized tools and expert knowledge in memory forensics and reverse engineering. Therefore, a methodology that is easy to use and operates with minimal semantic knowledge is essential to streamline this complex and time-consuming analysis process. Thus, given raw binary data without any semantic information, preprocessing can be done in two ways: processing the binary data using techniques such as byte frequency or n-gram analysis or transforming the binary data into another representation such as an image or audio signal. Our proposed approach favored audio encoding over direct binary analysis because audio encoding can highlight subtle features and variations in the data that might be missed in traditional binary analysis, such as n-gram. Additionally, audio encoding allows for applying sophisticated signal processing techniques for analyzing complex data patterns, making it a robust method for detecting anomalies in the acquired memory snapshot in our integrated IDS and triage framework.

Therefore, we encode the raw memory bytes into sound wave signals in this module using a **lossless transformation** algorithm for efficient and effective processing. The goal of lossless data-to-signal encoding is to ensure that the original data can be accurately recovered from the encoded signal without any loss of information, meaning the decoded data

is identical to the original data. The memory snapshot of an active process is temporal, with different memory regions (configuration block, data block, code block, etc.) exhibiting varying frequencies.

We start our audio signal generation process by inputting the memory dump, in the form of a binary file, into WaveSleuth. WaveSleuth parses the binary file, creating a byte stream, which is then converted into a Waveform Audio File Format (WAV) at a sampling rate of 48,000 Hz. This approach allows us to leverage advanced audio processing techniques to analyze the memory data, facilitating the identification of patterns and anomalies that might be missed in traditional binary analysis. Sampling refers to the process of converting an analog signal into a discrete time signal containing a sequence of samples. We opted for this specific sampling rate based on the Nyquist-Shannon sampling theorem, which stipulates that our sampling frequency must be at least twice the maximum frequency present in the signal [42]. Given that frequency in this context pertains to the auditory qualities of loudness or pitch, its values are inherently variable. Since the human auditory spectrum spans approximately from 20 Hz to 20,000 Hz, adherence to the theorem mandates a sampling rate surpassing 40,000 Hz to reproduce the sound wave signal in an audio format. As our goal is to extract the MFCCs from each sound wave signal, we use the minimum sampling rate of 48,000 Hz.

Subsequently, the values within the byte stream are interpreted as a sequence of amplitude values corresponding to the sound wave. This interpretation entails mapping the byte stream values to the potential amplitudes inherent in the waveform. Typically, these amplitudes range from 0 to 255 or are expressed bipolarly as -1 to 1. This transformation yields amplitude values accurately depicting the waveform's intensity at various points along its timeline.

This transformation process is iteratively applied to all memory instances, ensuring that each instance is converted into its corresponding sound wave representation. These representations are the foundation for our subsequent analyses and evaluations. To verify that our transformation algorithm is fully lossless, we plot a 100-byte sample of a memory dump onto a graph, showing the decimal value of each byte between 0 and 255. We then convert the memory sample into its audio waveform, with the amplitude value of each byte ranging from 0 to 255. Figure 2 demonstrates the transformation is 100% lossless, with the original binary sequence retaining identical values and patterns to the audio waveform..

*2) **Feature Extraction**:* In signal processing, mathematical techniques are often applied to extract specific characteristics unique to each signal. These deterministic attributes, also called domain features, are consistent across all similar signals and divergent in dissimilar signals; therefore, they can be used to identify distinguishable unique patterns, trends, or anomalies. Various domain features exist in audio signals, such as time, frequency, cepstral, and discrete wavelet transform domain features [43]. One commonly used audio analysis feature for effective audio analysis is MFCCs, a matrix of cepstral

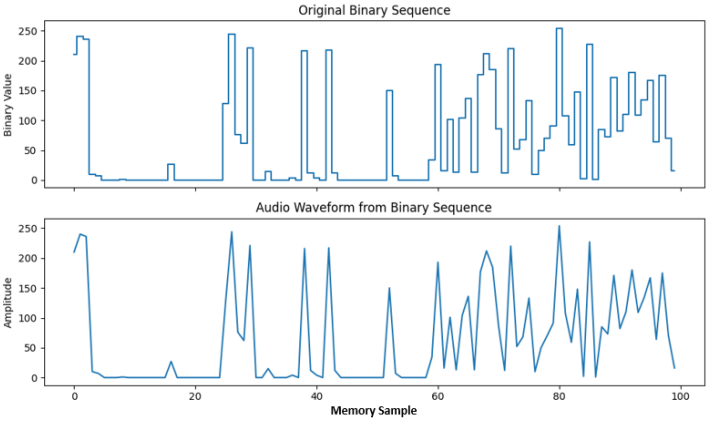


Fig. 2: Lossless Transformation from Binary data to Audio Waveform data

domain features. MFCCs are the coefficients that make up the mel-frequency spectrum. Cepstral features are frequency-smoothed representations of the log magnitude spectrum and capture timbral characteristics and pitch. The information on the rate of change in spectral bands of a signal is given by its cepstrum [44]. For the integrated framework, WaveSleuth leverages the MFCCs extracted from the encoded audio signals as the data points for anomaly detection measures.

Traditionally, sound wave signals are transitioned from the time domain to the frequency domain by applying the Discrete Fourier Transform (DFT). However, an inherent limitation of this approach lies in its propensity to yield a representation that reflects the averaged frequency components present throughout the signal's entire duration. In essence, it provides an overview of which frequency components exist within the system but fails to provide precise temporal details regarding when these frequencies rise or fall. To circumvent this limitation, we leverage the capabilities of the Librosa audio analysis package for extracting MFCCs from each individual sound wave signal. Librosa employs a Short-Time Fourier transform (STFT) in spectrogram operations [45]. This approach enables us to obtain a more granular and informative representation of the sound wave's frequency content over time, which is crucial for the integrated IDS and triaging process.

The structure of the MFCC features extracted from a sound wave signal takes the form of a matrix composed of nested arrays, each containing numerical values. These values encode the rate of change across distinct spectral bands within the signal, providing a comprehensive representation of the spectral characteristics of the signal, particularly its tonal content and distribution across the frequency spectrum. Specifically, when applied to the original memory snapshot, the MFCCs represent the rate of change of bytes across different memory regions. This memory-encoded audio signal representation identifies patterns and anomalies within the memory, highlighting areas of unusual activity or potential compromise. By transforming memory data into this new feature space, we can employ binary diffing techniques on the MFCCs matrix values to detect intrusion and cyber incidents more effectively.

The following summarizes the steps involved in the preprocessing of a PLC raw memory data until it is transformed into

a sound wave signal and the mathematical functions [46] that extract the MFCCs from the audio signal:

1) **Raw Memory to Sound Wave Signal:**
   - Acquire a raw memory snapshot, typically named $\texttt{Mem}_n\texttt{.bin}$.
   - Convert the binary file into a byte array object for easier manipulation and processing.
   - Further convert the byte array into a Numpy array object to facilitate mathematical operations.
   - Encode the Numpy array as a sound wave signal, transforming the memory data into an audio format.
2) **Fast Fourier Transform (FFT):** Apply the Fast Fourier Transform to the sound wave signal to convert it from the time domain to the frequency domain, revealing the signal's frequency components.
3) **Log-Amplitude Spectrum:** Convert the amplitude spectrum obtained from the FFT into a log amplitude spectrum to compress the dynamic range.
4) **Mel Scaling:** Map the frequencies to the Mel scale using:
$$\text{Mel}(f) = 2595 \times \log(1 + \frac{f}{700})$$
This step adjusts the frequency representation to apply the Discrete Cosine Transform in the next step.
5) **Discrete Cosine Transform (DCT):** Apply the Discrete Cosine Transform to the log amplitude Mel-scaled spectrum to decorrelate the features and compact the information into the first few coefficients.
$$C(x(t)) = F^{-1}(\log(F[x(t)]))$$
6) **Mel Frequency Cepstral Coefficients (MFCCs):** Extract the resulting coefficients from the DCT, known as Mel Frequency Cepstral Coefficients (MFCCs), which effectively capture the essential features of the signal. Compute the MFCCs using the `librosa` library in Python:
$$\text{mfcc1} = \text{librosa.feature.mfcc}(y1, sr1)$$

C. Similarity Measure

Following the extraction of MFCCs, the next step involves evaluating if two MFCCs are similar, i.e. if they are extracted from the same memory-encoded audio signal. In traditional similarity-matching algorithms, various distance measures such as Euclidean distance, Cosine similarity, Autocorrelation, and others are commonly employed to assess the similarity between feature vectors, such as MFCCs. These similarity measurement formulas differ according to the types of objects being compared. For example, cosine similarity measures the similarity between two non-zero vectors [47], Euclidean and Manhattan distance measure the similarity between two data points [48], Hamming distance measures the similarity between two strings [49]. However, there's a crucial consideration here: These algorithms are typically optimized for scenarios where input sequences are synchronized at both time and speed. Given that the extracted MFCC features exhibit variations in both time and speed, relying solely on the Euclidean distance or similar metrics may not yield accurate similarity scores. Moreover, the unique interplay of spectral energies, which can be concentrated or sporadic over time, further introduces temporal differences into the traces. These spectral energies correspond to different memory segments or regions, adding complexity to the analysis. Due to this inherent asynchronicity in the resulting sequences, it becomes crucial to leverage techniques such as DTW [50] for quantifying similarity between them.

DTW is a technique used to measure the similarity between temporal or two-time series sequences that may have differences in timing, speed, and length, i.e. not fully aligned [25], [50], [51]. It is often used in speech and audio recognition, gesture recognition, and pattern recognition and works by comparing two-time series sequences and warping one of the sequences to align with the other sequence to obtain an optimal alignment. The warping process adjusts the timing and scaling of the sequence to match the other sequence as closely as possible. The DTW algorithm first creates a matrix representing the distance between each point in the two sequences. The algorithm then finds the optimal path through this matrix, which corresponds to the minimum distance between the two sequences. The optimal path is determined by minimizing the total cost of all the cells in the matrix along a path that moves diagonally, horizontally, or vertically. DTW has several advantages over other techniques used for comparing time series data. One of the main advantages is that it can handle sequences of different lengths and variable time scales making it useful for analyzing memory-encoded signals of different sizes. It is also able to align sequences with non-linear warping functions, making it useful for analyzing complex patterns and signals. Additionally, DTW accommodates variations in the temporal alignment of sequences, which makes it particularly well-suited for comparing MFCC features that exhibit time and speed discrepancies, enabling more accurate and reliable similarity assessments in the context of memory analysis. Therefore, WaveSleuth calculates the DTW distance between the two memory instances. It is important to note that the DTW distances, when applied in the context of memory-based anomaly detection, provide insight into the variability of the system's behavior over time. Smaller distances suggest the behavior is relatively consistent, while larger distances suggest more variability.

D. Anomaly Detection

Once we've extracted the MFCCs and calculated the DTW distances between two memory images, a baseline memory image, and a DTW threshold must be determined to ascertain if a target device's runtime activity has deviated from its normal pattern of activity. This critical step is essential for detecting intrusions and forensics triaging after a cyber incident.

Generally, there are many techniques for determining the threshold for an anomaly detection algorithm, such as selecting the Max, Min, or Median of all known benign distances. Other techniques, such as clustering, distance to nearest neighbors,

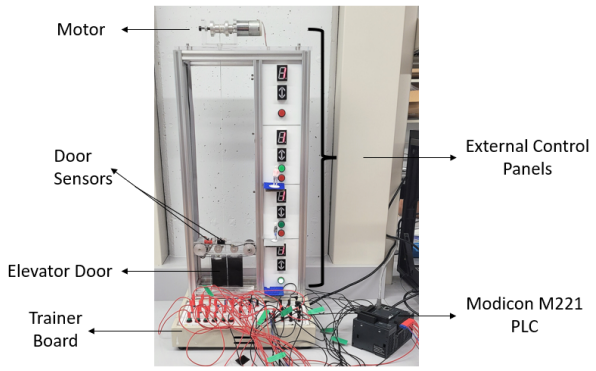


Fig. 3: Fully-functional Four-floor Elevator Testbed

etc, have also existed in the literature [52]–[54]. For our purposes, we used the most conservative approach: the Max of all known DTW distances between benign images. Although this approach may result in a threshold that is too high, it is guaranteed to ensure that all observed variations in normal behavior are accounted for, thereby reducing the risk of false positives. This trade-off is particularly crucial for physical processes with many states, which can result in variable DTW distances. Additionally, considering our features are based on runtime characteristics (memory), most known PLC attacks are likely to exceed the normal activity threshold. Therefore, this conservative approach effectively balances the need for thorough anomaly detection while minimizing the likelihood of flagging normal behavior as anomalous.

Thus; **to find the optimal threshold (x):**

$x = MAX(d_{t0}, d_{t1}, ..., d_{tn})$

To detect anomaly:

$If(d_{tn} <= x) \rightarrow mem_n$ is benign

$else \rightarrow mem_n$ is anomalous

Where d is the DTW between mem_n and baseline image mem_b

V. Experimental Setup

This section provides a brief overview of the experimental setup used to evaluate the effectiveness of WaveSleuth.

As shown in Figure 3, we designed a testbed using a Modicon TM221CE16R version 1.6 PLC connected to an elevator model. This testbed implements control logic for an elevator that travels between floors 1, 2, and 3, with each action representing a separate time instance. We used EcoStruxure Machine Expert - Basic software, a programming tool for M221 controllers, to build the control logic. Additionally, we utilized an older version of the PLC (Modicon TM221CE16R version 1.5) to exploit the vulnerabilities patched in the newer version for our evaluation.

In the context of the elevator system, we developed a testing framework that comprises three distinct states of operation to simulate various scenarios and potential real-world conditions. *State 1 (Power Cycling):* In this initial state, we cycle the power of the PLC by turning it off and then on between different tests. This scenario mimics scenarios where the PLC undergoes complete power interruptions and restarts.

State 2 (Controller Restart): In the second state, we maintained continuous power to the PLC but introduced controlled disruptions. Specifically, we halted the controller's operation and subsequently resumed it remotely from the engineering software. This state's setup is designed to emulate situations where the controller experiences controlled stoppages and restarts while the system remains powered.

State 3 (Continuous Operation): In the third state, we maintained continuous, uninterrupted operation of the PLC without any deliberate interventions or disruptions. This scenario represented a stable, ongoing system operation. Each state comprises a series of specific scenarios, each characterized by a unique time instance denoted as (t_n), representing the action and position of the elevator at the time of memory acquisition.

VI. Evaluation

To evaluate the effectiveness of WaveSleuth, we developed three sets of experiments that map three important hypotheses and the core contribution of this research. These experiments are:

1) **E1:** The **effectiveness** of the proposed system in detecting and/or triaging real attacks on PLCs, even those with small footprints.
2) **E2:** The **robustness** and efficacy of leveraging DTW compared to other distance measures, such as hamming distance, in determining changes to MFCCs.
3) **E3:** The **resiliency** of the proposed device-agnostic methodology, which utilizes features from memory-encoded sound signals to detect alterations.

A. (E1) WaveSleuth effectiveness to detect PLC attacks with varying memory footprints

The attacks on PLCs may have small memory footprints that raise questions about the effectiveness of WaveSleuth's threshold approach. This section evaluates WaveSleuth in detecting real PLC attacks with different memory footprints on a compromised PLC device. It further compares WaveSleuth's performance in detecting these attacks with PEM, a state-of-the-art related work [17].

1) Establishing Baseline and Threshold: Wavesleuth uses the following five steps to establish a threshold:

- **Initial Acquisition**: Acquire the external RAM image at a designated time instance t_n.
- **Baseline Setting**: Set this image as a baseline mem_b .
- **Subsequent Capture and Comparison**: Repeat the initial acquisition at t_{n+1}, then compare the new image with the baseline image using DTW distance.
- **Iteration**: Alternate between normal operation states, repeating steps 1 to 3 until a specified number of iterations m is achieved. For our experiments, we acquired 51 benign memory images in the three states, each constituting different actions. Each memory is compared with the preceding one, and a DTW distance is established.

- **Threshold Setting and Deployment**: The detection threshold for the PLC is set based on the max of all the DTWs before deploying the PLC.

Figure 4a shows what an unmodified (benign) PLC memory pattern looks like in State 1, where we systematically cycle the power of the PLC between each instance. Figure 4b demonstrates a benign pattern in State 2, where we maintain continuous power to the PLC but introduce controlled disruptions by stopping and restarting the controller remotely from the engineering software. Figure 4c shows a benign pattern in State 3, where the PLC is in continuous operation without any disruptions to simulate a stable, ongoing PLC operation.

It is important to note that a new threshold must be established for each PLC and control logic whenever an update or configuration change occurs to prevent any unusual but normal activity from being flagged as a malicious activity, resulting in false positives. This is done by repeating the 5-step installation process explained above. As noted in this process, the last step involves establishing a threshold. As discussed in Section IV, we chose to use the maximum DTW distance as the threshold. In our current example, the highest DTW distance value is 1272.00. Consequently, whenever we acquire a new memory snapshot (mem_n), we measure the DTW distance (d) between the new memory snapshot and the baseline (historical) memory snapshot.

With the threshold established, to detect an anomaly in the PLC, for each new memory image mem_n, compute its DTW distance (d) to the last baseline mem_b. Compare the DTW distance against the threshold (x). If the distance exceeds the threshold, flag it as malicious and alert the HMI; otherwise, set this new memory image mem_n as the new baseline mem_b and wait for the next image.

2) Detecting Real PLC Attacks using the Threshold: Using our designed testbed, we executed four different real-world attack scenarios targeting various memory regions. These attacks simulate situations where malicious control logic is executed to emulate potential security breaches or unauthorized intrusions. Specifically, these attacks target PLC control logic, firmware, and user authentication. After launching each attack, we acquired the memory images generated by these malicious actions. These new images now contain the altered or compromised states of the elevator system post-attack. For each one of the four attack scenarios, we discuss a graph showing patterns that the memory exhibited at different time instances both for pre- and post-attack instances (t_0 to t_n), illustrating the pre-attack threshold.

Additionally, in Table I we summarize all four attack footprints, including the regions of memory modified by the attack and the scale of the modification.

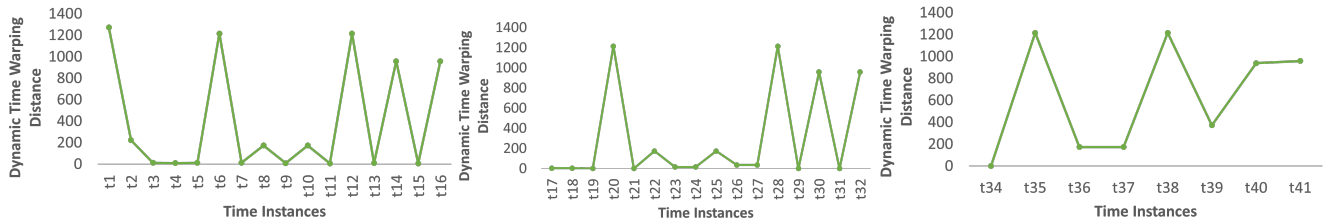
Data Execution, Fragmentation, and Noise Padding (DEF&NP) Attack. We evaluated Yoo *et al.*'s control logic injection attacks, focusing on data execution, fragmentation, and noise padding techniques [5]. In this scenario, the attacker aims to disrupt physical processes by injecting malicious control logic into a PLC. In the data execution attack, PLCs lacking data execution prevention (DEP) measures enable attackers to manipulate data blocks, executing any memory block for malicious activities. The attacker gradually transfers the malicious control logic to the PLC's data blocks through periodic I/O data reading by the HMI, then redirects the control flow to execute the injected logic using pointers, such as the one in an M221 PLC's configuration block. The fragmentation and noise padding attack capitalizes on the limitations of techniques such as Deep Packet Inspection (DPI), evading detection with small-sized payload packets containing fragmented control logic and added noise. Yoo et al. combine these attacks effectively, dividing control logic into 'N' fragments and sending each fragment in a different packet with appended noise. Each subsequent write request overwrites a byte of noise from the previous stage, ensuring stealthy consecutive writing of the control logic byte by byte in the data block until completion. This results in the elevator skipping the second floor while keeping the door open as it travels between different floors. The first entry in Table I illustrates the DE&NP attack footprint in the memory, where the configuration block contains a 4-byte pointer altered to point to the data block, which holds 660 bytes of injected control logic.

TABLE I: Memory Footprints for the Four PLC Attacks

Attack	Affected Region	Change Size in Bytes
DEF&NP	Configuration Block	4 bytes (Pointer)
	Data Block	660 bytes (Control Logic)
Control Logic Injection	Code Block	660 bytes (Control Logic)
Direct Firmware Object Manipulation	Code Block	11 bytes appended to Control Logic
	Free Space (in Ext RAM)	21 bytes
Password Reset	Ext RAM, right after ZIP	2000 bytes

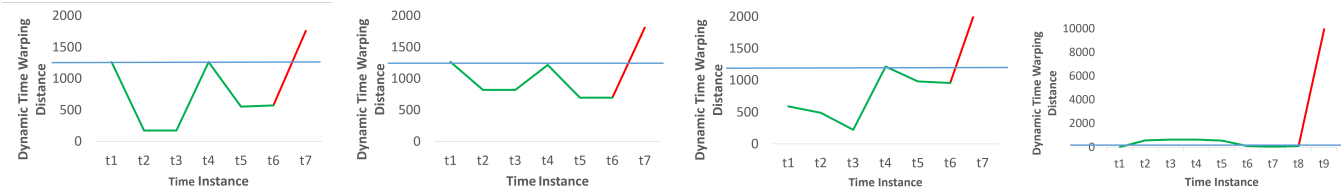
Figure 5a compares six unmodified (benign) memory snapshots taken pre-attack and one malicious memory snapshot taken post-attack. We started our experiment by downloading our benign control logic from our PC to the PLC and resetting the PLC. While the elevator remained at its resting state on floor 1, we acquired our first unmodified memory snapshot (t_0). Then, we called the elevator to the second floor and acquired our second memory snapshot (t_1). To observe how normal operation changes the PLC memory, we calculated the normal DTW distance between two consecutive normal states of the elevator. We processed the memory snapshots through WaveSleuth, converting them into two different audio signals and measuring their similarity, resulting in a DTW distance of 1263.14.

We then moved the elevator back and forth between different floors to emulate a real-life scenario, acquiring a new memory snapshot after each action and comparing it to the previous snapshot. Following the same procedure, we calculated the DTW distance between each pair of consecutive memory snapshots (t_n) and (t_{n-1}), yielding different DTW distances. The green line between t_1 and t_6 in Figure 5a demonstrates these benign changes and their effect on the PLC memory, as illustrated by the DTW distances between each consecutive memory images.



(a) Power Cycling (b) PLC Restarting (c) PLC Operational

Fig. 4: Benign DTW distance changes over time between power cycles, controller restarts, or continuous operation.



(a) DEF&NP (b) Control Logic Attack (c) DFOM Attack (d) Password Reset Attack

Fig. 5: Benign Memory (Green) followed by an attack (Red). The horizontal line represents the threshold.

Next, we introduced the DEF&NP attack and measured the DTW distance between our most recent benign memory snapshot (baseline) (t_6) and the new malicious memory snapshot (t_7). The DTW distance between (t_6) and (t_7), represented by the red line, jumps to 1758.95, demonstrating a much larger difference between benign and malicious memory snapshots compared to the smaller differences between consecutive benign snapshots. Thus, this larger DTW compared to our threshold of 1272.00, set up in the testbed design for this PLC, shows that our proposed methodology can effectively detect a DEF&NP attack even with changes of only about 664 bytes (less than 1KB).

Control Logic Injection Attack. In this scenario, the attacker aims to gain unauthorized access to the engineering software that interacts with the PLC, in this case, the SoMachine Basic. This can be achieved by social engineering, where the attacker targets personnel with access to SoMachine Basic. Phishing emails with malicious attachments or links can trick users into downloading malware that compromises their workstations. Once inside the network, the attacker can exploit legitimate access to SoMachine Basic to download the malicious control logic to the PLC. Another way to obtain access to SoMachine Basic is through a Watering Hole attack, where the attacker identifies websites frequented by personnel using SoMachine Basic. They compromise these websites to inject malicious code that exploits vulnerabilities in SoMachine Basic when users visit the site. This could be a drive-by download attack where the user unknowingly downloads malware that provides access to SoMachine Basic. We simulate this attack by downloading the malicious control logic in the previous attack to the PLC. We then acquire pre- and post-attack memory snapshots, as explained previously, and measure the DTW distances.

Figure 5b shows that the highest DTW distance observed during normal operation was 1262.14. Meanwhile, the DTW distance between a normal and a malicious memory snapshot increased to 1810.27, above the 1272.00 established threshold. The attack map in Table I shows the footprint of this attack in the code block, where 660 bytes of malicious control logic replaced the original control logic.

Direct Firmware Object Manipulation (DFOM) Attack. DFOM is a method of attacking a firmware data structure by targeting a specific firmware function. For example, in the case of our M221 PLC, a jump table containing the addresses of built-in firmware functions such as timers and counters is maintained to be used by the control logic. DFOM attacks the control logic by modifying the jump table to target a specific function. It achieves this by changing the jump table to redirect the timer function responsible for opening the elevator door after a 3-second delay to a malicious timer function that disables the delay, causing the door to open before the elevator reaches its destination. First, DFOM injects a malicious payload for the timer into the free space of the external RAM region of the PLC, and then it modifies the jump table with the address of the malicious timer code.

Figure 5c shows the same approach we followed in the previous attacks. Again, we start by resetting our PLC to remove any artifacts left from the previous attack. Our benign changes due to regular operation result in a maximum DTW distance of 1215.72. On the other hand, once we introduce the DFOM attack at (t_7), the DTW distance jumps up to 2415.55. Table I shows the footprints left in the memory, which are 11 bytes appended to the control logic in the code block to update the 'Call Handler' to redirect to the 21-byte malicious timer that was injected into the free space of the external RAM.

While DFOM attacks in PLCs, in general, are stealthy and persistent [55], [56] and often with a very minimal footprint (total of 32 bytes changes) as shown in Table I, nonetheless, our proposed approach was able to effectively detect these memory changes and flag this DTW distance as anomalous.

Password Reset Attack. Lastly, we set up an attack that

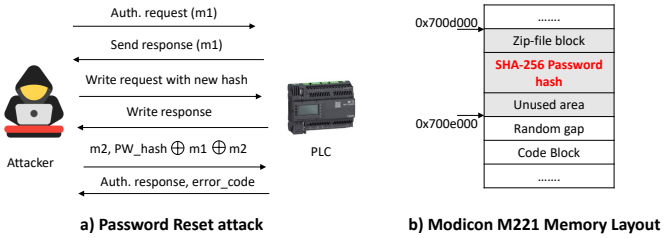


Fig. 6: Password reset attack in Modicon PLC

changes the external RAM in the region following the ZIP region. The ZIP region contains a compressed XML file with metadata about the control logic. PLCs rely on password-based authentication to protect control logic against unauthorized access. For this experiment, we utilized Kalle et al.'s [6] password reset attack on the M221 PLC. The authentication protocol comprises four messages, starting with an authentication request from the engineering software. The PLC responds with a one-byte key (m1). The next message includes a random one-byte value (m2) and an encoded SHA-256 password hash. The PLC then responds with an error code (success or failure).

Figure 6(a) illustrates the password reset attack used in our approach, while Figure 6(b) displays the PLC's memory layout. The password hash resides in the memory region `0xd000` to `0xe000`, but its exact location is variable due to varying ZIP file sizes.

The attack operates as follows: it initiates a request for m1 and receives a response from the PLC. Subsequently, it iteratively sends write requests with a SHA-256 password hash of the attacker's password, beginning from memory location `0xdfe0` (as the password hash is 32 bytes in size). After each write request, it follows up with an authentication request and repeats the process until authentication succeeds.

Figure 5d shows our evaluation of WaveSleuth's ability to detect this password authentication attack. For this evaluation, we ran our experiment on a Modicon TM221CE16R version 1.5 PLC that executed the same elevator control logic described earlier. The reason for using this specific version is that the password authentication vulnerability no longer exists in version 1.6. Because we are using a different PLC version, we need to establish a new baseline and threshold. We started by capturing 9 baseline normal memory snapshots (t_0 - t_8) to find the new threshold at 624.83. We then executed the password authentication attack and captured the malicious memory (t_9). WaveSleuth compared the latest two memory dumps (t_8) and (t_9) and found the DTW distance between them to be 9956.79, much higher than the established threshold. The reason for such a high DTW distance can be attributed to the size of this attack's footprint, approximately 2000 bytes, in Table I.

3) Benchmarking with Related Work: In this section, we compare Wavesleuth with PEM [17], a memory-based detection technique. Like WaveSleuth, PEM operates by comparing memory states before and after an attack. PEM analysis show that it is able to detect the DFOM attack effectively. However, it can not detect other attacks, as it only perceives them as memory changes without being able to tell whether they are malicious changes or not. Table II compares the detection effectiveness of Wavesleuth and PEM. As shown, Wavesleuth successfully detects all four attacks, while PEM is only able to identify one. This limitation in PEM arises from its reliance on semantic knowledge specific to each attack for detection.

TABLE II: Comparison with the related work PEM

Attack	WaveSleuth	PEM
DEF&NP	✓	✗
Control Logic Injection	✓	✗
Direct Firmware Object Manipulation	✓	✓
Password Reset	✓	✗

Summary Findings: From the discussion of the four attacks, our results indicate that WaveSleuth can detect (1) various different attacks, (2) attacks with very minimal footprints, and (3) stealthy and persistent attacks, irrespective of the region where the payload is placed, and (4) can perform better than a state-of-the-art solution, PEM. Collectively, considering our statistics for comparing benign (7 for DEF&NP, 7 for CL_INJECT, 7 for DFOM, and 9 for PSWD_RESET) and malicious (1 for DEF&NP, 1 for CL_INJECT, 1 for DFOM, and 1 for PSWD_RESET) images against the threshold, WaveSleuth accurately detects all the attacks, unlike PEM.

B. (E2) Robustness and efficacy of DTW compared to other distance measures, e.g., hamming distance, to determine changes to MFCCs.

As discussed in Section IV, DTW distance is not the only measure applicable for finding differences and similarities in MFCCs. Other distance measures, such as Cosine and Hamming distance, could also be applied. In this experiment, we compare the results of applying Hamming distance to those four attacks set in **E1** against DTW distance. The results, as shown in Figure 7, demonstrate that DTW performs well for all four attacks and is sensitive to even small normal operational changes. In contrast, Hamming distance is not sensitive to operational changes and failed to detect the memory changes of the DFOM attack. This clearly indicates that the choice of DTW improves the robustness and efficacy of WaveSleuth in detecting different types of PLC attacks while being resilient to normal operations.

Summary Findings: WaveSleuth DTW algorithm is robust in detecting different PLC attacks compared to other distance measures, such as Hamming distance.

C. (E3) WaveSleuth resiliency in utilizing features from memory-encoded sound signals to detect alterations.

In this last experiment, we conducted rigorous stress tests to further evaluate WaveSleuth's resiliency to memory alterations. To this end, we employ a bit-wise XOR manipulation technique using the mask 'FF' to examine the seven distinct regions within our baseline memory image. These changes were made directly to the acquired memory before the memory-to-audio transformation. The testing process is initiated by modifying the first 4 bytes of the memory dump. This alteration corresponds to the size of an address pointer

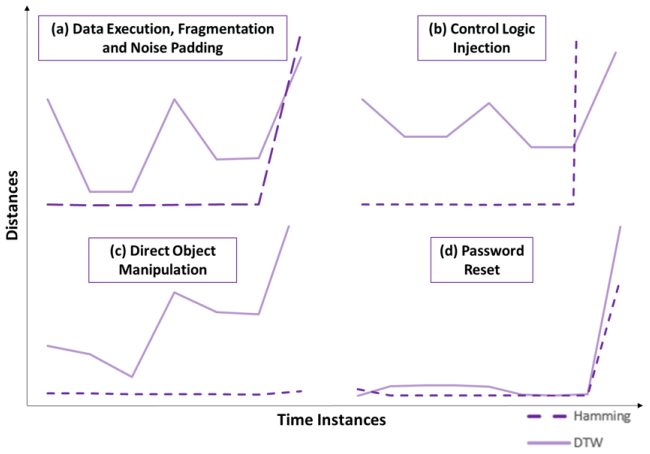


Fig. 7: Hamming (dotted line) vs. DTW (solid line) Distances between Memory Snapshots before and after Attacks

within a Modicon M221 PLC memory. For each region under examination, we continue the alteration by shifting our 4-byte alteration window sequentially until we traverse the entire extent of that region. Subsequently, we increase the alteration size by doubling the number of bytes, progressing from 4-byte alterations to 8, 16, 32, etc., and eventually reaching 512-byte alterations. Following this, we calculate the DTW distance between the baseline memory dump and each of the successively altered memory dumps.

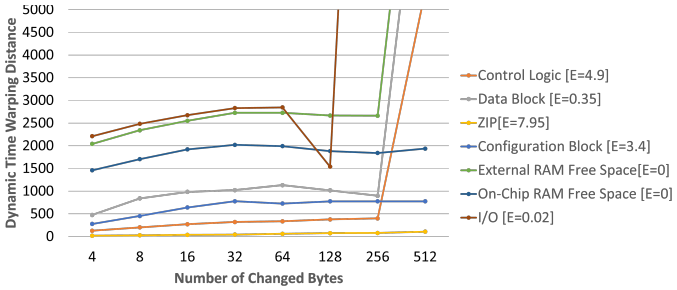


Fig. 8: The Average DTWs for the 7 Memory Regions after XOR alterations compared to the Baseline Memory Image

Figure 8 shows the effect of the alterations on the average DTWs of each of the 7 different memory regions compared to the baseline memory dumps (i.e., before alterations). We notice that the average DTWs calculated for the control logic, data, and configuration blocks that are located in the external RAM, as well the I/O region that is located outside of the external RAM, increase as the number of altered bytes increases from 4 bytes to 512 bytes. While the average DTW distances for almost all regions tend to increase with the increase in byte changes, certain regions, like the ZIP region with high entropy, remain unaffected by these alterations. Notably, the ZIP region achieved a high entropy score of 7.95 on a scale of 10. However, high-entropy regions need to be large to be unaffected by small alterations. Since they are simple to identify in memory, a separate threshold can be established for high-entropy regions to improve detection.

Additionally, in rare instances, our experiments showed a dip in the I/O region with more changes. The reason for that is that the high presence of '00' bytes in certain memory regions affects DTW distances when XOR-ed with 'FF'. This transformation causes original and XOR-ed bytes to mirror each other, reducing the distance between them. Regions with more '00' bytes exhibit lower DTW distances, reflecting their sensitivity to changes. This makes it difficult for attackers to alter specific memory values to execute effective attacks.

Summary Findings: WaveSleuth is resilient in detecting changes in memory regions. However, its performance may degrade in large regions with high entropy leading to false negatives, but is overcome by establishing separate thresholds for such regions.

VII. Related Work

This section presents related work on PLC memory analysis. Cook *et al.* introduce PLCPrint [18], a vendor-independent fingerprinting method that maps PLC registers to memory artifacts and detects memory attacks based on these mappings. Haris et al. focus on forensic analysis of Allen-Bradley ControlLogix PLCs, identifying key artifacts in memory dumps using specific string and data searches, and providing a Python library for this analysis [19].

Awad *et al.* perform memory forensics on the Schneider Electric Modicon M221 PLC by reverse engineering the communication protocol, creating a memory profile to assist in faster analysis during cyber incidents [15].

Yoo *et al.* develop "Shade," a shadow memory technique that observes network traffic to maintain a local copy of PLC memory and analyzes it with a classification algorithm [16]. Caselli et al. propose S-IDS, a sequence-aware intrusion detection system, to detect semantic attacks exploiting permitted operation sequences in ICS [57].

WaveSleuth can monitor and detect unusual PLC memory alterations periodically, complementing existing approaches focused on post-incident forensic analysis.

VIII. Conclusion

WaveSleuth is an integrated intrusion detection and triaging system designed to address cyber threats in PLC environments, which are increasingly vulnerable due to the evolving integration with modern IT infrastructures. The system works by converting complex binary data from PLC runtime activities into audio signals to improve the detection of compromised PLCs. Since WaveSleuth requires zero-semantic knowledge, it can work on heterogeneous PLC devices, enhancing its adaptability. The extensive evaluation and memory stress testing shows that WaveSleuth outperforms PEM, a state-of-the-art solution, and can accurately detect attack anomalies in PLC device memory. From a practical standpoint, WaveSleuth adds to defense in depth and complements network-based IDS to find compromised PLCs at the edge. Considering PLCs are resource-constrained devices, WaveSleuth should be used periodically or in case of intrusion alerts or network incidents to ensure the controllers are not compromised.

REFERENCES

[1] I. Ahmed, S. Obermeier, M. Naedele, and G. G. Richard III, "Scada systems: Challenges for forensic investigators," *Computer*, vol. 45, no. 12, pp. 44–51, 2012.

[2] P. Zhang, *Advanced industrial control technology*. William Andrew, 2010.

[3] M. A. Sehr, M. Lohstroh, M. Weber, I. Ugalde, M. Witte, J. Neidig, S. Hoeme, M. Niknami, and E. A. Lee, "Programmable logic controllers in the context of industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3523–3533, 2020.

[4] R. Sun, A. Mera, L. Lu, and D. Choffnes, "Sok: Attacks on industrial control logic and formal verification-based defenses," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 385–402.

[5] H. Yoo and I. Ahmed, "Control logic injection attacks on industrial control systems," in *ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings 34*. Springer, 2019, pp. 33–48.

[6] S. Kalle, N. Ameen, H. Yoo, and I. Ahmed, "Clik on plcs! attacking control logic with decompilation and virtual plc," in *Binary Analysis Research (BAR) Workshop, Network and Distributed System Security Symposium (NDSS)*, 2019, pp. 1–12.

[7] W. Alsabbagh and P. Langendörfer, "A flashback on control logic injection attacks against programmable logic controllers," *Automation*, vol. 3, no. 4, pp. 596–621, 2022.

[8] N. Govil, A. Agrawal, and N. O. Tippenhauer, "On ladder logic bombs in industrial control systems," in *Computer Security: ESORICS 2017 International Workshops, CyberICPS 2017 and SECPRE 2017, Oslo, Norway, September 14-15, 2017, Revised Selected Papers 3*. Springer, 2018, pp. 110–126.

[9] A. Ayub, N. Zubair, H. Yoo, W. Jo, and I. Ahmed, "Gadgets of gadgets in industrial control systems: Return oriented programming attacks on plcs," in *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2023, pp. 215–226.

[10] A. Ayub, W. Jo, and I. Ahmed, "Charlie, charlie, charlie on industrial control systems: Plc control logic attacks by design, not by chance," in *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2024.

[11] A. Ayub, H. Yoo, and I. Ahmed, "Empirical study of plc authentication protocols in industrial control systems," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 383–397.

[12] A. Ayub, W. Jo, S. A. Qasim, and I. Ahmed, "How are industrial control systems insecure by design? a deeper insight into real-world programmable logic controllers," *IEEE Security & Privacy*, vol. 21, no. 4, pp. 10–19, 2023.

[13] S. Qasim, A. Ayub, J. Johnson, and I. Ahmed, "Attacking the IEC-61131 Logic Engine in Programmable Logic Controllers in Industrial Control Systems," in *Critical Infrastructure Protection XV*, J. Staggs and S. Shenoi, Eds. Cham: Springer International Publishing, 2021.

[14] N. Zubair, A. Ayub, H. Yoo, and I. Ahmed, "Control logic obfuscation attack in industrial control systems," in *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2022, pp. 227–232.

[15] R. A. Awad, M. H. Rais, M. Rogers, I. Ahmed, and V. Paquit, "Towards generic memory forensic framework for programmable logic controllers," *Forensic Science International: Digital Investigation*, vol. 44, p. 301513, 2023.

[16] H. Yoo, S. Kalle, J. Smith, and I. Ahmed, "Overshadow plc to detect remote control-logic injection attacks," in *Detection of Intrusions and Malware, and Vulnerability Assessment: 16th International Conference, DIMVA 2019, Gothenburg, Sweden, June 19–20, 2019, Proceedings 16*. Springer, 2019, pp. 109–132.

[17] N. Zubair, A. Ayub, H. Yoo, and I. Ahmed, "Pem: Remote forensic acquisition of plc memory in industrial control systems," *Forensic Science International: Digital Investigation*, vol. 40, p. 301336, 2022.

[18] M. M. Cook, A. K. Marnerides, and D. Pezaros, "Plcprint: Fingerprinting memory attacks in programmable logic controllers," *IEEE Transactions on Information Forensics and Security*, 2023.

[19] M. H. Rais, R. A. Awad, J. Lopez Jr, and I. Ahmed, "Memory forensic analysis of a programmable logic controller in industrial control systems," *Forensic Science International: Digital Investigation*, vol. 40, p. 301339, 2022.

[20] S. A. Qasim, J. M. Smith, and I. Ahmed, "Control logic forensics framework using built-in decompiler of engineering software in industrial control systems," *Forensic Science International: Digital Investigation*, vol. 33, p. 301013, 2020.

[21] S. A. Qasim, J. Lopez, and I. Ahmed, "Automated reconstruction of control logic for programmable logic controller forensics," in *Information Security: 22nd International Conference, ISC 2019, New York City, NY, USA, September 16–18, 2019, Proceedings 22*. Springer, 2019, pp. 402–422.

[22] S. A. Qasim, M. Raza, and I. Ahmed, "vplc: A scalable plc testbed for iiot security research," 12 2023.

[23] P. Cano, E. Batle, T. Kalker, and J. Haitsma, "A review of algorithms for audio fingerprinting," in *2002 IEEE Workshop on Multimedia Signal Processing.* IEEE, 2002, pp. 169–173.

[24] Z. K. Abdul and A. K. Al-Talabani, "Mel frequency cepstral coefficient and its applications: A review," *IEEE Access*, 2022.

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

[26] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, and S. Sezer, "Threat analysis of blackenergy malware for synchrophasor based real-time control and monitoring in smart grid," in *4th International Symposium for ICS & SCADA Cyber Security Research 2016 4*, 2016, pp. 53–63.

[27] K. E. Hemsley, E. Fisher *et al.*, "History of industrial control system cyber incidents," Idaho National Lab.(INL), Idaho Falls, ID (United States), Tech. Rep., 2018.

[28] S. Simper, "Colonial pipeline attack: two year anniversary," 2023.

[29] J. Easterly and T. Fanning, "The attack on colonial pipeline: What we've learned & what we've done over the past two years," 2023.

[30] S. Sabin, "Colonial pipeline ransomware attack's unexpected legacy," 2023.

[31] C. Bing, "Exclusive: U.s. to give ransomware hacks similar priority as terrorism," 2021.

[32] R. Lemos, "Tsa official: Feds improved cybersecurity response post-colonial pipeline," 2023.

[33] B. Johnson, D. Caban, M. Krotofil, D. Scali, N. Brubaker, and C. Glyer, "Attackers deploy new ics attack framework "triton" and cause operational disruption to critical infrastructure," *Threat Research Blog*, vol. 14, p. 94, 2017.

[34] F. CISA, "Nsa,"understanding and mitigating russian state-sponsored cyber threats to us critical infrastructure," jt," *Cyber Secur. Advis*, 2022.

[35] S. A. Qasim, W. Jo, and I. Ahmed, "Pree: Heuristic builder for reverse engineering of network protocols in industrial control systems," *Forensic Science International: Digital Investigation*, vol. 45, p. 301565, 2023.

[36] M. H. Rais, R. A. Awad, J. Lopez Jr, and I. Ahmed, "Jtag-based plc memory acquisition framework for industrial control systems," *Forensic Science International: Digital Investigation*, vol. 37, p. 301196, 2021.

[37] M. Geiger, J. Bauer, M. Masuch, and J. Franke, "An analysis of black energy 3, crashoverride, and trisis, three malware approaches targeting operational technology systems," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2020, pp. 1537–1543.

[38] Y. Hu, Y. Sun, Y. Wang, and Z. Wang, "An enhanced multi-stage semantic attack against industrial control systems," *IEEE Access*, vol. 7, pp. 156 871–156 882, 2019.

[39] A. Case and G. G. Richard III, "Memory forensics: The path forward," *Digital investigation*, vol. 20, pp. 23–33, 2017.

[40] T. Wu and J. R. Nurse, "Exploring the use of plc debugging tools for digital forensic investigations on scada systems," *Journal of Digital Forensics, Security and Law*, vol. 10, no. 4, p. 7, 2015.

[41] G. Denton, F. Karpisek, F. Breitinger, and I. Baggili, "Leveraging the srtp protocol for over-the-network memory acquisition of a ge fanuc series 90-30," *Digital Investigation*, vol. 22, pp. S26–S38, 2017.

[42] E. Por, M. van Kooten, and V. Sarkovic, "Nyquist–shannon sampling theorem," *Leiden University*, vol. 1, no. 1, 2019.

[43] G. Sharma, K. Umapathy, and S. Krishnan, "Trends in audio signal feature extraction methods," *Applied Acoustics*, vol. 158, p. 107020, 2020.

[44] A. V. Oppenheim and R. W. Schafer, "From frequency to quefrency: A history of the cepstrum," *IEEE signal processing Magazine*, vol. 21, no. 5, pp. 95–106, 2004.

[45] 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, 2015, pp. 18–25.

[46] S. K. Mahanta, A. F. U. R. Khilji, and P. Pakray, "Deep neural network for musical instrument recognition using mfccs," *Computación y Sistemas*, vol. 25, no. 2, pp. 351–360, 2021.
[47] A. Singhal *et al.*, "Modern information retrieval: A brief overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35–43, 2001.
[48] A. Singh, A. Yadav, and A. Rana, "K-means with three different distance metrics," *International Journal of Computer Applications*, vol. 67, no. 10, 2013.
[49] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," *Advances in neural information processing systems*, vol. 25, 2012.
[50] L. Muda, M. Begam, and I. Elamvazuthi, "Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques," *arXiv preprint arXiv:1003.4083*, 2010.
[51] P. Senin, "Dynamic time warping algorithm review," *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, vol. 855, no. 1-23, p. 40, 2008.
[52] J. Malik, D. Girdhar, R. Dahiya, and G. Sainarayanan, "Reference threshold calculation for biometric authentication," *IJ Image, Graphics and Signal Processing*, vol. 2, pp. 46–53, 2014.
[53] L. M. McDowall and R. A. Dampney, "Calculation of threshold and saturation points of sigmoidal baroreflex function curves," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 291, no. 4, pp. H2003–H2007, 2006.
[54] S. Herbold, J. Grabowski, and S. Waack, "Calculation and optimization of thresholds for sets of software metrics," *Empirical Software Engineering*, vol. 16, pp. 812–841, 2011.
[55] Z. Basnight, J. Butts, J. Lopez Jr, and T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76–84, 2013.
[56] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. A. Mohammed, and S. A. Zonouz, "Hey, my malware knows physics! attacking plcs with physical model aware rootkit." in *NDSS*, 2017, pp. 1–15.
[57] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, 2015, pp. 13–24.