CPS Device-Class Identification via Behavioral Fingerprinting: From Theory to Practice

Leonardo Babun, *Member, IEEE*, Hidayet Aksu, and A. Selcuk Uluagac

***Abstract*— Cyber-Physical Systems (CPS) utilize different devices to collect sensitive data, communicate with other systems, and monitor essential processes in critical infrastructure applications. However, in the ecosystem of CPS, unauthorized or spoofed devices may danger or compromise the performance and security of the critical infrastructure. The unauthorized and spoofed devices may include tampered pieces of software or hardware components that can negatively impact CPS operations or collect vital CPS metrics from the network. Such devices can be outsider or insider threats trying to impersonate other real CPS devices via spoofing their legitimate identifications to gain access to systems, steal information, or spread malware. Device fingerprinting techniques are promising approaches to identify unauthorized or illegitimate devices. However, current fingerprinting solutions are not suitable as they disrupt critical real-time operations in CPS due to the nature of their extensive data analysis or too much overhead on the devices' computational resources. To address these concerns, in this work, we propose STOP-AND-FRISK (S&F), a novel fingerprinting framework to identify CPS device classes and complement traditional security mechanisms in CPS. S&F is based on a secure challenge/response mechanism that analyzes the behavior of the CPS devices at both the hardware and OS/kernel levels. Specifically, the proposed novel mechanism combines system and function call tracing techniques, signal processing, and hardware performance analysis to create specific device-class signatures. Then, the signatures are correlated against known behavioral ground-truth to identify the device types. To test the efficacy of S&F extensively, we implemented a realistic testbed that included different classes of CPS devices with a variety of computing resources, architectures, and configurations. Our experimental results reveal an excellent rate on the CPS device-class identification. Finally, extensive performance analysis demonstrates that the use of S&F yields minimal overhead on the CPS devices' computing resources.**

***Index Terms*— Cyber-physical systems, device-class fingerprinting, correlation, system calls, function calls, hardware performance.**

I. INTRODUCTION

AT THE core of the Cyber-Physical Systems (CPS) (e.g., transportation systems, smart grid, gas and oil plants), devices such as Remote Terminal Units (RTUs), Programmable Logic Controllers (PLCs), and Intelligent Electronic Devices (IEDs) are utilized to collect sensitive data from the infrastructure, provide two-way communication capabilities, and monitor the health of the CPS operations in real-time [1]–[3]. However, these devices also present an opportunity for attackers to have access to sensitive information and the critical CPS infrastructure [4]–[6]. For instance, insiders can impersonate real CPS devices via spoofing attacks to gain access to the systems, steal information, make other devices in the network to behave erratically, or spread malware (Figure 1) [7]–[11]. Also, illegitimate CPS devices may have installed unauthorized pieces of software and hardware that could degrade the performance of the devices and compromise the integrity of the CPS network. Recent studies demonstrate that compromised manufacturing stages within the supply chain may facilitate malicious manipulation and modification of CPS devices' components to deliver a downgraded product to the critical infrastructures [12].

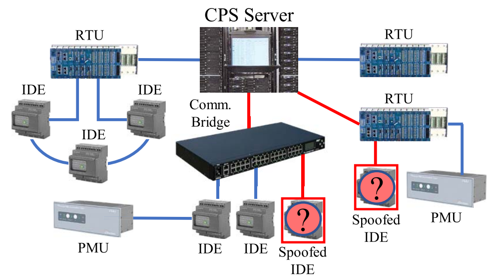


Fig. 1. Sample CPS critical infrastructure network under attack via spoofed IDE devices. The spoofed devices as well as the compromised communication links are marked in red.

Protecting against such attacks stemming from spoofed or unauthorized devices can be very challenging, considering the size and complexity of the CPS infrastructure. For instance, an attacker may use spoofed devices with software and hardware architectures very similar to real devices to increase the chances of stealthy malicious operations. In fact, the spoofed devices can perform the attacks while mimicking real CPS operations. Also, devices with unauthorized components are often capable of supporting most of the CPS operations, but are prone to under-performance and failure when in charge of more demanding and critical tasks. In these scenarios, device fingerprinting techniques are suitable to identify original devices and discriminate them from the unauthorized and the spoofed devices. However, current fingerprinting solutions either require extensive analysis of network packets and device features or study the behavior of very dynamic network metrics [11], [13]–[17]. Thus, in most cases, these solutions introduce significant overhead to devices and systems, putting the execution of critical time-sensitive CPS tasks at risk.

Manuscript received July 20, 2020; revised October 26, 2020 and December 30, 2020; accepted January 7, 2021. Date of publication January 29, 2021; date of current version February 22, 2021. This work was supported in part by the U.S. Department of Energy under Award DE-OE0000779 and in part by the U.S. National Science Foundation under Award NSF-1663051. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Wei Yu. *(Corresponding author: Leonardo Babun.)*

The authors are with the Cyber-Physical Systems Security Lab, Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174 USA (e-mail: lbabu002@fiu.edu; haksu@fiu.edu; suluagac@fiu.edu).

Digital Object Identifier 10.1109/TIFS.2021.3054968

A. Goals and Contributions

In this paper, we address these challenges by proposing STOP-AND-FRISK (S&F),[1] a novel signature-based fingerprinting framework intended to perform CPS device-class identification and complement other traditional security mechanisms in the CPS infrastructure. Specifically, the proposed approach combines system and function call tracing techniques, signal processing, and hardware performance analysis on the devices to implement a secure challenge/response-based fingerprinting solution. By using this approach, S&F studies the behavior of the devices within a CPS infrastructure, focusing on their software/hardware architecture and configuration to identify real CPS devices and discriminate them from unauthorized and spoofed devices. The benefits of S&F are two-fold. First, it combines a secure challenge-response approach with signature-based fingerprinting techniques to identify spoofed devices in the field. Second, it studies the performance of CPS devices to detect devices with degraded software and hardware that could compromise the CPS's critical operations. To test the efficacy of S&F, we implemented a realistic testbed containing different classes of CPS devices with different resources (i.e., memory and CPU), architectures, computing capabilities, and configurations. Our extensive experimental results demonstrate that, by combining OS/kernel behavioral analysis and hardware performance analysis, S&F achieves an excellent rate in the identification of the CPS device classes. Finally, S&F yielded minimal overhead to the CPS devices' computing resources.

The contributions and summary of this work are as follows:

- *Novel Features for CPS Behavioral Analysis*: We proposed a novel combination of features to study the behavior of CPS devices. These features are related to the OS/kernel and hardware performance behaviors of the devices.
- STOP-AND-FRISK : We designed a novel and lightweight signature-based fingerprinting approach that performs CPS device-class identification. The proposed framework combines system and function call information, signal processing, and hardware performance analysis to perform the identification of spoofed and unauthorized devices in the CPS network.
- *Realistic End-to-End Implementation:* We proposed and tested a secure end-to-end deployment mechanism for S&F using a realistic CPS testbed that includes 11 different classes of CPS devices with varying resources and configurations.
- *High Performance:* Our performance evaluation proved that the proposed framework achieves very high accuracy while introducing minimal overhead in the utilization of the computing resources available in the CPS devices.

B. Organization

In Section II, we present background information on the classification of CPS devices based on their behavior. Then, in Section III, we define the problem and present the threat model. Section IV details the architecture of S&F and discusses its main components and analysis approaches. Further, Section V presents implementation and deployment details of S&F. Then, in Section VI, we present and evaluate the experimental results. We discuss the performance of S&F under different compromised scenarios in Section VII. Finally, in Section VIII we discuss the related work and Section IX concludes the paper.

[1]The term STOP-AND-FRISK makes reference to the architectural principles of the proposed fingerprinting mechanism and does not constitute an endorsement to any law enforcement practice.

II. BACKGROUND INFORMATION

A. Cyber-Physical Systems

Cyber-Physical Systems (CPSs) permit the integration of virtual and physical processes. In this context, the physical domain refers to capabilities that act over physical objects. On the other hand, the virtual domain constitutes the set of software and embedded systems intended to guarantee two-way communications, monitor the realization of the physical processes, and provide control [18]. In general, one can characterize CPSs networks by using the following features:

- *Type of Task Performed*: Depending on the specific application and their logical location inside the CPS architecture, the type of task performed by CPS devices may range from just a simple service generated by a local host device to an essential component of a more complex and centralized process. In any case, *individual CPS processes are assumed to be simple, deterministic, and very specific actions* that support the entire system in a distributed topology [19].
- *Resource Availability*: The total amount of available computing resources to perform CPS processes *depends on the type of device performing every particular task*. In general, we can group CPS devices into *resource-rich* and *resource-limited* devices [8], [20]. Resource-limited devices have simple hardware (e.g., single-core CPU and limited memory) and software architecture that allows for the execution of simple, specific tasks. On the other hand, resource-rich devices have more complex Operating System (OS) architecture and run with multi-core CPUs and plenty of memory. These capabilities allow them to execute more complex processes inside the CPS network.
- *Timing Properties*: As we noted before, one of the main goals of the cyber domain in CPS is the monitoring and control of physical processes, which is achieved through rigorous timing control mechanisms. In general, *temporal behavior of CPS is expected to be precise, and should not change too much over time* [19].

B. Device-Class Identification

Traditionally, device-class classification has been performed by considering the branch, model, specific device metrics characteristics, and the activities the devices should perform in the network [21]. S&F implements a more comprehensive approach to identify types of devices in the network that also considers (1) the device behavior at the OS or kernel level and (2) its performance metrics at the hardware level. The main advantage of an approach that includes behavior and hardware performance into its analysis is that it allows for a more secure identification approach that does not depend on device characteristics or metrics that can be spoofed by savvy (or even naive) attackers.

In this work, we consider the following features to define specific classes of CPS devices:

- *Device Metrics*: We use well-known metrics like device's branch, model, and expected functionality to perform initial classification of the devices. The expected functionality of the device mostly refers to the intended application of the device based on the device characteristics (e.g., routers and firewalls may be divided into two different groups based on their unique application and intended use). This preliminary analysis supports the labeling process performed before evaluating S&F (more details in Section V).
- *Device Behavior*: This feature characterizes the device response to specific challenges at the OS and kernel levels. S&F studies the device's behavior based on the collection of system and function calls triggered while reacting to specific challenges.
- *Device Performance*: It characterizes the device's response to specific challenges at the hardware level. S&F studies the device's performance by evaluating the device's memory and CPU utilization as well as the application execution time while reacting to specific challenges or stimulants.

III. Problem Definition and Threat Model

This work assumes a CPS network $\mathcal{N}$ within a critical infrastructure (i.e., the smart grid) that contains devices with various functionalities and computational resources. First, we consider that the supply chain that provides the devices cannot be trusted and is assumed compromised during any of the sourcing, manufacturing, assembling, packing, and delivery processes. Hence, the devices being used in the critical infrastructure may contain unauthorized pieces of hardware and software that could either degrade their performance, or execute malicious or unexpected functionalities. At some point, a network administrator may install a Programmable Logic Controller (PLC) device that was manufactured with a low-cost Central Processing Unit (CPU). The low-end CPU adds additional delays to the device capabilities to react to inputs, which causes degradation to the device's performance and to the CPS network's effectiveness to respond to time-critical tasks [8]. Second, we consider that external attackers may have the capability of inserting fake (i.e., spoofed) devices into the network. These spoofed devices have similar computing characteristics and can execute real tasks with similar performance as legitimate devices. Also, these devices are capable of malicious activities allowing the attackers to gain unauthorized access to different regions of the CPS network and its critical data. Mallory, an insider that has access to the CPS network, could insert a spoofed device (e.g., a BeagleBoard) that has been programmed using open source IEC61850 libraries freely available online [22]. With this device, Mallory can establish communications with real CPS devices and implement GPS spoofing attacks [23].

A. Threat Model

This work considers an attacker (insider or outsider) capable of inserting spoofed devices into a CPS infrastructure. The unauthorized devices spoof real CPS devices and operations to gain access to restricted areas of the network and perform malicious activities. These malicious operations may include: (1) stealing sensitive information, (2) poisoning physical measurements, and (3) creating the conditions to facilitate new types of attacks in the future. We do not consider insiders that have access and can compromise original CPS devices with the same hardware and software configurations used in the field. Instead, we assume that attackers utilize spoofed devices that mimic real CPS network operations to gain access to the network. We also consider unauthorized or illegitimate CPS devices that contain unauthorized software or hardware added during any of the production stages (i.e., raw material sourcing, manufacturing, assembly, testing, and delivery) of the supply chain. The use of unauthorized hardware or software may cause degradation to the device's performance and create unsafe states in the CPS critical infrastructure.

IV. Host-Based CPS Device Class Fingerprinting

In this section, we overview Stop-and-Frisk and present the details about its modules and processes. S&F is a novel and lightweight fingerprinting framework that uses a secure challenge-response approach to extract behavioral data from unknown CPS devices to create specific device signatures. These signatures are then correlated with known device profiles for identification purposes. Specifically, we take advantage of the fingerprinting and identification capabilities of S&F to solve the problems above. First, as S&F considers device performance into its analysis, we can identify devices that are under-performing due to unauthorized hardware of software. Second, the proposed fingerprinting framework can identify spoofed CPS devices that either fail the challenge-response process or that are incapable of generating the expected signature. Finally, S&F's capabilities support automated configuration mechanisms for similar devices that share akin tasks in the network.

A. Overview of Stop-and-Frisk

Assume that there is a CPS critical infrastructure where devices of different types interact to execute a task T. The specific class of some of the devices in the setup is known (i.e., Type A and Type B); however, an *Unknown* device is also present. With S&F, the network operators may be able to verify that (1) the devices in the critical infrastructure are of the expected class (based on the specific tasks they are executing) and (2) they may be able to identify unknown devices and determine if they are authorized to be present or not. Since most of the CPS devices perform time-critical operations in the network, we envision our CPS device class fingerprinting framework to become active at the device's patch- or maintenance-time (i.e., downtime). That way, S&F's operations would focus on individual devices and would put minimal overhead on the systems. Such operations require the interaction of two different services: a server-based remote service (running from a remote server that monitors the CPS environment) and a host-based local service (running on the CPS devices).

Figure 2 depicts the general overview of the proposed device-class fingerprinting framework. First, a scheduler running in the remote S&F's server sends a secure request containing a secret challenge to the unknown CPS device (i.e., localhost) at downtime (①). Such a challenge implements the host-based local service that activates the *Device*

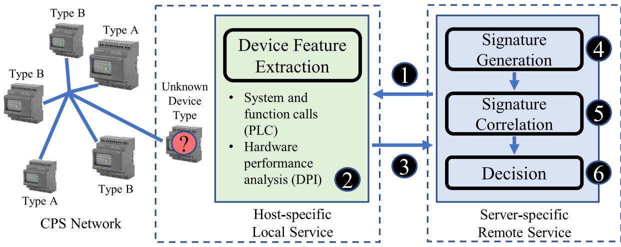


Fig. 2. The architecture of STOP-AND-FRISK proposed to identify different CPS device classes.

Feature Extraction module (②). This module is in charge of running the secret challenge and extracting software- and hardware-related data generated during the device's reaction to the challenge. Specifically, it hooks into the device's activity and extracts lists of system and function calls. Additionally, the Device Feature Extraction monitors the performance of the device regarding CPU utilization, memory utilization, and the execution time of the challenge while extracting the calls. Once finished, the module derives specific features from the collected data. These features are related to the set of functions and system calls triggered, the number of different call types, and their arguments, respectively. Additionally, it computes the CPU utilization, the amount of memory allocated to execute the challenge's response, and the total execution time. Once all the required features are acquired, the local service securely sends them to the remote server using a for further analysis (③). On the server side, the collected features are then utilized to generate the signature of the unknown CPS device inside the *Signature Generation* module (④). Further, the generated signature is correlated (⑤) with ground-truth data previously extracted from known-class CPS devices (i.e., ground-truth) included in the CPS network. Finally, a threshold-based decision algorithm defines the class of the unknown device (⑥).

B. Device Feature Extraction

The first step into the fingerprinting process is to collect the necessary features used to create the unique device-class signature. These features include OS/kernel behavior via system and function calls and hardware performance via memory/CPU utilization and execution time. Compared to current fingerprinting techniques, the implementation of S&F does not require extensive network traffic monitoring. Also, the devices are always monitored at downtime, so no critical CPS operation is interrupted.

1) The Challenge: S&F uses a challenge-response mechanism to generate data that accurately describe the behavior and performance of the unknown CPS devices. Such data is utilized to generate device-type specific signatures that are used later for identification purposes. The Device Feature Extraction module running in the host is the one in charge of securely storing and executing the challenge. There are some advantages associated with executing the challenge locally in the host devices. First, it eliminates the need for creating extra secure channels to deliver files from the remote server, especially at downtime, when connection capabilities may be limited. Second, S&F automatically flags as unauthorized those devices with the wrong reaction to the challenge or where the Device Feature Extraction module is unavailable at test time (which adds additional security layers in cases where capable attackers may try to mimic the performance of S&F). Third, even if the attackers can still implement S&F in the spoofed devices, the final decision depends on the device behavior rather than on metrics that can be easily spoofed. Assuming that the attackers can change the device's behavior and also modify the hardware performance results in S&F's signature, they still need to guess the right values to guarantee that the fake signature strongly correlates with the one stored in S&F's server, for the specific device-type analyzed.

2) Parametric Call List (PCL): S&F utilizes system and function call hooking techniques [24] to collect all the system and function calls that a specific CPS device-class triggers as a response to the pre-determined challenge. From the call lists, the Device Feature Extraction module extracts distinctive device metrics such as (1) the set of specific triggered calls, (2) the total number of calls by type (e.g., *malloc*, *free*, *open*), and (3) the value of specific call's arguments (e.g., the amount of memory allocated by *malloc*). We refer to this list of parameters extracted from the system and function calls as *Parametric Call List* (PCL), which is defined as:

$$PCL_i = \{x_i \in X_i : \exists X_i \wedge X_i \neq \emptyset\}, \tag{1}$$

where PCL_i represents the PCL data extracted from device i, x_i represents the arguments of the calls extracted from device i, and X_i represents the call lists extracted from device i. In general, the hooking technique utilized to extract the system and function calls is a configurable attribute of S&F, and it may be specific to every device's architecture and OS [24]. Also, the effectiveness of the PCL in identifying CPS classes does not depend on the amount of system and function calls included in the PCL. S&F does not impose a specific sampling rate to collect the calls, but relies on well-known hooking techniques to collect the call data. That is, as opposed to a rate of system and function calls collected over time, the number of function and system calls is determined by the specific device's response to the challenge, and would always characterize the device behavior.

During our implementation of S&F, we used library interposition and `ptrace` to hook into the challenge execution and collect the function and system calls. While library interposition is a hooking technique that can be applied to wide range of operating systems to collect system calls, `ptrace` is a UNIX-based system call that is used to hook into specific function calls [24]. Specifically, we instrumented relevant system call definitions and hooked into the process that executes the S&F's challenge in the CPS devices. From there, we were able to build the PCLs with relevant information extracted from every call triggered by the challenge process. In Listing 1, we provide an example of instrumentation of the system call *malloc*, and detail the extra code that we added to enable the collection process.

Finally, to reduce complexity on the host and minimize overhead, once the PCL generation is completed, the collected data is sent to the S&F remote server for processing using secure communication channels. We discuss how to secure S&F's communications in later sections.

3) Device Performance Index (DPI): The second feature used by the proposed framework to identify CPS device classes is the *Device Performance Index* (DPI). Since call lists can be

```
/* An example of a malloc instrumentation */

void* malloc (size_t size) {
    static void* (*my_malloc)(size_t)=NULL;
    if(!my_malloc)
        my_malloc = dlsym(RTLD_NEXT,"malloc");
    calls[index] = 1;
    param[index] = (int)size;
    index++;
    return my_malloc(size);}
```

Listing. 1. An example of a `malloc` sys call instrumentation.

```
/* A sample script to capture GNU data */

/usr/bin/time -f
    "\nReal Time (sec): %e
    \nSystem Time (sec): %S
    \nUser Time (sec): %U
    \nMemory (Kb): %M
    \nCPU: %P\n"
```

Listing. 2. A sample script to capture system performance information from the GNU `time` command.

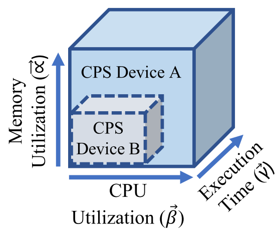


Fig. 3. Three-dimensional representation of the DPI of two different classes of CPS devices. The DPI of device-class A is greater than the DPI of device-class B by around 2x, 1.4x, and 2.5x of memory, CPU utilization, and execution time, respectively.

manipulated by attackers to mimic authentic CPS device operations, S&F includes hardware performance analysis as part of its identification mechanisms. As mentioned in Section II, CPS devices are not expected to change their overall functionality, so the average performance is also expected to describe a near to deterministic behavior over time [19]. Hence, as every class of CPS device has specific functionalities, the device's performance obtained while devices execute the challenge can also be used for identification purposes.

S&F integrates three different metrics into the DPI analysis: memory utilization (α), CPU utilization (β), and execution time (γ). Since every metric is evaluated over a certain time interval t of interest, we consider every metric as a vector quantity as:

$$\begin{aligned}\vec{\alpha} &= \{|\alpha|, \phi_\alpha\},\\ \vec{\beta} &= \{|\beta|, \phi_\beta\},\\ \vec{\gamma} &= \{|\gamma|, \phi_\gamma\},\end{aligned} \tag{2}$$

where $|\alpha|$, $|\beta|$, and $|\gamma|$ represent the magnitude of memory utilization, CPU utilization, and execution time, respectively and ϕ_α, ϕ_β, and ϕ_γ represent the *direction of change* (e.g., positive for increased utilization and negative for decreased utilization) of every metric in t, respect to any previous time interval t_0. Then, we can define DPI as the volume of the parallelepiped whose adjacent sides are defined by the averaged metrics $\vec{\alpha}$, $\vec{\beta}$, and $\vec{\gamma}$ (Figure 3). The parallelepiped volume can be found via the *scalar triple product* of the considered metrics as:

$$\begin{aligned}DPI &= |\overline{\vec{\gamma}} \cdot (\overline{\vec{\alpha}} \times \overline{\vec{\beta}})|,\\ &= \overline{\epsilon_{ijk}\gamma^i\alpha^j\beta^k} : i, j, k \in \ 1, 2, 3 \ldots n,\\ &= det\begin{bmatrix}\gamma_1 & \gamma_2 & \gamma_3 & \ldots & \gamma_n\\ \alpha_1 & \alpha_2 & \alpha_3 & \ldots & \alpha_n\\ \beta_1 & \beta_2 & \beta_3 & \ldots & \beta_n\end{bmatrix}.\end{aligned} \tag{3}$$

where ϵ_{ijk} defines the three-dimensional Levi-Civita symbols that represent the collection of memory, execution time, and memory utilization over time. As with the PCL, there are different mechanisms that can be used to extract the DPI information. Again, it is the goal of S&F's architecture not to enforce a specific implementation strategy to extract the signature information from the CPS devices. Depending on the operating system in use, the implementation of S&F may take advantages of tools available within the system if a secure environment can be guaranteed.[2] In Listing 2, we show a sample script used to capture system performance information related to device hardware's behavior using the GNU `time` command.

S&F computes the DPI for every CPS device class and appends this value to the PCL data to generate the unique device signature. Specifically, S&F computes the DPI in the Signature Generation module before creating the specific device signature. Similarly to the PCL parameter, S&F relies on OS functions running on secure memory regions to collect the DPI information. As explained before, the goal of collecting the DPI is to extract hardware-device information that can be used to characterize the device's behavior.

C. Device Signature Generation

Once S&F completes the device feature acquisition (i.e., PCL and DPI), it generates a unique CPS device-class signature based on the extracted features. These features are selected so they guarantee a comprehensive analysis of the device behavior and performance at both OS/kernel and hardware levels, while keeping unique host characteristics (e.g., network metrics, device ID) out of the analysis to preserve privacy. The signature structure must solely guarantee the correct identification of the device classes so the spoofed or unauthorized devices can be detected and rejected. The final signature for every CPS device type has the following format:

$$[\mu(PCL_{lists}), \mu(DPI_{lists})]. \tag{4}$$

where $\mu(PCL_{lists})$ represents the average of parameters extracted from the system and function calls, and $\mu(DPI_{lists})$ represents the average of CPU utilization, memory utilization, and execution time values included in the DPI.

D. Ground Truth Devices - Learning Phase

The effectiveness of S&F in identifying different classes of CPS devices relies on the use of reliable ground-truth device-class signatures. S&F correlates the ground-truth signatures against the behavior and performance of the unknown devices for identification purposes. In general, ground truth-capable devices must adhere to two basic rules: (1) they must indeed characterize the behavior of the device classes in their network

[2] We discuss details of the performance of S&F in non-secure environments in Section VII.

region, and (2) they must perform stationary deterministic operations inside the CPS infrastructure over time. The first rule guarantees that the device's metrics and the *type of activity* (i.e., the device's specific application inside the CPS network) performed by the ground-truth device are both considered to define its class (Section II). On the other hand, the second rule guarantees reliability. S&F requires that ground-truth devices behave in a deterministic way to guarantee that, if the same challenge is applied, every device class always generates the same signature over time. Steady behavior constitutes a realistic requirement since previous research works have highlighted the deterministic behavior of CPS [19]. Finally, the mechanism of obtaining signatures from ground-truth devices is known as *learning phase*. Once S&F completes this process, it stores the ground-truth device-class signatures into a signature database (SDB).

Ground-truth devices characterize a CPS network region. As discussed in Section II, device-class classification has been traditionally focusing on the branch, model, and the specific application of the devices. We also utilize these metrics to perform preliminary classification of the potential ground-truth devices and to determine how many different classes of devices may be present in the network. We also utilize this information to better organize signatures in the SDB. Thus, we first *group* the different classes of devices in the network based on device-specific metrics. Further, we apply behavioral and performance analysis to extract the signatures from every device class. To evaluate the reliability of the ground truth devices, we calculate the autocorrelation of different PCLs and DPIs obtained while the devices execute the same process (i.e., challenge) but at different time intervals t. We use Equations 5 and 6 to calculate PCL and DPI autocorrelation, respectively, as follows:

$$\rho_{PCL_{t_i},PCL_{t_{i+1}}} = \frac{\sum PCL_{t_i} PCL_{t_{i+1}} - n\overline{PCL_{t_i} PCL_{t_{i+1}}}}{n s_{PCL_{t_i}} s_{PCL_{t_{i+1}}}}, \quad (5)$$

$$\rho_{DPI_{t_i},DPI_{t_{i+1}}} = \frac{\sum DPI_{t_i} DPI_{t_{i+1}} - n\overline{DPI_{t_i} DPI_{t_{i+1}}}}{n s_{DPI_{t_i}} s_{DPI_{t_{i+1}}}}, \quad (6)$$

where PCL_{t_i}, $PCL_{t_{i+1}}$, DPI_{t_i}, and $DPI_{t_{i+1}}$ represent PCL and DPI metrics extracted from the same CPS process, but executed at different time interval, n represents the size of the arrays PCL and DPI, and s represents the standard deviation.

Algorithm 1 details the process of obtaining the ground-truth signatures during the learning phase. Initially, in Line 1, the number of iterations (for averaging purposes) is defined, and the local variables PCL_{lists} and DPI_{lists} are declared and initialized. These variables contain the list of parameters (i.e., PCL and DPI) from every iteration i learning iteration. The goal of running several iterations of the challenge on the ground-truth devices is two-fold. First, we can study the behavior of the devices over time (devices with a random behavior that would not guarantee a stable signature would be rejected). Second, we compute and average all the data from all the iteration to so we consider potential fluctuations of the device behavior on the final signature. Going back to Algorithm 1, in Lines 5 and 6, system and function call tracing techniques are applied to obtain the PCL and DPI at different time intervals t.

Algorithm 1 Generate Signature (Learning Phase)

1: $iterations \leftarrow N$
2: $PCL_{lists} \leftarrow null$
3: $DPI_{lists} \leftarrow null$
4: **for** $i = 0$ to $iterations - 1$ **do**
5: $\quad PCL_{lists}[i] \leftarrow getParamList()$
6: $\quad DPI_{lists}[i] \leftarrow getDPIndex()$
7: **end for**
8: **for** $i \in 0 \ldots size(PCL_{lists}) - 1$ **do**
9: $\quad gTVecPCL \leftarrow \rho_{y_i, y_{i+t}}(PCL_{lists}[i], PCL_{lists}[i+1])$
10: **end for**
11: **for** $i \in 0 \ldots size(DPI_{lists}) - 1$ **do**
12: $\quad gTVecDPI \leftarrow \rho_{y_i, y_{i+t}}(DPI_{lists}[i], DPI_{lists}[i+1])$
13: **end for**
14: $grdThPCL \leftarrow \mu(gTVecPCL)$
15: $grdThDPI \leftarrow \mu(gTVecDPI)$
16: **if** $(grdThPCL \ \& \ grdThDPI) > \xi$ **then**
17: $\quad SDB \leftarrow [\mu(PCL_{lists}), \mu(DPI_{lists})]$
18: **end if**

Algorithm to obtain CPS ground truth device-based signatures during the learning process.

To calculate autocorrelation between different challenge iterations, S&F converts the PCL lists into random variables R_{PCL}. To do so, the framework assigns weights δ_{PCL_i} to every different type of function and system call in the PCL. STOP-AND-FRISK follow a specific pattern to assign the weight values, which depends on the type of system and function call in the PCL. For instance, if the PCL list that characterize the device behavior to the challenge contains sys calls of type *malloc* and *free*, the weight assigned to these calls are desire to have certain statistical relationship to preserve the correlation among these calls. In regular system tasks, *malloc* and *free* calls are frequently invoked as part of the same process. Thus, S&F tries to preserve such a correlation by assigning numerical weight values that are also correlated (e.g., $\delta_{PCL_{malloc}} = 2$ and $\delta_{PCL_{free}} = 4$). The result of the weight assignment is a random variable R_{PCL} that takes values between δ_{min} and δ_{max}, and that statistically describes the reaction (i.e., behavior) of the CPS device class to the challenge. The rest of the collected parameters (i.e., call arguments, call amount, and DPI information), are considered without modification for the autocorrelation calculation as they constitute numerical values.

In Lines 9 and 12, the autocorrelation vector between the different time intervals of PCL and DPI is calculated. Later, in Lines 14 and 15, the average of all autocorrelation values is computed. Finally, if the autocorrelation values of PCL and DPI are greater than the threshold ξ (Line 16), the algorithm accepts the evaluated CPS device as ground-truth and stores its signature into the SDB (Line 17). In practice, the value of the threshold ξ is agnostic and can be determined based on the specific characteristics of the operations in the ground-truth device.

E. Signature Correlation and Decision - Prediction Phase

During decision, S&F correlates the signature obtained from unknown CPS devices against the ground-truth signatures

Algorithm 2 Identify Device Class (Prediction Phase)

```
1: CPSsignList ← SDB
2: iterations ← N
3: PCL_lists, DPI_lists, CPSdeviceID ← null
4: signatue ← null
5: for i = 0 to iterations − 1 do
6:   PCL_lists[i] ← getParamList()
7:   DPI_lists[i] ← getDPIndex()
8: end for
9: signature ← [μ(PCL_lists), μ(DPI_lists)]
10: corrXYmax ← 0
11: for i = 0 to size(CPSsignList) − 1 do
12:   corrXY ← ρ_{x,y}(CPSsignList(i), signature)
13:   if corrXY > δ & corrXY > corrXYmax then
14:     CPSdeviceID ← i
15:     corrXYmax ← corrXY
16:   end if
17: end for
```

STOP-AND-FRISK algorithm for CPS device class identification.

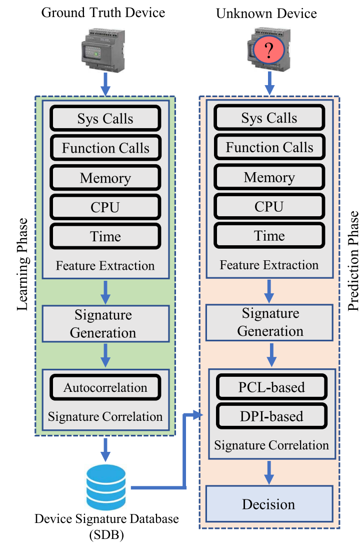


Fig. 4. Proposed device-class identification framework using call tracing techniques, signal processing, and device performance analysis.

stored in the SDB. This process is known as *prediction phase* and is detailed in Algorithm 2.

1) Signature Correlation: The process for obtaining the signature of the unknown CPS device follows similar steps as in Algorithm 1. However, this time the system is not required to calculate autocorrelation, as S&F assumes that all devices in the network are capable of generating a valid signature (Lines 2, 6, and 7 in Algorithm 2). Once the unknown signature is finally generated in the server (Line 9), S&F calculates the correlation between $signature$ and all the unique CPS ground-truth signatures from the SDB (Line 12) using Equation 7 and Equation 8:

$$\rho_{PCL_X,PCL_Y} = \frac{\sum PCL_{X_i} PCL_{Y_i} - n\overline{PCL_X PCL_Y}}{n s_{PCL_X} s_{PCL_Y}}, \quad (7)$$

$$\rho_{DPI_X,DPI_Y} = \frac{\sum DPI_{X_i} DPI_{Y_i} - n\overline{DPI_X DPI_Y}}{n s_{DPI_X} s_{DPI_Y}}, \quad (8)$$

where n represents the size of PCL_X (i.e., ground truth PCL), PCL_Y (i.e., unknown device PCL), DPI_X (i.e., ground-truth's DPI), and DPI_Y (i.e., unknown device's DPI), $\overline{PCL_X}$, $\overline{PCL_Y}$, $\overline{DPI_X}$, and $\overline{DPI_Y}$ represent the mean value, and s_{PCL_Y}, s_{PCL_Y}, s_{DPI_X}, and s_{DPI_Y} represent the standard deviation, respectively.

After computing both ρ_{PCL_X,PCL_Y} and ρ_{DPI_X,DPI_Y} correlations, the decision process starts. The logical condition in Line 16 evaluates that (1) the correlation between the unknown device and signature i from the database is over a certain threshold δ and (2) this value of correlation is a maximum obtained from all the iterations in Algorithm 2. If such a condition holds, the unknown CPS device is deemed to be the same CPS device class as CPS device i from the database (Line 14). On the other hand, if the condition in Line 13 is never satisfied, the unknown device is classified as $Unknown$, and flagged by S&F.

2) Decision: One can observe that, from Algorithms 1 and 2, the value of the correlation threshold δ is a configurable parameter that can be inferred based on the threshold ξ used to generate the ground-truth device signature. That is, the value of ξ that achieves the highest accuracy during the learning phase is selected as δ value for the prediction phase. With this, we guarantee the highest accuracy for S&F decisions. In addition, since the signatures are generated as a result of the device's response to a controlled stimulus (i.e., challenge), S&F's approach minimizes potential decision errors due to signature deviations. A device's response that clearly deviates from a known signature should be considered as from a different device (known or unknown), as opposed to a signature error due to different random processes running in the device. The signature generation process hooks into the specific challenge execution and does not consider other processes running in the device. Also, as we explain in Section IV-D, we extracted the ground-truth signatures after averaging different learning iterations. With this, we guarantee that the final device signatures considers "expected" deviations that may occur to the device signature over time, due to random kernel and operating system operations.

Finally, the S&F's processes described in Algorithms 1 and 2 are summarized in Figure 4. One can notice that, for both the learning and prediction phases, S&F reuses the first two modules of the proposed architecture since they contain similar operation steps. In both phases, S&F needs to extract features and create signatures from ground truth or unknown devices, respectively.

F. Space- and Time-Complexity Analysis

We analyze the time and space complexity of S&F's algorithms, which highly depends on their specific implementation strategies. For instance, the correlation calculation can be performed either in time or frequency domains, which would differently impact their overall complexity. In addition, highly efficient correlation implementations can be used to reduce the use of computational resources and time [25], [26]. We perform analysis on the space- and time-complexity of S&F's implementation (see Algorithms 1 and 2). We do not consider the encryption and signing steps used to guarantee the secrecy and freshness of S&F'a operations, but the

TABLE I

REPRESENTATIVE SAMPLE OF THE 11 DIFFERENT CLASSES OF CPS DEVICES IN THE TESTBED. WE ALLOWED MORE THAN ONE DEVICE PER CLASS TO ALSO EVALUATE THE EFFECTIVENESS OF S&F IN GROUPING DEVICES FROM THE SAME CLASS TOGETHER

Class #	Device ID	Hardware Specifications	Operating System	Release
1	BB_1	AM355x Cortex-A8 @ 1GHz 512MB DD3 RAM	Linux Beaglebone 4.1.5	Debian 8.3 Jessie
2	BB_2	AM355x Cortex-A8 @ 1GHz 512MB DD3 RAM	Linux Beaglebone 3.8.13	Debian 7.9 Wheezy
2	BB_3	AM355x Cortex-A8 @ 1GHz 512MB DD3 RAM	Linux Beaglebone 3.8.13	Debian 7.9 Wheezy
3	GZ	AMD GX210HA @ 1GHz 1GB DD3 RAM	Linux Ubuntu-mate 4.4.0	Ubuntu 16.04 xenial
4	LM_1	A10 Cortex-A8 @ 1GHz 512MB DD3 RAM	Linux A10Lime 3.4.90	Debian 3.4.90
4	LM_2	A10 Cortex-A8 @ 1GHz 512MB DD3 RAM	Linux A10Lime 3.4.90	Debian 3.4.90
5	ODR	A15 Cortex Quad core @ 2GHz A7 Quad core @ 900MHz 2GB DD3 RAM	Linux Odroid 3.10.96	Ubuntu-Mate 16.04
6	OPi_1	H3 Quad core Cortex-A7 @ 1GHz 1GB DD3 RAM	Linux Orange Pi Kali 3.4.39	Kali 2.0
6	OPi_2	H3 Quad core Cortex-A7 @ 1GHz 1GB DD3 RAM	Linux Orange Pi Kali 3.4.39	Kali 2.0
7	RPi_1	Cortex-A7 @ 900MHz 1GB DD3 RAM	Linux Raspberry Pi 4.1.7	Raspbian 8.0 jessie
7	RPi_2	Cortex-A7 @ 900MHz 1GB DD3 RAM	Linux Raspberry Pi 4.1.7	Raspbian 8.0 jessie
8	RPi_3	Cortex A53 Quad core @ 1.2GHz 1GB DD3 RAM	Linux Raspbery Pi 4.4.11	Raspbian 8.0 jessie
8	RPi_4	Cortex A53 Quad core @ 1.2GHz 1GB DD3 RAM	Linux Raspbery Pi 4.4.11	Raspbian 8.0 jessie
9	RPi_5	ARM1176 @ 700MHz 512MB DD3 RAM	Linux Raspberry Pi 4.1.13	Raspbian 7.0 wheezy
10	LTP_1	Intel Core i7-2760 QM @ 2.4GHz 6GB DD3 RAM	Linux 3.19.0	Ubuntu 14.04 trusty
11	LTP_2	Intel Core i5-5200 @ 2.7GHz 6GB DD3 RAM	Linux 4.4.0	Ubuntu 16.04 xenial

mathematical operations and analysis performed during data extraction, signature generation, correlation, and decision (see Figure 4). Recall that within the host, S&F only collects device data to generate the corresponding signature. The worst case performance scenario would create and query lists of n PCL and DPI elements, yielding an upper bound time and space complexity of $O(n)$. On the server side, the two most complex operations, autocorrelation and correlation, have similar time and space complexity. In total, considering PCL and DPI datasets of size n, S&F computes the correlation calculations within control flow statements of n iterations. These calculations can be executed in constant time, yielding an upper bound time complexity of $O(n)$. On the other hand, the space complexity depends on the terms needed to calculate the correlation. For devices with storage limitations, it is possible to only save the value of the average calculations, yielding an overall space complexity of $O(log(M))$; where M would be the maximum value within the PCL and DPI lists. In this case, however, it would be necessary to spend more time in calculating a higher number of subtractions. In a different approach, the algorithm could store the average values and the differences, allowing for a faster computation but yielding an upper bound space complexity of $O(nlog(M))$.

V. IMPLEMENTATION DETAILS OF S&F

In this section, we assume a realistic CPS network and present implementation and deployment details of S&F. Also, we detail the key characteristics of the devices included in the testbed.

A. Realistic CPS Testbed Implementation

We implemented a CPS testbed considering the characteristics of the CPS device classes described in Section III (Table I). These characteristics were included in our testbed as follows:

1) Diversity in Hardware and Software Resources: We included 11 different classes of CPS devices with a variety of available computing resource and different hardware/software configurations. This diversity makes our testbed representative of a large population of real CPS devices; from small devices with limited resources to resource-rich devices [8], [20]. Despite the expected diversity, we allow certain similarities among the different device classes. We explain later in this section how we use these similarities to challenge the identification capabilities of S&F.

2) Discriminate then Regroup: We expect the proposed classification technique to be effective in discriminating different classes of CPS devices (to avoid false positives outcomes). However, we also expect S&F to be capable of grouping devices of the same class together (to avoid false negatives outcomes). To evaluate both metrics, we allowed more that one device of the same class in some cases (Table I). With this, we may determine how S&F performs on classifying types of devices into different classes and devices of the same type into a common class.

3) CPS-specific Tasks and Processes: During the learning phase, the devices included in our testbed performed real CPS networking operations following the IEC61850 communication standard [27]. The IEC61850 is a protocol-suite that defines the communication standards for electrical substation automation systems. To implement this functionality, we utilized an open-source version of the IEC1850 standard (i.e., $libiec61850$) that is freely available online [22]. In addition, we designed challenge-response approaches for the prediction phase that are also suitable for CPS operations (e.g., extract specific device information, write it into a file, and further delete the file from memory).

4) Multiple But Similar OSes: The CPS devices included in the testbed run 11 different versions of Linux-based OSes. Using different versions of Linux constitutes a realistic approach since most of the legitimate CPS devices used in the field include some variant of Unix-based OS [28]. Additionally, the open-source approaches in S&F enable the implementation of flexible solutions that would not impact the evaluation process of S&F. Indeed, previous research works have detailed system and function call hooking techniques that can be applied to all major operating systems [24]. In fact, obtaining the PCL and DPI data from devices with different operating systems is independent of S&F's architecture and more related to specific implementation challenges. Finally, despite their noted differences, we purposely kept similarities among the different devices classes in the testbed. For instance, as shown in Table I, most of the devices are Debian-based systems using ARM CPU architecture. Such an implementation approach would additionally challenge S&F into identifying device classes based on small deviations of software and hardware-based features instead of taking advantages of very noticeable architectural differences.

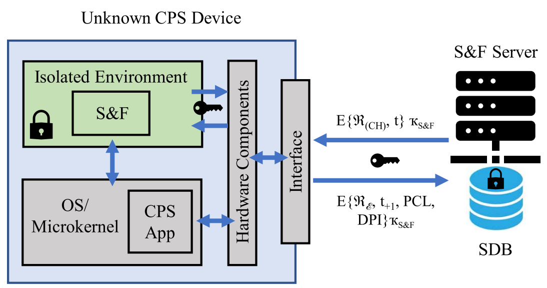


Fig. 5. S&F encrypts and signs the challenge request that is sent to an unknown device. The S&F's module inside the host processes the request and sends back the signed and encrypted response.

For fairness, we followed a *black-box* approach where an independent third-party assigned specific labels (i.e., IDs) to the devices in the testbed based on their hardware and OS characteristics. Table I details the results of the labeling process. For instance, devices with CPU of type AM355x Cortex-A8 1GHz clock and 512MB of DD3 RAM, which also has installed the same type of OS (Linux Beaglebone 3.8.13) are considered as from the same "Class #2" (e.g., BB_2 and BB_3). However, devices with the same hardware characteristics but different OS are considered as from different classes (e.g., BB_1 receives the label "Class #1"). With this labeling strategy, we demonstrate that S&F is capable of fine-grained classification, where devices with very similar OS and hardware characteristics can fall into different classes as their behavior differ. The rationale behind including many different devices classes in the testbed is justified as CPS infrastructures may contain a high diversity of devices that behave differently, but that share similar software and hardware characteristics. Also, the hardware and software characteristics of the devices may vary considerably in real scenarios. Specifically, in the CPS infrastructure, one can find devices with limited and rich computing resources, various software configurations, and different architectures. In this context, the hardware and software characteristics of the CPS devices are specific to their functionalities and applications. As a consequence, small changes in the CPS devices' configuration should be highly noticeable in their general behavior [8]. In this work, we exploit these characteristics of CPS networks and devices to propose a fingerprinting technique that identifies different CPS device classes based on their behavior and performance.

B. Secure Deployment of S&F

We guarantee secure deployment and implementation of STOP-AND-FRISK via device attestation [29]. Figure 5 depicts a high-level representation of the end-to-end attestation process implemented to support S&F. First, a centralized scheduler running in the remote S&F server sends a secure request $\mathcal{R}$ containing the specific challenge $\mathcal{C}_{\mathcal{H}}$ to the unknown CPS device. The request is secured via the use of encryption, digital signature, and timestamps. On the one hand, the use of encryption prevents passive observers (i.e., eavesdroppers) from learning the structure and content of the challenge. On the other hand, digitally signing the requests allows the CPS device to verify that the challenge comes as part of a legitimate S&F request that has not been illegally modified (i.e., preserving integrity). Finally, the use of timestamps prevents attackers from using old S&F requests to implement reply attacks. The encryption step of S&F uses a combination of symmetric and asymmetric encryption to protect the challenge and the device responses from external attackers. First, the requester generates a session key $\mathcal{S}$ that is used to encrypt the challenge $\mathcal{C}_{\mathcal{H}}$. Then, the session key is also encrypted, along with a timestamp $\mathcal{T}$ using the public key $\mathcal{PB}$ of the targeted host, and appended to the $\mathcal{C}_{\mathcal{H}}$. Finally, the entire request is signed using the targeted host's public certificate. Once the CPS device receives the request, it is re-directed to the isolated secure environment where the host S&F resides. There, the signature of $\mathcal{R}$ is verified, and the part of the request containing the session key is decrypted using the private key of the device host $\mathcal{PK}$. Then, the rest of the request corresponding to the challenge $\mathcal{C}_{\mathcal{H}}$ is decrypted using the session key. We use software and, where possible, hardware-based isolation to prevent unauthorized access and modification of S&F's modules. As we consider Linux-based devices in our CPS testbed, we first set up standard UNIX access control mechanisms, POSIX capabilities to limit access to the isolated area. Further, we ran the S&F's modules in Virtual Machines (VM) hosted in hypervisors implemented using the Kernel-based Virtual Machine (KVM) [30]. We envision that, in cases where the architecture of the CPS device allows, S&F may take advantage of the principle of Platform Security Architecture (PSA) featuring Trusted Execution Environment (TEE) and TrustZone [31] to guarantee software isolation via hardware separation.

Once the CPS host processes the challenge, it generates a response $\mathcal{R}_{\mathcal{E}}$ that contains specific system and function calls triggered by the challenge in the host CPS device. These calls are collected to create the PCL. In addition, the device's hardware reaction to the challenge is collected to generate the DPI. The PCL and DPI information are combined in a form of a response $\mathcal{R}_{\mathcal{E}}$ to S&F's challenge, which is encrypted and signed using the same session key $\mathcal{S}$ and the public key and certificate of the STOP-AND-FRISK server. Finally, the response $\mathcal{R}_{\mathcal{E}}$ is sent back to the S&F server which processes it and decides if the host device class is authorized or not.

VI. PERFORMANCE EVALUATION

In this section, we present experimental results that demonstrate the effectiveness of S&F to fingerprint and identify different classes of CPS devices. With this performance evaluation, we aim to answer the following research questions:

- **RQ1: Learning Phase.** How the proposed framework performs during the learning phase? (Section VI-B).
- **RQ2: Prediction Phase.** What is the accuracy of S&F in fingerprinting CPS devices while using (1) PCL correlation only, (2) DPI correlation only, and (3) both PCL and DPI analysis simultaneously? (Section VI-C).
- **RQ3: Overhead.** What is the overhead introduced by S&F to the CPS devices? (Section VI-D).

In all the evaluation experiments, we computed the results after averaging 30 different runs for all the covered scenarios. Every scenario comprised the execution of the challenge-response process. Further, we applied Algorithm 1 and Algorithm 2 on the devices included in our testbed (Table I) to (1) generate a trustworthy signature database, (2) evaluate the correlation between the signatures and the devices' behavior, (3) identify different classes of CPS devices, and finally, (4) evaluate the overhead that S&F introduces to the CPS devices' computing resources.

During the learning phase, we studied the PCL and DPI behavioral characteristics of the CPS devices included in the testbed. We expect the devices to have a PCL and DPI behavior that is deterministic enough to guarantee repeatability (i.e., test-retest reliability) during the signature generation step. To evaluate determinism, we calculated the statistical autocorrelation among signatures extracted from the same device while it executed similar CPS processes at different time intervals. For the cases where a deterministic behavior was identified, the devices were accepted as ground-truth. Finally, for this phase of the evaluation, we set the threshold $\xi = 0.7$ in Algorithm 1 (Section IV), which marks the point from moderate to strong statistical autocorrelation that is widely accepted in the literature [32].

Further, S&F applied the challenge-response approach discussed in Section IV to extract data and create the devices' signatures. These signatures were then stored in the SDB for identification purposes during the prediction phase. For the prediction phase, we also set the threshold δ to 0.7. In real-life scenarios, this value of δ may be adjusted depending on the specific behavioral characteristics of the devices in the CPS network. For instance, in practical applications of S&F, the analysis over a group of well-known devices (i.e., control group) may give the best decision threshold value for the specific network region. Finally, since we are working with UNIX-based OSes, we utilized library interposition [33] and *ptrace* function [34] to extract the lists of system and function calls, respectively and generate the PCL. Also, we utilized the *top* and GNU *time* commands to extract information related to execution time as well as CPU and memory utilization for the DPI analysis. Finally, we used the same challenge to trigger responses from all the tested devices. That way, S&F detects and flags the differences between devices classes based not on the CPS tasks they process, but only on their relative differences in specific behavior of kernel and hardware performance. Finally, for statistical evaluation of the PCL, we converted the list of system and function calls into random variables. We followed the process of assigning λ_i weights to every specific type of system and function call. To maintain the statistical correlation among similar processes, we assign close weight values to system and function calls that are related to a similar process. For instance, we may assign the values λ and $\lambda + 1$ to system calls of the type *malloc* and *calloc*, respectively.

A. Performance Metrics

To evaluate the performance of S&F, we compute the standard statistical metrics of accuracy, recall, precision, and specificity. We define these metrics as follows:

$$A_{CC} = \frac{(T_P + T_N)}{(T_P + T_N + F_P + F_N)}, \tag{9}$$

$$R_{EC} = \frac{T_P}{(T_P + F_N)}, \tag{10}$$

$$P_{REC} = \frac{T_P}{(T_P + F_P)}, \tag{11}$$

$$S_{PEC} = \frac{T_N}{(T_N + F_P)}, \tag{12}$$

where T_P stands for true positive or the case where a CPS device is correctly classified as of some specific class; T_N stands for true negative or the case where a CPS device is correctly classified as of not from some specific class; F_P stands for false positive or the case where a CPS device is identified using the wrong signature. Finally, F_N stands for false negative or the case where a CPS device whose signature has been previously stored in the database cannot be correctly identified.

B. Performance of S&F During the Learning Phase

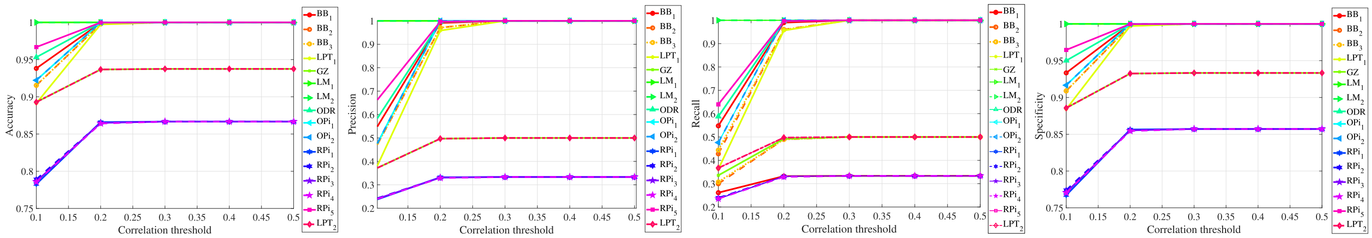
As described in Section IV, the first step towards applying S&F is to find a reliable set of unique signatures that characterize the different CPS device classes. The signature generation process uses statistical autocorrelation between different realizations of PCL and DPI to determine if deterministic behavior can be inferred from different time interval realizations of a similar process in the devices. Moderate to high values of autocorrelation (typically over 0.7 [32]) indicate that the specific CPS device (which is assumed to be a trusted CPS device with no prior tampering or unauthorized components) can be used as ground-truth to create a reliable signature for its class.

Figure 7(a) depicts the evaluation results after applying Algorithm 1 (Section IV) over randomly selected devices from all the different classes included in the testbed. One can observe that, in all the cases, the autocorrelation values are over the threshold ξ, which indicates a deterministic behavior of the devices over time. Again, we obtained these results after 30 different PCL and DPI runs in every device at different time intervals. These results constituted a strong indicator that ground-truth signatures can be obtained for all the devices in the testbed. Finally, once the ground-truth CPS devices were identified, we generated the signatures and stored them into the SDB.

C. Performance of S&F During the Prediction Phase

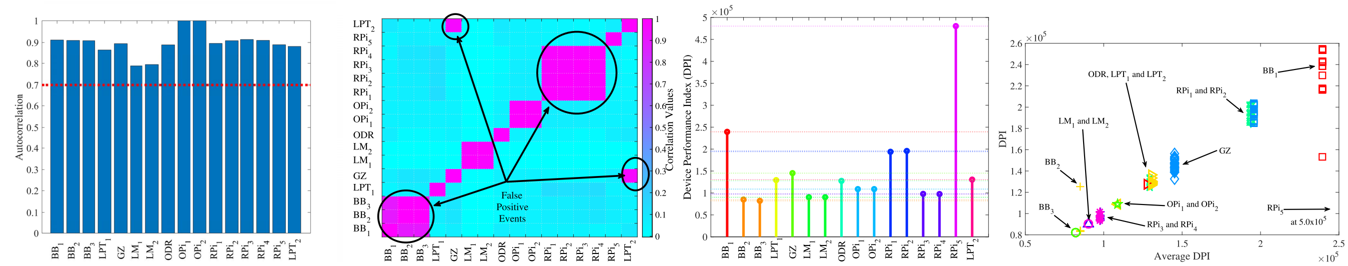
The primary goal of S&F is to classify CPS devices to the right class based on similarities in OS's behavior, hardware performance, and configuration. Additionally, S&F must be able to cluster devices from the same class effectively. Before executing the prediction phase, S&F securely sent a challenge to the unknown devices, collected its features, and created their unique signatures. S&F applied system and function call hooking techniques (i.e., library interposition and *ptrace*) to generate the PCL of the unknown devices. Similarly, S&F extracted the hardware performance features used to calculate the DPI of every single device. Once these processes were completed, the host-based portion of S&F (Figure 2) sent this information to the remote server for processing. The prediction phase was then initiated by applying Algorithm 2 (Section IV) to the collected data.

To thoroughly test the efficacy of S&F and evaluate the real contribution of every fingerprinting feature that we have chosen, we first analyzed the performance of the framework by using PCL- and DPI-based correlation only. Then, we evaluated how the results improved after combining both analyses.



(a) Accuracy (b) Precision (c) Recall (d) Specificity

Fig. 6. Evaluation of the experimental results after considering PCL-based correlation only: (a) accuracy, (b) precision, (c) recall, and (4) specificity. One can observe that, in some cases, lower accuracy results were obtained due to false positives among some device classes. These results were improved after combining PCL-based correlation with DPI analysis (Figure 9).



(a) Autocorrelation (b) Confusion Matrix (c) Average DPI (d) Spatial DPI

Fig. 7. Performance of S&F; (a) during the learning phase, (b) after applying PCL-based detection only, and (c), (d) after applying DPI-based detection only.

1) PCL-Based Correlation Analysis: Figure 6 depicts details of the performance evaluation metrics after applying PCL-based correlation only. S&F achieved lower accuracy values of slightly over 86% for devices RPi_4, RPi_2 and BB_1. Also, it obtained accuracy results of over 94% for devices of classes GZ, BB_3, LPT_2 and BB_2, respectively. We believe that these results are caused by similarities in kernel behavior from different classes of devices while processing the challenge. The metrics of precision, recall, and specificity were also affected by these false-positive events. Figure 7(b) represents the confusion matrix (NxN PCL-based correlation matrix) among all the device classes in the testbed for PCL-based analysis only. A darker color indicates a high correlation, while lighter colors indicate lower correlation values between PCLs from different devices. As per Table I, one should expect a total of 11 different CPS device classes based on the different computing resources and software/hardware configurations. However, in this case, S&F was only able to identify 9 out of a total of 11 different classes of devices. From Figure 7(b), one can observe that, for instance, the proposed framework mistakenly confused GZ and LPT_2 as of the same class. Also, the devices BB_1, BB_2, and BB_3 were wrongly grouped together.

2) DPI-Based Correlation Analysis: We calculated the DPI values for every CPS device in the testbed using the approach proposed in Equation 3. In Figure 7(c), we show the results of the DPI calculation. As observed, several DPI values from different devices were very similar, which negatively impacts the feasibility of using this feature for identification purposes. However, for some specific devices, DPI values were significantly different among device classes. To better understand this analysis, we further represent the obtained DPI values versus the average DPI of all the devices classes in the testbed in Figure 7(d). From this figure, it is clear that DPI analysis may not significantly contribute to discriminating devices from different classes. For instance, one can observe

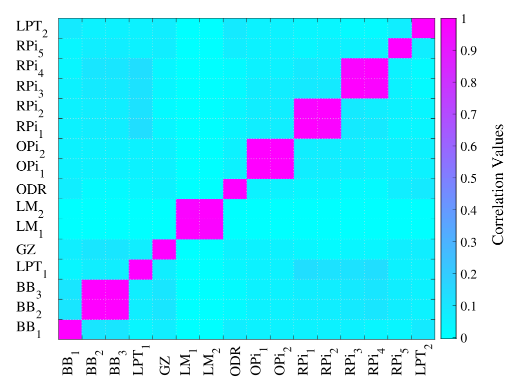
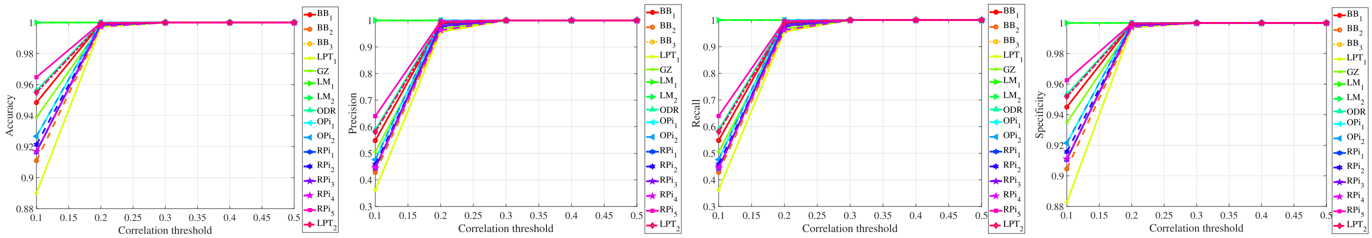


Fig. 8. After combining PCL-based correlation and DPI analysis, S&F was able to identify the 11 different classes of devices included in the CPS testbed.

that devices from different classes like ODR and LPT_1 are wrongly overlapping in Figure 7(d).

3) PCL and DPI Analysis Combined: We further combined both PCL- and DPI-based correlation analysis and obtained a new decision map in Figure 8. This time, S&F was able to identify 11 different device classes by avoiding the false positives events obtained in previous results. Also, performance metrics significantly improved after combining PCL and DPI, if compared with PCL-only analysis. In Figure 9, we detail the new performance metrics results. One can observe that all the different metrics achieved excellent results if compared with the values presented in Figure 6. For the four different metrics, S&F obtained excellent performance with values close to 100%. The excellent detection performance of STOP-AND-FRISK is a direct consequence of some of the design strategies of S&F's architecture. First, the combination of two different features, the PCL and the DPI, into the analysis permits monitoring the behavior of the CPS devices across two different low-correlated dimensions. While the PCL provides information regarding how the OS and kernel handle the applications running on the devices, the DPI characterizes



(a) Accuracy (b) Precision (c) Recall (d) Specificity

Fig. 9. Evaluation of the experimental results after considering correlation and device performance index for decision: (a) accuracy, (b) precision, (c) recall, and (4) specificity. One can notice how the overall metrics improved if compared results shown in Figure 6.

TABLE II

AVERAGE PERFORMANCE METRICS OF STOP-AND-FRISK FOR ALL THE DIFFERENT DEVICES CLASSES EVALUATED

ID	Acc	Pre	Rec	Spec	TPR	FPR	TNR	FNR
BB_1	0.9896	0.9076	1.0000	0.9889	1.0000	0.0111	0.9889	0.0000
BB_2	0.9816	0.8777	1.0000	0.9803	1.0000	0.0197	0.9803	0.0000
BB_3	0.9827	0.8807	1.0000	0.9815	1.0000	0.0185	0.9815	0.0000
GZ	0.9874	0.8948	1.0000	0.9865	1.0000	0.0135	0.9865	0.0000
LM_1	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	0.0000
LM_2	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	0.0000
ODR	0.9912	0.9169	1.0000	0.9906	1.0000	0.0094	0.9906	0.0000
OPi_1	0.9853	0.8952	1.0000	0.9843	1.0000	0.0157	0.9843	0.0000
OPi_2	0.9853	0.8952	1.0000	0.9843	1.0000	0.0157	0.9843	0.0000
RPi_1	0.9831	0.8860	1.0000	0.9819	1.0000	0.0181	0.9819	0.0000
RPi_2	0.9839	0.8869	1.0000	0.9828	1.0000	0.0172	0.9828	0.0000
RPi_3	0.9831	0.8846	1.0000	0.9819	1.0000	0.0181	0.9819	0.0000
RPi_4	0.9829	0.8816	1.0000	0.9817	1.0000	0.0183	0.9817	0.0000
RPi_5	0.9929	0.9273	1.0000	0.9924	1.0000	0.0076	0.9924	0.0000
LTP_1	0.9908	0.9139	1.0000	0.9902	1.0000	0.0098	0.9902	0.0000
LTP_2	0.9774	0.8638	1.0000	0.9759	1.0000	0.0241	0.9759	0.0000
Total	**0.9873**	**0.9070**	**1.0000**	**0.9864**	**1.0000**	**0.0136**	**0.9864**	**0.0000**

the hardware configuration used to support such applications. Second, S&F combines function and system calls into the PCL, increasing the sensitivity of the proposed fingerprinting technique for cases in which similar devices have different OS or kernel configurations. These configurations may define distinctive device classes with different permission levels for users and applications that S&F would not be able to detect without considering kernel- and application-level calls simultaneously. Third, the challenge-response strategy followed by S&F permits the analysis of the devices' behaviors on a controlled environment, which minimizes errors introduced by differences in device tasking over time. Furthermore, we summarize the average performance of S&F for all considered threshold values in Table II. In addition to the already define metrics, we include True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), and False Negative Rate (TNR). The results depicted in Table II demonstrate the effectiveness of S&F in fingerprinting CPS device classes with different behavior.

Finally, we compare S&F's accuracy with the performance of other existing similar tools in Table III. Although there exist tools that combine software and hardware features to implement fingerprinting mechanisms, not all published works share their performance metrics. One can observe that despite some of the tools implement sophisticated analysis based on machine learning algorithms, S&F's performance is competitive, if not better, in all the cases.

TABLE III

WE COMPARE THE PERFORMANCE OF S&F WITH OTHER BEHAVIORAL ANALYSIS-BASED FINGERPRINTING TECHNIQUES. THE ACCURACY OF S&F IS COMPUTED AFTER CONSIDERING DECISION THRESHOLDS FROM 0.1 TO 0.5

Tool	Approach or Technique	Acc (%)
FLOWPrint [35]	ML-based network traffic analysis	89.20
FINAuth [36]	ML-based sensor data analysis	97.66
Z-IoT [37]	ML-based network traffic analysis	92.15
ICS-F [19]	ML-based response/operation time	95.50
GTID [38]	ML-based hardware and clock	93.00
BT [39]	ML-based Bluetooth traffic analysis	98.50
S&F	PLC and DPI	98.73

D. Overhead Introduced by S&F

Table IV summarizes the overhead introduced by S&F to the CPS devices. Despite the benefits of the proposed technique, the expected limitations of CPS devices' computing resources do not allow for excessive overhead (Section III). We calculated the system overhead by analyzing the performance of the devices (1) under normal operating conditions and (2) while executing our fingerprinting technique. In Table IV, ET refers to Execution Time, CPU refers to CPU utilization, and MEM refers to memory utilization. Also, the term $TotalMEM$ refers to the percent of memory that S&F utilizes, out of the total memory available in every device. One can observe that the maximum overhead introduced by S&F, regarding increments in execution time, memory, the percentage of total memory, and CPU utilization, is 0.04%, 19.8%, 0.0218%, and 1.5%, respectively. Different from other existing solutions, the fingerprinting framework proposed in this work did not require to monitor the CPS devices over long periods. S&F was able to identify the different device classes with high accuracy from the kernel's behavior and hardware performance data extracted after only 30 sec of challenge execution time. Finally, one can notice that, since S&F is executed during the device's downtime, the current overhead introduced, besides minimal, only affect the devices while they are not performing time-critical CPS operations.

VII. DISCUSSION

In this section, we discuss the performance of S&F under different compromised scenarios.

A. Fingerprinting vs. Anomaly Detection

We propose S&F as a host-based signature-based fingerprinting mechanism capable of identifying unauthorized device

TABLE IV
AVERAGE OF SYSTEM OVERHEAD INTRODUCED BY S&F ON THE DEVICES INCLUDED IN THE TESTBED

Devices	Without STOP-AND-FRISK			With STOP-AND-FRISK			Overhead			
	RT (s) value	Mem (KB) value	CPU (%) value	RT (s) value	Mem (KB) value	CPU (%) value	ET (s) %	Mem (KB) %	Total Mem (KB) %	CPU (%) %
BB_1	60.050	1304	3.0	60.055	1343	3.0	0.008	3	0.0039	0
BB_2	60.025	652	2.0	60.049	696	2.03	0.04	6.7	0.0085	1.5
BB_3	60.015	640	2.0	60.029	684	2.0	0.023	6.9	0.0085	0
GZ	60.020	2358	1.0	60.02	2419	1.0	0	2.6	0.0061	0
LM_1	60.007	640	2.0	60.026	752	2.0	0.03	17.5	0.0218	0
LM_2	60.001	640	2.0	60.02	752	2.0	0.03	17.5	0.0218	0
ODR	60.042	656	3.0	60.042	708	3.0	0	7.3	0.0026	0
OPi_1	60.070	504	2.96	60.057	604	3.0	0	19.8	0.01	1.35
OPi_2	60.070	504	3.0	60.06	604	3.0	0	19.8	0.01	0
RPi_1	60.040	1568	2.0	60.04	1614	2.0	0	2.9	0.0046	0
RPi_2	60.040	1585	2.0	60.04	1629	2.0	0	2.7	0.0044	0
RPi_3	60.030	1575	1.0	60.038	1630	1.0	0.013	3.5	0.0055	0
RPi_4	60.030	1566	1.0	60.041	1624	1.0	0.018	3.7	0.0058	0
RPi_5	60.040	1537	5.0	60.047	1599	5.0	0.01	4	0.012	0
LPT_1	60.001	2042	1.0	60.001	2154	1.0	0	5.5	0.0018	0
LPT_2	60.013	2125	1.0	60.015	2173	1.0	0.003	2.2	0.0008	0

classes within a CPS infrastructure. The implementation and further evaluation of S&F (Sections V and VI) consider the behavior of CPS devices while reacting to known challenges that are securely sent from a centralized secure server. S&F's signatures do not characterize the behavior of the devices while performing regular networking activities in the field. As explained before, the requests are signed, encrypted, and timed to prevent potentially compromised devices from uncovering the challenge. With the challenge, attackers would be able to modify the OS or kernel to generate a fake signature that could potentially bypass S&F's analysis, hiding the real malicious behavior of the compromised device. Also, signing the requests makes it harder for potential attacker to perform illegal request which may also reveal the specific device-class signatures. Although the design approach of the proposed fingerprinting approach could also support anomaly-based intrusion detection systems (IDSs) [8], S&F does not monitor the devices' behaviors while they are performing regular network operations. Thus, S&F does not consider specific attack vectors that may deviate the devices' behavior from their expected signature [8], [20]. Detecting and classifying specific anomalies coming from compromised devices is outside the scope of this work. However, we analyze S&F's performance under potentially compromised environments below.

B. S&F *on Compromised Environments*

We discuss on the performance of STOP-AND-FRISK on compromised devices. The main goal of S&F is to fingerprint CPS devices based on their behavior. Since we combine software and hardware characteristics of the devices into our analysis, there are several steps an attacker must go through to properly bypass S&F.

1) The Attacker Knows the Challenge: Since S&F is expected to run on a secure environment, the attacker must be able to break the encryption scheme and take control of the challenge. At this point, S&F relays on the secrecy of modern symmetric and asymmetric encryption schemes. Most modern microcontrollers incorporate low-power state-of-the-art encryption capabilities into their design [40]. Thus, it is nowadays feasible to perform long-key encryption on resource-limited devices, enabling the implementation of a secure S&F in practice.

2) The Attacker Controls the Device: Our threat model assumes that an attacker is able to introduce fake devices into a critical infrastructure. In this case, we should assume that the attacker has control over the device's OS and apps and can generate fake PCL and DPI sequences. However, the attacker still needs to infer the device signature which resides in a secure server, outside the edge devices. Under these circumstances, the attacker's only capability is to infer the exact behavior of the devices at both hardware and software levels to create a counterfeit signature that looks exactly like the ground-truth signature of the device class is being impersonated.

3) The Device Under Attack: An special case of the previous threat assumes that a legitimate device is compromised and the attacker is able to perform stealthy attacks. In this case, S&F would not be able to directly detect the attacks as the signatures only computes the device behavior while reacting to a specific challenge and not while the devices perform malicious activities. However, previous works have demonstrated that approaches similar to S&F can be used to detect stealthy attacks (e.g., stealing sensitive information), as the attacks would change the normal behavior of the devices [8]. S&F can be adapted to these scenarios if the right signature is collected. For this, the learning phase simply needs to study the behavior of the devices while their perform their expected operations over time, and not while reacting to a specific stimulus.

4) S&F *Under Attack:* It is possible for a legitimate device to get compromised. In this case, the attacker may be able to detect the S&F's requests and modify the PCL and DPI parameters, causing the device not to correctly react to the challenge. As a result, S&F would flag the device as unauthorized and reject the device, potentially causing a Denial-of-Service (DoS) situation. Although the DoS attack is the result of S&F's analysis, it is definitely positive to reject a compromised device from the network.

VIII. RELATED WORK

A. Device Fingerprinting

Device fingerprinting is an appealing research area that follows two main paths: device-class and device-host fingerprinting. In [41] the feasibility of large-scale host fingerprinting via motion sensors is analyzed with 90% accuracy. Other works use microscopic deviations in clock skews to identify specific devices [42]–[44]. However, these approaches are vulnerable to simple countermeasures and require the analysis of several network packets for accurate results. Authors in [45] use embedded acoustic devices (microphone and speakers) on smartphones to fingerprint individual devices. Even though they report accuracy values in the range of 98%, these results are only possible in close-range distances (0.1 meters). Similar research paths were followed in [46], [47] where frequency responses of devices' speakers are used to identify individual devices. The authors in [48] use RF-based fingerprinting to identify attacks to key-less entry systems. Also, the work in [49] uses the inertial measurement unit sensors found in iOS and Android devices to create globally unique fingerprints. Similarly, the work in [50] performs authentication of different hardware based on checksum extracted from micro-architectural implementation differences. However, this work limits its evaluation to simulation environments. The work in [51] analyzes hardware imperfections from 3D printers to identify unlawful 3D printed products. The recent impact of the Internet of Things (IoT) has motivated researchers to look into mechanisms to fingerprint IoT devices. One interesting approach uses a stimulation-response mechanism to identify specific devices via magnetic signals emitted from the CPU module [52]. Finally, Channel State Information (CSI) has also been proposed to fingerprint WLAN devices [53].

As for the identification of different classes of devices, in [54], the authors propose a passive *blackbox* technique for determining the type of access point (AP) connected to a network based on its behavior. In [19], [55], the authors use time as a baseline for device type fingerprinting. In this case, the proposed fingerprinting methods are mainly based on (1) the response time to network-based interactions (cross-layer fingerprinting) and (2) the response time to physical operations (physical fingerprinting). Although their results are promising, the first approach highly depends on configurable network attributes like the level of priority of TCP messages and ACK implementation. Further, the second proposed method also depends on the SCADA system configuration. In different works, passive device-class fingerprinting is proposed by using the timing distributions between network packets as the fingerprinting features [38], [56]. In similar approaches applied to domains other than CPS, researchers propose the analysis of network dynamics to infer IoT device classes [37], [57]–[60].

B. Function and System Calls

Several security approaches make use of system and function call analysis to regulate and monitor the behavior of specific applications [24], [61], [62]. For instance, researchers have proposed the use of system and function call analysis for the design of intrusion detection systems (IDS) [63], the identification of operating system functions [64], sandboxing [65], and the implementation of software portable packages. Also, some works have demonstrated that similar approaches are suitable for the classification of behavioral anomalies [66], [67]. Although these last works report high overhead introduced to the systems, other similar implementations are more lightweight [68].

TABLE V
S&F AND OTHER FINGERPRINTING TECHNIQUES

Fingerprinting Tool	Reference	Requires Extensive Analysis	Utilizes Network Dynamics	Analyzes Physical Metrics	Analyzes OS's Metrics	Yield High Overhead
Device Fingerprinting	[19], [38], [42], [54]	●	●	○	○	X
Behavioral Fingerprinting	[35], [36], [69], [70], [72]	●	○	●	○	X
S&F	–	○	○	●	●	○

● - Yes, ○ - No, and X - Not Discussed

C. Behavioral Analysis

The authors in [69] use OS and hardware features extracted from computing systems to fingerprint browser users. In [35], behavioral (i.e., network traffic's temporal features) and static features are combined to fingerprint mobile device apps. Other works report the use of the devices' behavior as a response to stimulant network packets [70], [71]. In spite of their positive results, these types of fingerprinting techniques also come with some limitations. For instance, the proposed approaches only apply for specific types of network protocols (e.g., transport layer protocols like UDP, and TCP), or they are vulnerable to network dynamics such as WiFi channel characteristics and traffic delay. Also, the behavior of API functions in computer systems has been proposed to fingerprint specific devices [72]. The work in [36] proposes the utilization of sensor behaviors from mobile devices to create fingerprints used to authenticate users.

D. Differences From Existing Works

In Table V, we further compare S&F against fingerprinting techniques discussed before based on specific design and implementation criteria. The existing works (1) require extensive analysis, (2) depend on network dynamics, (3) consider physical metrics, (4) consider OS/kernel metrics, or (5) yield high overhead. We selected these comparison features as we believe they directly impact the performance of the time-critical CPS infrastructure. For instance, as opposed to S&F, other fingerprinting techniques require extensive data analysis or depend on specific network metrics to achieve their results. Surprisingly, these solutions do not offer specific overhead performance analyses even though fingerprinting solutions that focus on the behavior of the network dynamics may impact the performance of the CPS infrastructure. Additionally, their effectiveness either depends on the network's configuration, the analysis of extensive amount of data, or the observation of fingerprinting features over long periods. S&F is different from other discussed solutions since it is host-based and device-centered. Also, it does not require traffic monitoring, the study of the interaction between CPS devices and other network equipment, nor the need to overcome inevitable errors or overhead (e.g., latency) that can be introduced from changes in network dynamics. Our framework implements a signature-based device type fingerprinting mechanism that studies the behavior and performance of the CPS devices at both hardware and kernel levels. S&F utilizes a challenge-response approach where the devices perform standard CPS functionalities and operations. Finally, our technique achieves excellent identification results while introducing very little overhead to the CPS devices at downtime.

IX. Conclusion

CPS critical infrastructure networks use different devices to collect data and monitor critical operations. However, these devices can be spoofed by attackers to get access to systems, steal information, or disrupt critical operations. Also, legitimate CPS devices may include unauthorized pieces of software and hardware that could degrade critical CPS tasks. In this paper, we presented Stop-and-Frisk (S&F), a novel and lightweight CPS signature-based fingerprinting framework used to identify CPS device classes based on their behavior. Specifically, our novel approach combined system and function call tracing techniques, signal processing, and hardware performance analysis to implement a secure challenge/response-based device-class identification solution. Moreover, we evaluated the efficacy of S&F on a realistic testbed that included different classes of CPS devices with different hardware, software resources, and configurations. Our extensive experimental results demonstrated that S&F achieves an excellent rate in the identification of CPS devices classes. Also, our analysis revealed that the use of the proposed framework does not yield a significant overhead on the CPS devices' computing resources.

Acknowledgment

The views expressed are those of the authors only, not of the funding agencies.

References

[1] H. Aksu, L. Babun, M. Conti, G. Tolomei, and A. S. Uluagac, "Advertising in the IoT era: Vision and challenges," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 138–144, Nov. 2018.

[2] A. I. Newaz, A. K. Sikder, M. A. Rahman, and A. S. Uluagac, "HealthGuard: A machine learning-based security framework for smart healthcare systems," in *Proc. 6th Int. Conf. Social Netw. Anal., Manage. Secur. (SNAMS)*, Oct. 2019, pp. 389–396.

[3] A. K. Sikder, L. Babun, H. Aksu, and A. S. Uluagac, "Aegis: A context-aware security framework for smart home systems," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, Dec. 2019, pp. 28–41.

[4] Z. B. Celik *et al.*, "Sensitive information tracking in commodity IoT," in *Proc. USENIX*, 2018, pp. 1687–1704.

[5] L. Babun, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "Real-time analysis of privacy-(un)aware IoT applications," in *Proc. PETS/PoPETS*, 2021, pp. 145–166.

[6] Z. B. Celik, P. McDaniel, G. Tan, L. Babun, and A. S. Uluagac, "Verifying Internet of Things safety and security in physical spaces," *IEEE Secur. Privacy*, vol. 17, no. 5, pp. 30–37, Sep. 2019.

[7] D. van Opstal, U.S. Resilience Project. (Sep. 2012). *Supply Chain Solutions for Smart Grid Security: Building on Business Best Practices*. [Online]. Available: http://usresilienceproject.org/wp-content/uploads/2014/09/report-Supply%_Chain_Solutions_for_Smart_Grid_Security.pdf

[8] L. Babun, H. Aksu, and A. S. Uluagac, "A system-level behavioral detection framework for compromised CPS devices: Smart-grid case," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 2, pp. 1–28, Feb. 2020.

[9] C. Kaygusuz, L. Babun, H. Aksu, and A. S. Uluagac, "Detection of compromised smart grid devices with machine learning and convolution techniques," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[10] K. Denney, E. Erdin, L. Babun, M. Vai, and S. Uluagac, "USB-watch: A dynamic hardware-assisted USB threat detection framework," in *Proc. Int. Conf. Secur. Privacy Commun., Syst.*, 2019, pp. 126–146.

[11] L. P. Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac, "PoisonIvy: (In)secure practices of enterprise IoT systems in smart buildings," in *Proc. 7th ACM Int. Conf. Syst. Energy-Efficient Buildings, Cities, Transp.*, Nov. 2020, pp. 130–139.

[12] M. Corporation. (2020). *MITRE ATT&CK: Supply Chain Compromise*. Accessed: Apr. 17, 2020. [Online]. Available: https://resources.infosecinstitute.com/mitre-attck-supply-chain-comprom%ise/#gref

[13] J. Spooren, D. Preuveneers, and W. Joosen, "Mobile device fingerprinting considered harmful for risk-based authentication," in *Proc. 8th EuroSec*, 2015, pp. 1–6.

[14] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 94–104, 1st Quart., 2016.

[15] A. I. Newaz, A. K. Sikder, L. Babun, and A. S. Uluagac, "HEKA: A novel intrusion detection system for attacks to personal medical devices," in *Proc. IEEE CNS*, Jun. 2020, pp. 1–9.

[16] L. Babun, H. Aksu, and A. S. Uluagac, "Detection of counterfeit and compromised devices using system and function call tracing techniques," U.S. Patent 10 027 697, Jul. 17, 2018.

[17] L. Babun, H. Aksu, and A. S. Uluagac, "Method of resource-limited device and device class identification using system and function call tracing techniques, performance, and statistical analysis," U.S. Patent 10 242 193, Mar. 26, 2019.

[18] I. Graja, S. Kallel, N. Guermouche, and A. H. Kacem, "BPMN4CPS: A BPMN extension for modeling cyber-physical systems," in *Proc. IEEE 25th Int. Conf. Enabling Technol., Infrastruct. Collaborative Enterprises (WETICE)*, Jun. 2016, pp. 152–157.

[19] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah, "Who's in control of your control system? Device fingerprinting for cyber-physical systems," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–15.

[20] L. Babun, H. Aksu, and A. S. Uluagac, "Identifying counterfeit smart grid devices: A lightweight system level framework," in *Proc. IEEE ICC*, Paris, France, May 2017, pp. 1–6.

[21] U.S. Food & Drug Administration. (Nov. 2010). *Tissue Adhesive with Adjunct Wound Closure Device Intended for the Topical Approximation of Skin—Guidance for Industry and FDA Staff*. [Online]. Available: https://www.fda.gov/MedicalDevices/ucm233027.htm

[22] M. Sillgith. (Feb. 2016). *Open source library for IEC 61850: Release 0.9*. [Online]. Available: http://libiec61850.com/libiec61850/

[23] P. Pradhan, K. Nagananda, P. Venkitasubramaniam, S. Kishore, and R. S. Blum, "GPS spoofing attack characterization and detection in smart grids," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2016, pp. 391–395.

[24] J. Lopez, L. Babun, H. Aksu, and A. S. Uluagac, "A survey on function and system call hooking approaches," *J. Hardw. Syst. Secur.*, vol. 1, no. 2, pp. 114–136, Jun. 2017.

[25] W. Xiao, "Novel online algorithms for nonparametric correlations with application to analyze sensor data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 404–412.

[26] P. Misra, W. Hu, M. Yang, M. Duarte, and S. Jha, "Sparsity based efficient cross-correlation techniques in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 7, pp. 2037–2050, Jul. 2017.

[27] C. Kriger, S. Behardien, and J.-C. Retonda-Modiya, "A detailed analysis of the goose message structure in an IEC 61850 standard-based substation automation system," *Int. J. Comput. Commun. Control*, vol. 8, no. 5, pp. 708–721, 2013.

[28] D. B. Rawat and C. Bajracharya, "Cyber security for smart grid systems: Status, challenges and perspectives," in *Proc. SoutheastCon*, Apr. 2015, pp. 1–6.

[29] J. Myers, L. Babun, E. Yao, S. Helble, and P. Allen, "MAD-IoT: Memory anomaly detection for the Internet of Things," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.

[30] KVM. (2016). *Main Page—KVM*. Accessed: Apr. 17, 2020. [Online]. Available: https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792

[31] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–36, Feb. 2019.

[32] S. M. Ross, *Probability Models for Computer Science*, 1st ed. Orlando, FL, USA: Academic, 2001.

[33] P. J. Guo and D. Engler, "CDE: Using system call interposition to automatically create portable software packages," in *Proc. USENIXATC*, 2011, p. 21.

[34] R. E Faith and M. Kerrisk. *The Linux Man-Pages Project: Ptrace*. [Online]. Available: http://man7.org/linux/man-pages/man2/ptrace.2.html

[35] T. van Ede *et al.*, "FlowPrint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–18.

[36] C. Wu, K. He, J. Chen, Z. Zhao, and R. Du, "Liveness is not enough: Enhancing fingerprint authentication with Behavioral biometrics to defeat puppet attacks," in *Proc. USENIX*, 2020, pp. 2219–2236.

[37] L. Babun, H. Aksu, L. Ryan, K. Akkaya, E. S. Bentley, and A. S. Uluagac, "Z-IoT: Passive device-class fingerprinting of ZigBee and Z-Wave IoT devices," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.

[38] A. S. Uluagac, S. V. Radhakrishnan, C. Corbett, A. Baca, and R. Beyah, "A passive technique for fingerprinting wireless devices with wired-side observations," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2013, pp. 305–313.

[39] H. Aksu, A. S. Uluagac, and E. Bentley, "Identification of wearable devices with Bluetooth," *IEEE Trans. Sustain. Comput.*, early access, Feb. 21, 2019, doi: 10.1109/TSUSC.2018.2808455.

[40] A. S. Security. (2020). *Layered Security for the Next One Trillion Devices*. Accessed: Oct. 17, 2020. [Online]. Available: https://www.arm.com/solutions/security

[41] A. Das, N. Borisov, E. Chou, and M. H. Mughees, "Smartphone fingerprinting via motion sensors: Analyzing feasibility at large-scale and studying real usage patterns," 2016, *arXiv:1605.08763*. [Online]. Available: http://arxiv.org/abs/1605.08763

[42] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Depend. Sec. Comput.*, vol. 2, no. 2, pp. 93–108, Feb. 2005.

[43] F. Lanze, A. Panchenko, B. Braatz, and A. Zinnen, "Clock skew based remote device fingerprinting demystified," in *IEEE GLOBECOM*, Dec. 2012, pp. 813–819.

[44] S. Sharma, A. Hussain, and H. Saran, "Experience with heterogenous clock-skew based device fingerprinting," in *Proc. LASER*, 2012, pp. 9–18.

[45] A. Das, N. Borisov, and M. Caesar, "Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components," in *Proc. CCS*, 2014, pp. 441–452.

[46] Z. Zhou, W. Diao, X. Liu, and K. Zhang, "Acoustic fingerprinting revisited: Generate stable device ID stealthily with inaudible sound," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 429–440.

[47] T. Park, S. Beack, and T. Lee, "A noise robust audio fingerprint extraction technique for mobile devices using gradient histograms," in *Proc. IEEE 5th Int. Conf. Consum. Electron. Berlin (ICCE-Berlin)*, Sep. 2015, pp. 287–290.

[48] K. Joo, W. Choi, and D. H. Lee, "Hold the door! Fingerprinting your car key to prevent keyless entry car theft," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–37.

[49] J. Zhang, A. R. Beresford, and I. Sheret, "SensorID: Sensor calibration fingerprinting for smartphones," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 638–655.

[50] D. Y. Deng, A. H. Chan, and G. E. Suh, "Hardware authentication leveraging performance limits in detailed simulations and emulations," in *Proc. 46th Annu. Design Autom. Conf. ZZZ DAC*, 2009, pp. 682–687.

[51] Z. Li, A. S. Rathore, C. Song, S. Wei, Y. Wang, and W. Xu, "PrinTracker: Fingerprinting 3D printers using commodity scanners," in *Proc. CCS*, 2018, pp. 1306–1323.

[52] Y. Cheng, X. Ji, J. Zhang, W. Xu, and Y.-C. Chen, "DeMiCPU: Device fingerprinting with magnetic signals radiated by CPU," in *Proc. CCS*, 2019, pp. 1149–1170.

[53] F. Adamsky, T. Retunskaia, S. Schiffner, C. Köbel, and T. Engel, "WLAN device fingerprinting using channel state information (CSI)," in *Proc. WiSec*, 2018, pp. 277–278.

[54] K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2010, pp. 383–392.

[55] Q. Gu, D. Formby, S. Ji, H. Cam, and R. Beyah, "Fingerprinting for cyber-physical system security: Device physics matters too," *IEEE Secur. Privacy*, vol. 16, no. 5, pp. 49–59, Sep. 2018.

[56] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "GTID: A technique for physical device and device type fingerprinting," *IEEE Trans. Depend. Sec. Comput.*, vol. 12, no. 5, pp. 519–532, Sep. 2015.

[57] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "DEFT: A distributed IoT fingerprinting technique," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 940–952, Feb. 2019.

[58] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2177–2184.

[59] T. Duc Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," 2018, *arXiv:1804.07474*. [Online]. Available: http://arxiv.org/abs/1804.07474

[60] A. K. Dalai and S. K. Jena, "WDTF: A technique for wireless device type fingerprinting," *Wir. Pers. Commun.*, vol. 97, pp. 1911–1928, Nov. 2017.

[61] T. Garfinkel, "Traps and pitfalls: Practical problems in system call interposition based security tools," in *Proc. NDSS*, 2003, pp. 1–14.

[62] R. B. Blunden, *The Rookit Arsenal: Escape and Evasion in the Dark Corners of the System*, 2nd ed. J. B. Learning, Ed. Burlington, MA, USA: Cathleen Sether, 2013.

[63] K. Jain and R. Sekar, "User-level infrastructure for system call interposition: A platform for intrusion detection and confinement," in *Proc. NDSS*, 1999, pp. 1–16.

[64] G. Hunt and D. Brubacher, "Detours: Binary interception of Win32 functions," in *Proc. WINSYM*, vol. 3, 1999, pp. 1–10.

[65] T. Kim and N. Zeldovich, "Practical and effective sandboxing for non-root users," in *Proc. USENIX ATC*, 2013, pp. 139–144.

[66] H. H. Feng, O. M. Kolesnikov, P. Fogla, W. Lee, and W. Gong, "Anomaly detection using call stack information," in *Proc. 19th Int. Conf. Data Eng.*, May 2003, pp. 62–75.

[67] N. A. Milea, S. C. Khoo, D. Lo, and C. Pop, "NORT: Runtime anomaly-based monitoring of malicious Behavior for windows," in *Proc. Int. Conf. Runtime Verification*, 2011, pp. 115–130.

[68] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni, "A fast automaton-based method for detecting anomalous program Behaviors," in *Proc. IEEE Symp. Secur. Privacy. S&P*, May 2001, pp. 144–155.

[69] Y. Cao, S. Li, and E. Wijmans, "(Cross-)browser fingerprinting via OS and hardware level features," in *Proc. NDSS*, 2017, pp. 1–15.

[70] C. Neumann, O. Heen, and S. Onno, "An empirical study of passive 802.11 device fingerprinting," in *Proc. 32nd ICDC Workshops*, Jun. 2012, pp. 593–602.

[71] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, "Active Behavioral fingerprinting of wireless devices," in *Proc. WiSec*, 2008, pp. 56–61.

[72] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Clock around the clock: Time-based device fingerprinting," in *Proc. CCS*, 2018, pp. 1502–1514.

Leonardo Babun (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Florida International University in 2020. His research interests include cyber-physical systems (CPS) and the Internet of Things (IoT) security and privacy. He is currently a CyberCorps Scholarship for Service Alumnus and a member of the Cyber-Physical Systems Security Lab (CSL), Florida International University.

Hidayet Aksu received the Ph.D. degree from Bilkent University in 2014. His research interests include security for cyber-physical systems, the Internet of Things, security for critical infrastructure networks, security analytics, big data analytics, distributed computing, wireless networks, wireless ad hoc and sensor networks, localization, and P2P networks.

A. Selcuk Uluagac received the M.S. and Ph.D. degrees from the Georgia Institute of Technology. He leads the Cyber-Physical Systems Security Lab, Florida International University, focusing on security and privacy of the Internet of Things and cyber-physical systems. He received the U.S. National Science Foundation CAREER Award and the U.S. Air Force Office of Sponsored Research's Summer Faculty Fellowship. He currently serves on the editorial boards for the IEEE TRANSACTIONS ON MOBILE COMPUTING, *Computer Networks* (Elsevier), and the IEEE COMMUNICATIONS AND SURVEYS AND TUTORIALS.