IVYCIDE: Smart Intrusion Detection System Against E-IoT Driver Threats

Luis Puche Rondon, Leonardo Babun, *Member, IEEE*, Ahmet Aris, Kemal Akkaya, *Senior Member, IEEE*, and A. Selcuk Uluagac

***Abstract*—The rise of Internet of Things (IoT) devices has led to the proliferation of smart environments worldwide. Although commodity IoT devices are employed by ordinary end users, complex environments, such as smart buildings, government, or private offices, or conference rooms require customized and highly reliable IoT solutions. Enterprise IoT (E-IoT) connect such environments to the Internet and are professionally managed solutions usually offered by dedicated vendors As E-IoT systems require specialized training, closed-source software, and proprietary equipment to deploy. In effect, E-IoT systems present an unprecedented, under-researched, and unexplored threat vector for an attacker. In this work, we focus on E-IoT drivers, software modules used to integrate devices into E-IoT systems, as an attack mechanism. We first present PoisonIvy, a series of generalized proof-of-concept attacks used to demonstrate that an attacker can use a malicious driver to perform denial-of-service attacks, gain remote control, and abuse E-IoT system resources. To defend against E-IoT driver-based threats, we introduce IVYCIDE, a novel intrusion detection system used to detect unexpected E-IoT network traffic from an E-IoT system. IVYCIDE operates as a passive monitoring system within an E-IoT system using machine learning and signature-based classification to detect POISONIVY attacks. We evaluated the performance of IVYCIDE in a realistic E-IoT deployment. Our detailed evaluation results show that IVYCIDE achieves an average accuracy of 97% in classifying the type of POISONIVY attack and operates without modifications or operational overhead to the existing E-IoT systems.**

***Index Terms*—Cyber attacks, enterprise IoT (E-IoT) systems, intrusion detection.**

I. INTRODUCTION

THE INTRODUCTION of modern commodity Internet of Things (IoT) devices has changed the everyday lives of users with the deployment of millions of smart environments (e.g., smart buildings, offices, homes, etc.) worldwide [2]. While many IoT systems are easily installed by average end users via Do-it-Yourself (DIY) applications, Enterprise IoT (E-IoT) systems exist as an automation solution for professional settings. As such, E-IoT systems are used exclusively for applications, such as smart buildings, luxury smart homes, expensive yachts, classrooms, meeting rooms, government offices, and business establishments. In these professional settings, proprietary E-IoT systems (e.g., Crestron, Control4, and Savant) introduce robust, reliable, and custom solutions catered to meet an enterprise client's needs. As such, E-IoT systems require professional installation and specialized training to deploy. Additionally, maintenance, service, and upgrades of E-IoT systems are handled by specialized integrators.

Although many consumer-grade commodity IoT systems are well understood due to their mainstream popularity, very little security research exists on E-IoT systems' design, development, verification processes, and vulnerabilities. The lack of research on these systems has led many users to overlook E-IoT systems as possible attack vectors and assume that these systems are secure. With many of these professional systems present in high-profile locations, evaluating threats for E-IoT systems should be of utmost importance. In this article, we systematically explore E-IoT system vulnerabilities and insecure development practices, specifically, the usage of drivers as an attack mechanism. To demonstrate that malicious actors can easily attack E-IoT systems, we introduce POISONIVY, a collection of novel attacks that leverages E-IoT system vulnerabilities to an attacker's benefit. Specifically, with POISONIVY, we demonstrate that an attacker can use malicious E-IoT drivers to remotely: 1) perform Denial-of-Service (DoS) attacks on E-IoT systems; 2) assume control of E-IoT systems as an effective botnet; and 3) use E-IoT system resources to perform illicit activities (e.g., bitcoin mining and distributed password cracking).

Securing an E-IoT system against POISONIVY attacks presents distinct challenges as E-IoT systems are closed-source. E-IoT systems cannot be modified without the help of the vendor to monitor the running processes or API/system calls to detect the activities of the malicious drivers in POISONIVY. Therefore, a passive network monitoring solution remains as an applicable methodology to detect such driver-based attacks based on the network traffic they create. Hence, to defend against POISONIVY-style threats we present IVYCIDE; a passive network monitoring-based intrusion detection system (IDS) designed to protect E-IoT deployments against POISONIVY-like threats using machine learning (ML) and signature-based classification. As POISONIVY attacks rely

Manuscript received 25 March 2022; revised 18 June 2022; accepted 25 July 2022. Date of publication 4 August 2022; date of current version 9 May 2023. This work was supported in part by the U.S. National Science Foundation under Award NSF-CAREER CNS-1453647, Award NSF-1663051, and Award NSF-2219920; and in part by the Microsoft Research Grant. *(Corresponding author: Luis Puche Rondon.)*

Luis Puche Rondon, Leonardo Babun, Ahmet Aris, and A. Selcuk Uluagac are with the Cyber-Physical Systems Security Lab, Florida International University, Miami, FL 33199 USA (e-mail: lpuch002@fiu.edu; lbabu002@fiu.edu; aaris@fiu.edu; suluagac@fiu.edu).

Kemal Akkaya is with the Advanced Wireless and Security Lab, Florida International University, Miami, FL 33199 USA (e-mail: kakkaya@fiu.edu).

Digital Object Identifier 10.1109/JIOT.2022.3196282

on network communication to communicate with the attacker or apply the attack, IVYCIDE operates as a standalone framework for E-IoT systems, passively monitoring network traffic for unexpected and malicious behavior. IVYCIDE first classifies individual incoming and outgoing network packets into four types of distinct behaviors caused by POISONIVY attacks using ML-based techniques. IVYCIDE then uses these individual network packets and signature-based classification to determine the type of POISONIVY attack occurring. To test IVYCIDE's performance against POISONIVY attacks, we conducted a set of evaluations in a realistic E-IoT testbed using real E-IoT devices. Our results show that IVYCIDE achieves an average accuracy of 97% and precision of 94% against POISONIVY-style attacks without any operational overhead or modification to the E-IoT system.

Contributions: The contributions of this work are as follows.

1) We demonstrate that E-IoT system drivers are a viable attack vector for smart buildings by introducing POISONIVY, a series of novel attacks against E-IoT systems.
2) We test and evaluate POISONIVY attacks in a real E-IoT system and leverage malicious drivers to cause undesired behavior in a smart building on behalf of a remote attacker.
3) We propose IVYCIDE, a novel IDS to protect E-IoT systems against driver-based threats. IVYCIDE monitors active E-IoT network traffic and detects unexpected network traffic generated by the malicious drivers of POISONIVY.
4) We evaluate IVYCIDE in a realistic E-IoT environment with a variety of E-IoT devices, achieving an overall accuracy of 97% and precision of 94%.

Organization: The remainder of this work is organized as follows: In Section II, we provide background information on E-IoT systems. Section III introduces the problem scope and threat model of the POISONIVY attacks. In Section IV, we cover the architecture, attack implementation, and evaluation of the POISONIVY attacks. Section V covers definitions and the IVYCIDE architecture. In Section VI, we cover the IVYCIDE defense system implementation. The effectiveness of IVYCIDE is covered in Section VII. Section VIII discusses the related work. Finally, we conclude this article in Section IX.

II. BACKGROUND

In this section, we introduce some concepts about E-IoT systems.

A. E-IoT Systems

E-IoT systems are specialized closed-source smart systems with unique design and deployment practices distinguishing them from off-the-shelf IoT solutions [44]. As all installations are custom, *an integrator*, a specialized programmer and installer, configures and integrates all devices with an E-IoT system. An integrator is hired to perform the physical installation, device configuration, testing, and technical support of the E-IoT system [24], [25]. There are many different use-cases where E-IoT is the best solution for automation and integration of multiple devices. Automation may be done in single-room systems (e.g., a theater, a conference room) or multiroom systems. Fig. 1 highlights applications of E-IoT in smart buildings. For instance, a smart office may be automated with CCTV systems, lighting control systems, and access control components with an E-IoT system. As such, E-IoT systems are customized for each specific application and deployment. We highlight some E-IoT use-cases on smart buildings, where these use cases can work together under a single E-IoT system. Thus, if the E-IoT system is compromised, the integrated devices may also be compromised.

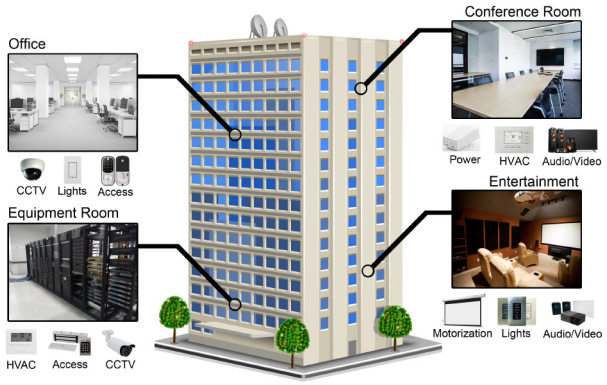

Fig. 1. Applications of E-IoT systems.

Lighting Control: Any control of physical lighting or electrical loads by an E-IoT system (e.g., lights, fans, outlets). E-IoT systems may be used in this use-case to schedule light events, program independent keypads, and allow remote control of lighting functions. E-IoT allows users to control their lights remotely, schedule light events (e.g., wake up, turn outdoor lights on sundown, and trigger light-based events from other devices.

Security and Safety: E-IoT systems are often integrated to control security components. This integration grants authorized users the ability to control security aspects of a location (e.g., CCTV systems, access control systems, motion sensors, fire alarms, and security alarm systems). Thus, E-IoT systems allow for remote access, control, and camera activation based on motion sensor triggers. E-IoT allows users to integrate other components, such as lights with security systems. For example, an E-IoT system may start flashing lights when the alarm system is triggered.

Advanced Media Control: The control and management of media and audio/video (A/V) components with E-IoT systems (e.g., projectors, televisions, video distributions, HDMI networks, and audio matrix management). E-IoT systems manage complex audio/video distribution networks from a single interface through audio/video zones, audio switchers, video switchers, and amplifiers. With the complexity of many A/V systems, E-IoT is a reliable method of control through a single user interface.

B. Architecture of E-IoT Systems

E-IoT systems follow a centralized system design. We refer to Fig. 2 which shows the generalized design of an E-IoT

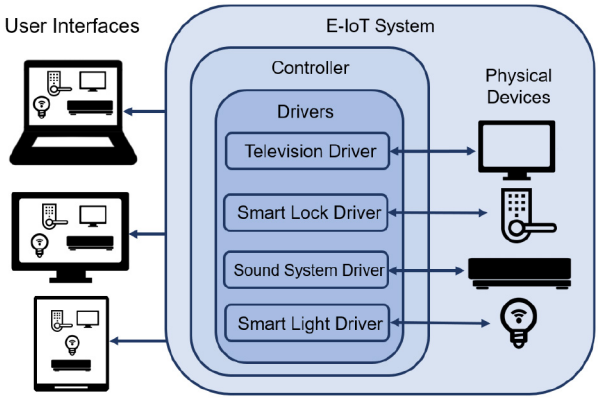

Fig. 2. E-IoT system with four different control drivers, controller, and user interfaces. Individual devices are controlled through the user interfaces after being integrated.

system. These systems consist of several modules, the user interfaces, a controller, drivers, and physical devices. *Physical devices* are devices, such as televisions or smart lights which are integrated into an E-IoT system. E-IoT systems usually integrate physical devices without cloud-based access [24]. To accomplish this, E-IoT systems use a central *controller*, a dedicated processing device that contains the main logic, communication behavior, and configuration of an E-IoT system. The controller stores *drivers* locally, which contain all the necessary information to integrate a specific physical device or service into the E-IoT system. Each physical device requires a driver of its own to be integrated to a smart system. For instance, to integrate a smart door lock, an E-IoT driver for that specific door lock must be added to the controller. After a device is integrated, that device becomes available to the user through any of the smart system's user interfaces (e.g., laptop app, phone/tablet app, and television on-screen display). In contrast to traditional system drivers, E-IoT drivers function on top of the proprietary E-IoT operating system. The implementation may also be different between E-IoT systems.

C. E-IoT Drivers

One of the primary components of many E-IoT systems is the inclusion of *drivers* which may have different names depending on the manufacturer (e.g., Crestron modules and Control4 drivers). Drivers provide all the information and software modules necessary to integrate a device into a centralized E-IoT system. For instance, to integrate a Sony television into an E-IoT system, the controller must know what the protocol of communication is (e.g., IR, serial, IP, ZigBee, and *z*-wave), the physical inputs (HDMI ports, analog ports), and the vendor-specific proprietary commands to interface with the device. Drivers are not limited to simply integrating physical devices. They also integrate services such as Weatherbug to add more functionality to an existing system [21], [22].

Drivers are inserted and configured during programming or maintenance stages of a smart environment by the integrator. Integrators may obtain drivers in three different ways: 1) they may get drivers directly from the E-IoT system software (preloaded drivers); 2) directly from a catalog hosted by the manufacturer of the E-IoT system devices; or 3) download from a third-party site in the Internet (from a third-party vendor or a developer). Vendors of E-IoT systems often validate drivers distributed in their platforms for functionality such as Control4's certified drivers [21]. However, with millions of different devices to be integrated, certifying every driver is not possible. In effect, integrators may be forced to use third-party drivers for their installations if no drivers are available for their specific solutions from the vendor or manufacturer. In this work, we focus on unverified drivers, or drivers available on third-party sites that have not been checked for malicious content.

Unverified Drivers: Integrators may opt for unverified, third-party drivers due to several reasons.

1) *Driver Availability:* Verified drivers may not always be available to an integrator. Therefore, the only recourse to integrate a third-party device to an E-IoT system may be with an unverified driver from an untrusted source. Additionally, to integrate less-known devices, the driver has to be made by the manufacturer, who may be untrusted and their code closed-source. For instance, integrator forums offer a floury of unverified drivers for projectors, televisions, and other devices [16]. Additionally, many vendors do not offer E-IoT drivers for their devices, leading to third-party developers to offer their own drivers.
2) *Cost:* Developers may charge for verified drivers (e.g., Atlona HDMI Switcher drivers for U.S. $110), which, in turn, has to be paid by the integrator and end user [28]. Integrators may be tempted to use free unverified drivers available on forums and online storefronts. Further, while paid drivers may be made by trusted developers, they are not necessarily verified by the E-IoT vendor.
3) *Compatibility:* Devices may change commands and specifications when their firmware is updated [38]. As such, verified drivers need to be updated to remain compatible with the latest models and firmware. To get a system running quickly after an update, an integrator may use an unverified driver that claims to run perfectly with a newer firmware version of the device when a verified driver is not available.
4) *Phishing:* It is possible that an integrator may install an untrusted driver through a phishing link offering a "driver update" or a tampered vendor website. It is possible to receive drivers through email attachments, impersonating a trusted vendor.

III. Problem Scope and Threat Model

This section presents the problem scope and threat model for PoisonIvy attacks.

A. Problem Scope

This work assumes the existence of an E-IoT system installed in a smart building. Indeed, such E-IoT systems have experienced a rapid increase in popularity in smart buildings, luxury smart homes, expensive yachts, classrooms, meeting

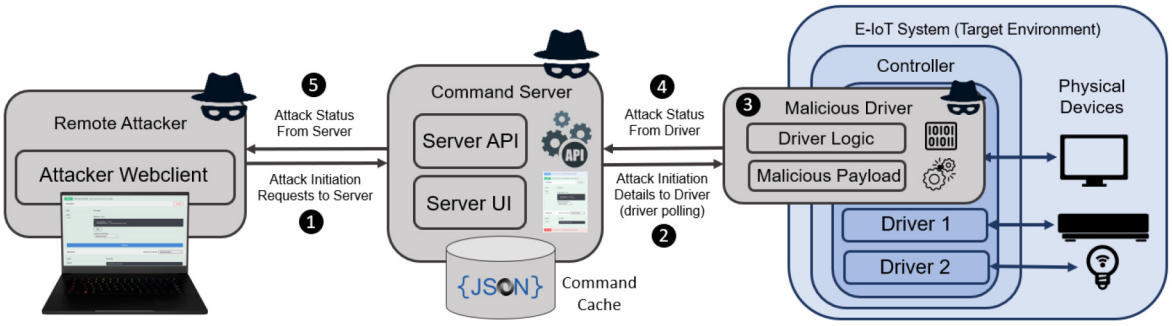

Fig. 3. General end-to-end implementation for POISONIVY-based attacks. Attack-related components are highlighted in gray, E-IoT system components are in blue.

rooms, government offices, and business establishments. The E-IoT system's controller is connected to a network and the Internet. The attacker, named Mallory, is a malicious actor with knowledge of E-IoT systems and their weaknesses. In this scenario, Mallory develops a malicious driver for a popular device and advertises the driver through user boards such as online forums [16] to integrators. Mallory also creates fake accounts to give good reviews on the driver and mislead integrators. The driver advertised is not available by the manufacturer or through verified drivers, making Mallory's driver the only way to integrate a particular device into an E-IoT system. With this malicious driver, Mallory assumes the control of E-IoT system controllers and uses her machine to execute remote attacks.

Additionally, we assume that an integrator uses Mallory's unverified malicious driver for the E-IoT system deployment, introducing it into the system without issues as there are no security mechanisms in place. These assumptions are realistic as online drivers from third-party sites are not verified, and smart systems require Internet connectivity for many of their services (e.g., remote access, music streaming, movies) [33]. Anyone can upload a driver to public forums easily. As integrators may download unverified drivers from any website, an attacker can create an attractive driver for integrators to download and install in their systems. For instance, unverified drivers may be offered at a third-party website that advertises them. In our current scenario, Mallory compromises E-IoT system devices indirectly through the use of a downloaded unverified malicious driver.

B. Threat Model

In our work, we consider the following powerful adversaries as part of the threat model.

Threat 1 (DoS): This threat considers DoS attacks where Mallory disrupts the availability of an E-IoT system through the use of a malicious driver.

Threat 2 (Remote Control): This threat considers a case where Mallory assumes the control of E-IoT system devices to execute attacks (e.g., DoS) against remote targets such as webservers.

Threat 3 (Malicious System Resource Farming): In this threat, Mallory uses local system resources in a compromised device to perform unauthorized processor-intensive actions to her benefit (e.g., cryptocurrency mining [45], password cracking [4]).

IV. POISONIVY

In this section, we present an overview of the POISONIVY attacks. The complete details of this work can be found in our previous publication [42]. Fig. 3 depicts the end-to-end implementation of POISONIVY attacks. Such implementation involves the interaction of four modules: 1) *remote attacker*; 2) *command server*; 3) *malicious driver*; and 4) *target environment*.

A. POISONIVY Overview

In this architecture, the attacks begins with a remote attacker (e.g., Mallory) initiating an attack with a webclient such as a laptop by communicating with the command server ❶. The command server grants Mallory an intermediary point of communication between her device and infected controllers, and includes three components: 1) the *server API*; 2) *server UI*; and 3) the *command cache*. The server API represents the primary endpoints (e.g., REST Architecture) of the server, which can be requested by Mallory or the malicious driver. Mallory uses the server UI component executed by the webclient, which grants her a visual interface, to initiate attacks and view the attack's status. As the last component of the command server, the command cache stores attack initiation requests fetched by malicious drivers through the server API endpoints. Once the command server receives initiation requests from Mallory, the malicious driver can now query the command server for new attack details ❷. The Malicious driver is the core of POISONIVY attacks and contains the *driver logic* and the *malicious payload*. As such, the driver logic controls a smart device in an expected manner, which allows the driver to appear as a benign driver. In contrast, the malicious payload contains the attacker's malicious code for the execution of the attacks ❸. Finally, the target environment contains the smart system's controller and the drivers of the smart environment. Mallory takes control of the target environment's functions through the malicious driver. As attacks complete, the malicious driver sends back the attack status to the command server ❹. The attack status includes feedback to the attacker from the driver, such as hashing results, errors, or success codes, and can be then queried by Mallory from the command server ❺.

B. POISONIVY *Implementation*

To implement the POISONIVY attacks in a realistic manner, we created a malicious television E-IoT driver and an E-IoT system testbed with real Control4 devices. Control4 was selected as it is one of the most popular E-IoT systems available in the market, named a leading brand in E-IoT for five years in a row [23]. The testbed includes vendor-specific devices and is configured to function as a small E-IoT system. We utilized the Driver Editor, a tool available for the development of drivers in Control4 [15]. Additionally, we used LUA, an open-source programming language which is the core development platform of Control4 drivers [51]. We configured the E-IoT system using Control4's Composer 2.10.6 and with an EA-1 as the main controller. To grant Internet access to the devices included in the testbed, we configured a network with the TP-Link TL-WR841N Router. We verified the running version of LUA in Control4 devices as LUA 5.1 programmatically (executing a script which returned the running LUA version). To implement POISONIVY realistically, we created a command and control webserver with a RESTful API in JAX-RS hosted in Amazon AWS. A Swagger-based Web interface was implemented for the command server. Furthermore, we highlight that these examples noted are generalized, proof-of-concept attacks which were designed to prove the capabilities of POISONIVY. The capabilities proved with these attack concepts may be used to create more numerous and specific attack cases.

C. POISONIVY *Attacks*

In this section, we realized the POISONIVY attacks. More details on the realization and evaluation can be found on our prior publication [42].

Attack 1 (DoS): This attack was developed to demonstrate that Threat 1 (DoS) is possible through POISONIVY. This attack implements a DoS condition in a local system through memory exhaustion with a deliberate memory leak in the driver and an iterating loop. Using this attack, the E-IoT system was rendered inoperable within five seconds of activation. As such, any function tied to the E-IoT system was unreachable.

Attack 2 (Remote Control): The remote control attack served as a way to demonstrate the feasibility of Threat 2 (Remote Control) using POISONIVY. In this attack, the infected controller received a remote request to target a specific webhost "www.pucherondon.com." The attack was successful, the controller began repeated requests on the target webhost upon initiation. We note that with multiple infected controllers, it may be possible to negatively impact a target webpage. As such, this generalized attack demonstrates the viability of POISONIVY being for a variety of attacks, such as DoS or remote code execution.

Attack 3 (Malicious Resource Farming): The resource farming attack was developed to demonstrate that Threat 3 (Malicious Resource Farming) is possible through POISONIVY for purposes such as bitcoin mining. For this sample attack, the required cryptographic operations were created in the driver. The attack was successful, the driver began to perform hashing operations upon the command server's request. As such, Attack 3, proves that an attacker can easily misuse E-IoT system resources, with the given example of cryptographic operations. Functions for other operations can be created and deployed for other types of malicious resource farming.

Summary and Findings: All of the proposed attacks were implemented successfully, the implications which could negatively impact E-IoT systems. In Attack 1, we demonstrate that an entire system can be rendered unusable at the command of an attacker. The attack presents a viable method of disabling access to security systems, gates, doors, or any other system which is integrated into an E-IoT system. For instance, if the gate access or panic button is controlled purely through an E-IoT system, a user will not be able to operate the gate access or a panic button while a DoS attack is active. Attack 2 is made possible due to the lack of limitations on connections to external websites and shows how an attacker can perform DDoS-type attacks on target webpages using multiple controllers. There have been documented cases of malware purposely accessing illegal websites to frame the system owners [37]. An attacker with a compromised E-IoT system may request illegal websites and frame the owners for illicit activity. These results (Attack 2) also imply that an attacker may also perform any other hardware-intensive actions such as password cracking. Attack 3 is possible due to a lack of restrictions in the LUA implementation and unfettered access to system resources. Further,more with processor-intensive operations, a compromised controller could also be used for cracking hashed passwords. An attacker with a list of passwords to crack could use the processing power of compromised controllers to attempt to reverse password hashes, a very similar operation to cryptocurrency mining. As POISONIVY-style attacks present a substantial negative impact on E-IoT systems, acknowledging these threats and finding solutions should be of utmost importance.

Generalizability of POISONIVY *Attacks:* As noted, these attacks only act as a generalized representation of what is possible with POISONIVY and, thus, the effects and impact of each attack concept may differ from each custom deployment and attack implementation. Even with this, POISONIVY attacks rely on some basic concepts that come with drivers. For Attack 1, the method of attack relies on memory exhaustion. Thus, any method that can cause memory exhaustion, independent of the E-IoT system or implementation will cause the same effect. Attack 2 and the general communication between the attacker and driver rely on the ability for drivers to perform API calls. This feature needs to be available for E-IoT controllers to communicate with API-controlled devices and Web services. As such, it is likely that many, if not all, E-IoT systems have some method for drivers to perform API calls. Finally, Attack 3 relies on the ability to perform mathematical operations. Thus, any driver with a scripting language that allows the mathematical operations for cryptographic function will be able to perform Attack 3 or similar.

Existing Defenses and Need for New Solutions: Research into defending IoT and E-IoT systems has been an active research topic, with several solutions presented to different problems [6], [14], [44]. For instance, software solutions such

as context-aware defense mechanisms have been proposed to defend IoT systems [47]. While some concepts are applicable, E-IoT suffers from limitations such as closed-source design. As such, API/system call hooking techniques employed by some defense mechanisms are not entirely viable. Other widely accepted defense strategies, such as encryption, are not a viable solution as the E-IoT systems can be compromised by an integrator willingly installing a driver. Solutions, such as signature-based detection, for known malicious drivers are a possibility, however, such frameworks rely on vendor implementation. As these frameworks do not currently exist for E-IoT, development would require substantial research and funding. Another solution is the manual code verification of software modules, such as drivers and apps. While verification poses an attractive solution, there are some drawbacks. For instance, each E-IoT manufacturer uses their own implementation of a "driver" [1], [21]. Thus, while implementing E-IoT-wide solutions through standardization is a possibility, it requires the collaboration of E-IoT vendors, software, and framework changes to individual E-IoT systems and deployment practices. The necessary collaboration would require the development of a secure driver standard and agreements made between all vendors. However, older E-IoT systems with older drivers would be still vulnerable as legacy and discontinued equipment cannot be updated. In summary, the specific limitations of E-IoT environments make existing defense solutions impractical or inadequate to defend E-IoT systems against PoisonIvy-style attacks. In effect, new defense solutions must be proposed considering the threats and the distinct challenges of E-IoT systems.

V. Ivycide Architecture

To address PoisonIvy attack threats, we introduce Ivycide, a passive network-based IDS, easily configurable to detect traffic anomalies in E-IoT controller network traffic.

A. Design Considerations and Challenges

In this section, we first include the distinct challenges of E-IoT systems that make it difficult to protect against PoisonIvy attacks and require a specialized solution such as Ivycide. The design considerations of the Ivycide framework are driven by these challenges.

Closed-Source E-IoT: E-IoT systems are very often closed-source that makes many accepted defense strategies very difficult without vendor cooperation. Furthermore, software for configuration is not available to researchers and consumers. As such, a defense strategy must be designed with closed systems in mind. In the case of Ivycide, mechanisms must be created using features available to integrators and consumers. Without special permissions, source code, API/system call hooking, performance analysis, and other features, a defense system is a notable challenge to outside developers. Thus, Ivycide is needed as current defense systems may not work with the limited access of E-IoT systems and the lack of support from E-IoT vendors.

System-to-System Differences: There are many different E-IoT system vendors, each E-IoT system often with their type of configuration. As such, traffic will vary from E-IoT system to system, even if they are from the same vendor. For instance, a system that controls a single room (e.g., conference room and theater) will be vastly different in traffic than a large-scale system (e.g., whole home, smart office, and yacht complete integration). Furthermore, systems may differ from the services integrated. For instance, some users may opt for a fully offline system, while other users may request a system that integrates music services (e.g., Spotify, TuneIn, and Weatherbug). Since all systems are custom, a custom solution needs to be proposed for PoisonIvy attacks as existing solutions may not consider or be too costly for complex. A solution for E-IoT needs to be flexible, affordable, and needs to consider that systems may be updated and modified by the integrator.

E-IoT Traffic: In contrast to E-IoT devices, E-IoT controllers have some unique characteristics in E-IoT environments due to their role in E-IoT systems. First, the controller is the hub of all communication, as such, integrated devices (e.g., keypads, touchscreens, and televisions) communicate with the controller. Second, E-IoT controllers will often have audio and video interfaces, such as audio out, for streaming services and Internet radio (e.g., Spotify, Rhapsody, and TuneIn). Thus, in some systems, the E-IoT controller will handle the streaming service communication traffic. Finally, the E-IoT controller often communicates with the vendor's Web services and configuration software. In most cases, this means that the only way to modify the system (and drivers) for both benign or malicious purposes will be through the E-IoT controller and a network connection. As the E-IoT controller acts as the central communication hub, monitoring the network traffic of only the controller instead of all of the E-IoT devices can be useful to detect PoisonIvy attacks.

Constraints of E-IoT on the PoisonIvy Attacker: While PoisonIvy attacks demonstrate the capabilities of attackers using malicious drivers, there are limitations of E-IoT systems on PoisonIvy attacks that can be of use by a defense system such as Ivycide. First, a PoisonIvy attacker is limited on how they may communicate to the Internet. Namely, an attacker must rely on the driver's API to communicate remote servers. Second, an attacker must rely on this form of external communication for the core of Attacks 2 and 3 (Remote Control and Malicious Resource Farming, Section IV). Finally, traffic from a malicious driver originates from the controller. Thus, an attacker using the malicious drivers has only one device they can establish communication to external servers [e.g., Command-and-Control (CnC) server and target servers] and cannot execute attacks from other devices integrated in the E-IoT system. Knowing these limitations, a defense solution such as Ivycide can rely on network communication from the E-IoT controller to identify and detect malicious activities originating from a malicious driver.

B. E-IoT Devices, Drivers, and Expected Traffic

Modifications to E-IoT systems are not done frequently for several reasons. First, there are costs associated with contracting an integrator and purchasing new drivers after initial

installations. Additionally, if a device integrated into an E-IoT system needs a replacement (e.g., damages and upgrades), an integrator will often replace the device with a similar device to fulfill the same purpose. As such, it may not be necessary to retrain a learned model for an E-IoT system with similar replacements. Devices integrated into E-IoT have expected communication traffic dependent on the type of driver and device. In Table I, we show some examples of how device type defines the communication traffic of each integrated device. For instance, a display (e.g., television and projector) will communicate with the E-IoT system with information, such as power state, firmware version, and volume levels. Furthermore, depending on the E-IoT devices and their drivers, network traffic will be different. For instance, a driver for a device controlled through ZigBee by the E-IoT system should not create any additional IP network traffic. We highlight some examples of driver types as follows.

Driver Types: We highlight three types of drivers used for devices in E-IoT, and how each type of driver affects the E-IoT network traffic.

1) *Non-IP Drivers:* Drivers (e.g., ZigBee drivers) that do not use any IP network communication to control integrated devices. These drivers should not add any additional traffic to the E-IoT system. Since POISONIVY attacks require Internet connection, they will not function as this type of driver.
2) *IP Drivers:* Drivers that use IP network communication to connect to devices or services (e.g., IP TV driver, Spotify Drivers). These drivers will create network traffic relevant to the device or service. More information on expected network traffic is highlighted in Table I.
3) *Drivers With Remote Validation:* Drivers that require online validation, such as a licensed driver that must validate a license key with the developer of the driver. These drivers will have communication with a remote server.

C. Terminology

Expected Operation: We define the expected operation of an E-IoT system, as usage of an E-IoT system in a manner that is benign, such as selecting video sources, listening to music, and menu navigation. Activity occurring from malicious drivers is outside of expected operation.

Expected Traffic: We define benign traffic as any IP network communication which is caused by the expected operation of the E-IoT system.

Unexpected Network Behavior: Unexpected behavior occurs from unexpected network traffic due to active POISONIVY attacks in the E-IoT system. For the scope of this work, unexpected behavior is observed through the IP network communication.

D. IVYCIDE Overview

IVYCIDE is designed to protect E-IoT systems from POISONIVY-based threats. It aims to detect POISONIVY attacks via passive network monitoring and a two-step classification approach. In the first step of the classification, individual attack patterns are detected via a ML-based classifier, whereas in the second step, series of patterns are checked against attack signatures and the type of the attack is determined via a signature-based classifier.

TABLE I
EXAMPLES OF EXPECTED NETWORK TRAFFIC BY DEVICE TYPE

Device/Service Type	Expected Traffic
Displays	Power state, device state, volume state, version, media metadata.
A/V Equipment	Power state, device state, volume state, authentication.
User Interfaces	Device state, user input, external communication, authentication.
Sensors	Power state, device state, sensor data, user input.
Software Services	Media metadata, volume state, external communication, authentication.
Lighting Control	Power state, device state, light levels, user input.
Motorization	Power state, device state.

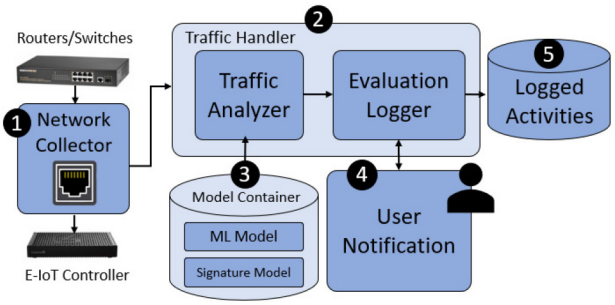

Fig. 4. Architecture of IVYCIDE, modules numbered.

The proposed IVYCIDE architecture is composed of five different modules as seen in Fig. 4. The first module is the *Network Collector* which captures network traffic incoming and outgoing from the E-IoT controller ❶. Furthermore, the Network Collector preprocesses E-IoT network traffic and forwards it to the Traffic Handler. The *Traffic Handler* evaluates incoming traffic and logs suspicious network activity using two submodules: the *Traffic Analyzer* and *Evaluation Logger* ❷. The Traffic Analyzer submodule is used as the first step, performing ML-based classification of individual E-IoT network traffic packets. These packets are classified into the four types of behaviors: 1) benign; 2) UER; 3) CnC; and 4) activation. As POISONIVY attacks are composed of a series of such behaviors, as the second step, the Traffic Analyzer applies a signature-based classification on a set of classified packets within a time window to determine the type of attack occurring. The Evaluation Logger submodule is then used to forward suspicious network packets and analysis results to the user notification and logged activities modules. The *Model Container* stores the ML model and Signature Model used by IVYCIDE's traffic analyzer ❸. The *User Notification* module is used to alert and notify the user on warnings and suspicious activities ❹. Finally, the *Logged Activities* module stores all the suspicious packets and classification results from the Traffic Handler ❺. This logged information may be queried later for reference, or further analysis.

E. Network Collector

The Network Collector allows IVYCIDE to passively collect incoming and outgoing traffic to the E-IoT controller. As such, Network Collector only captures TCP/IP network

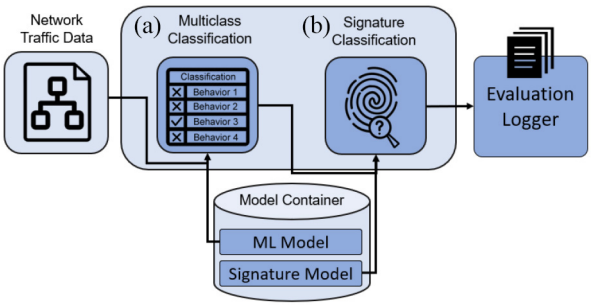

Fig. 5. IVYCIDE classification process.

traffic relevant to the E-IoT controller. Additionally, this capture is passive, as packet manipulation of E-IoT traffic may cause undesired operation for the E-IoT system. Furthermore, the captured traffic data that is irrelevant to IVYCIDE is then filtered out (e.g., internal LAN communication). Relevant packets to IVYCIDE (e.g., communication from controller to external servers) are parsed and features are extracted by the Network Collector for further processing. Filtered and formatted network data coming out of the Network Collector includes all packet information and additional attributes necessary for IVYCIDE training and classification (e.g., timestamp, data length, TCP or UDP flags).

F. Traffic Handler

The Traffic Handler acts as the classification stage for IVYCIDE and is composed of two submodules; the Traffic Analyzer and the Evaluation Logger.

1) Traffic Analyzer: The Traffic Analyzer is one of the core components of IVYCIDE and performs the ML multiclass classification of incoming/outgoing data to the controller. Additionally, the Traffic Analyzer performs signature-based classification using a series of attack patterns/behaviors to determine the type of the attack. This two-stage process is required since the type of the POISONIVY attacks cannot be identified from a single malicious packet or a single attack behavior. Furthermore, classifying behaviors per packet can yield to more flexibility and types of signatures for future attacks. We refer to Fig. 5, for the ML and signature-based classification processes employed by IVYCIDE. The first step of the Traffic Analyzer is the Multiclass Classifier. In this step, IVYCIDE attempts to classify network traffic as benign or as different types of unexpected behaviors/patterns (UER, activation, malicious CnC). Once network traffic is classified with ML, signature-based classification can take place. However, if a packet is classified as benign, no further classification is needed. For signature-based classification, IVYCIDE follows a set of rules and attempts to determine the type of attack that occurred depending on the unexpected activity observed within a set timeframe. The resulting classification and timestamps are then forwarded to the Evaluation Logger and user notification modules.

Multiclass Classifier: The Traffic Analyzer uses ML classification to infer the type of network activity occurring with the E-IoT controller. As IVYCIDE is designed to be flexible and better fit the heterogeneous nature of E-IoT systems, different ML algorithms and models may be used for better accuracy. For IVYCIDE, we defined four distinct types of behaviors for E-IoT systems and POISONIVY attacks. Specifically, attacks can be identified by a combination of these behaviors. While we define four distinct types of behaviors associated with E-IoT systems and attack behaviors, more types of behaviors may be learned and added with newer threats. A more detailed description of behaviors classified during this stage are highlighted as follows.

1) *Benign:* Benign behavior is expected network traffic and does not raise any flags for IVYCIDE. Benign behavior is dismissed from further analysis.
2) *UER:* Traffic classified as UERs is Unexpected traffic from the E-IoT controller to external servers. Usually, these requests are repetitive during a short span of time and can be associated with POISONIVY DoS attacks.
3) *Malicious CnC Requests:* CnC requests are unexpected network traffic used by a malicious POISONIVY driver to communicate with the command server. As such, malicious CnC requests are associated with an infected E-IoT system actively communicating with a command server.
4) *Activation:* Activation requests are unexpected network traffic used by POISONIVY to initiate attacks. When activation requests are detected, IVYCIDE can determine that an attack was initiated. Thus, these requests may be used to determine the type of traffic that occurs after an attack.

Signature Classifier: Signatures are an accepted method of identifying types of attacks [30]. The Traffic Analyzer uses a signature model and signature-based classification to infer the type of attack occurring from unexpected behavior found during the multiclass classification stage. First, the signature classifier stage will determine if the threshold of unexpected behavior found during the multiclass classification stage was reached within a given window timeframe. If this is the case, the classifier will refer to the Signature Model, a set of rules that define the behaviors that make up the POISONIVY attacks. IVYCIDE will then determine the type of attack that occurred in ongoing traffic using the Signature Classifier. For instance, if a number of unexpected requests for an external server is observed, IVYCIDE will infer that a POISONIVY remote control attack is occurring. As such, it is possible to configure IVYCIDE to classify for future attacks using additional rulesets and additional behaviors. For the purpose of this work, we only consider POISONIVY driver-based attacks.

2) Evaluation Logger: The Evaluation Logger receives the evaluation results and any relevant packet data from the Traffic Handler. As such, the Evaluation Logger acts as a middle stage between the evaluation and the data logs, caching and formatting data into a database compatible format. Essentially, this module allows IVYCIDE users to view prior warnings, see ongoing network communication, and review activity that was deemed to be suspicious.

G. Model Container

IVYCIDE's model container stores the classification model for the E-IoT system. The model container uses several packet attributes as the features to classify E-IoT network activity

and is divided into two submodules, the ML model and the signature-based model.

1) ML Model: The ML model is one of the core subcomponents used by the traffic analyzer for multiclass classification. The ML model should be learned from an active E-IoT system, or generated from expected network traffic. As such, the ML model includes several common features in IP communication (e.g., packet size, TCP flags, UDP flags, and TCP Source/Destination port). Additionally, IVYCIDE includes two custom features described as follows.

1) *Packet Rate:* The number packets an IP source communicates with an IP destination within a 0.1-s time window before and after the given packet. For instance, if the E-IoT controller requests information from a Spotify service at time `t`, the frequency value will show the number packets were sent from the E-IoT controller to the Spotify servers were from time `t-0.1 s to t+0.1 s`.
2) *External Origin:* A Boolean value is set to true if the packet originates from an external source to the E-IoT controller.

2) Signature-Based Model: IVYCIDE uses a signature-based model to infer the type of attack occurring from traffic classified in the multiclass classification stage. The signature-based model contains the signatures of each attack. For instance, the signature-based model would dictate that an activation command, followed by a large number of unauthorized requests to an external target address is likely to be a POISONIVY DoS attack. While POISONIVY attacks are the focus of this work, there may be many more attacks in the future, as new attacks become known, the signature model can be updated to include new attacks. These signatures should be created manually or with tool-assisted definitions. As such, the Signature Model should be flexible, and easily expandable to include current and future attack signatures.

H. User Notification

The User Notification module is used to notify a user or the network administrator on warnings and messages from IVYCIDE. After traffic is analyzed, the Notification Module will detail the traffic logs and give the user all the information necessary to evaluate a possible breach of security. Additionally, the user should receive a notification (e.g., mobile notification) that suspicious activity is occurring in the controller so that they may take action and prevent further issues.

I. Logged Activities

The Logged Activities module acts as a storage database for any information found during IVYCIDE monitoring. The administrator queries the logged activities module and can view the suspicious packets and activity detected by IVYCIDE. Logged Activities only includes packets and data deemed to be of interest to the administrator as well as IVYCIDE's evaluation of the traffic data stored. This module acts as the final stage of IVYCIDE and acts as a point of reference for any network administrator that needs to review logged information and prior events.

TABLE II
HARDWARE AND SOFTWARE USED IN IVYCIDE IMPLEMENTATION AND TESTING

Hardware and Software	
Hardware	Hak5 Plunder Bug
	Acer GX-785 Desktop
Software	Wireshark 3.4.3
	Python 3.9
	Python Scikit-learn
	Visual Studio Code 1.55.2
	Control4 Composer 2.10.6
	Python Scapy
	JupyterLab 3.0.12

VI. IVYCIDE IMPLEMENTATION

To implement IVYCIDE's necessary modules, we used open source, freely available software and libraries. We detail software and hardware used for IVYCIDE in Table II. Our testing environment is identical to the POISONIVY attacks implementation, with the addition of the Hak5 Plunder bug as an active network sniffer between the E-IoT controller and the network router. We assume that the attacker executes the POISONIVY attacks in the same manner as discussed in Section IV, receiving the attack initiation commands from the remote command server and executing the attacks on the local E-IoT system.

A. Network Collector Implementation

The implementation of the Network Collector required the use of the Hak5 Plunder Bug, Wireshark, and Python scripts to process incoming network data. For the Network Collector, the Hak5 Plunder Bug was placed between the E-IoT controller and the network router. The Plunder Bug was then connected the Acer GX-785 desktop for data collection. Data were collected using Wireshark and then preprocessed using `Scapy`, a Python-based library used to manipulate and extract data from Wireshark .pcap files. These data were then passed through our preprocessing software and exported as a comma-delimited string that extracted all of each packet's relevant data (e.g., TCP/UDP ports, TCP/UDP flags, timestamp, source/destination IP, packet size). Additionally, our software added additional attributes.

B. Traffic Handler Implementation

The Traffic Handler and both submodules were implemented using Python with JupyterLab and Visual Studio Code.

1) Traffic Analyzer: The Traffic Analyzer was implemented using JupyterLab and the Python Scikit-learn library used for ML applications. The Traffic Analyzer receives traffic data formatted by the Network Collector and performs classification on each individual packet using KNN, Decision Tree and Random Forest classifiers using the ML Model submodule. Packets are tagged by the Traffic Handler as four distinct types of network activity (e.g., benign, UER, activation, or malicious CnC request) as highlighted in Section V. Packets

TABLE III
SIGNATURES USED IN IVYCIDE

Attack	Rule	Example Signature
DoS	Signature pattern ends in 'A'.	PPPPPPPPA
RC	More than eight 'U' observed in pattern.	PPAUUUUUUUUUUP
RF	'PAP' or 'PAPPPPP' observed in pattern.	PAPAPPPPAPPPP

marked as one of the three types of malicious behaviors are sent sequentially to the signature-based classification stage of the Traffic Analyzer. In the signature-classification stage, the Traffic analyzer refers to the Signature-Based Model for attack classification. All classified packets within the cache window are converted into a string. If this string is beyond a threshold (e.g., seven malicious packets per window) and falls under the known signatures of attacks in the signature-based model, the activity is classified as one of the three well-known POISONIVY attacks. For instance, remote control is detected if two activation packets followed more than eight unexpected request packets are received. A full description of the signatures used in IVYCIDE can be found in Table III. We chose a 3-min cache window as POISONIVY attacks take less than 3 min to execute. Reducing the speed of the attack network throughput (e.g., less packets per second for a remote DoS attack) would greatly reduce an attack's effectiveness.

2) Evaluation Logger Implementation: The Evaluation Logger was implemented as a Python-based console notification that provides the classification of E-IoT traffic data.

C. Model Container Implementation

The IVYCIDE Model Container contains two models used for classification purposes: 1) the *ML Model* and 2) the *Signature-Based Model*. In this section, we overview both models and the data collection process used to implement these models.

1) ML Model Implementation: The ML Model was stored as a Python object in Jupyterlabs as a list of fitted models. Each model was then queried by the program to process each incoming packet sequentially. For the implementation, we evaluated several classifiers including: Nearest Neighbors, Decision Tree, and Random Forest classifiers. We chose the Decision Tree classifier for the final implementation as it provided adequate classification accuracy and precision for the purposes of IVYCIDE. For better classification, we also introduced the following features.

1) *Frequency:* The frequency was calculated using a sliding window during data collection. Essentially, IVYCIDE stores packets for a given time window (100 ms), then calculates how many packets share the same source and destination address within the time window.
2) *External Origin:* The external origin feature was created by comparing the known IP address of the E-IoT controller and setting this flag to "True" if the E-IoT controller was the destination of the packet.

2) Signature-Based Model Implementation: The Signature-Based model was implemented as a set of rules in our custom Python software. We apply these rules to the signature string generated from the first stage of classification. We highlight the signature rule table as follows.

1) *Benign:* A set of traffic data is classified as Benign if the attack pattern does not contain activation commands, indicating no attack was initiated and malicious communication was infrequent.
2) *DoS:* A set of traffic data is classified as DoS if the final commands (last five) in attack pattern are classified under "activation." This behavior indicates that the E-IoT controller became unavailable after an activation command was received from the attacker.
3) *Remote Control:* Traffic data is classified as Remote Control if there is a high frequency of packets classified as UERs in the Multiclass stage. This signature string indicates that the E-IoT controller is making multiple UERs in a short timeframe.
4) *Resource Farming:* A set of traffic data is classified as Malicious Resource farming if CnC requests are observed after activation without UER. This behavior indicates an attack was activated, however, the E-IoT controller is still functional after attack activation.

D. Other Implementations

The User Notification module was implemented using the Python `ctypes` library to create a notification on the machine running the core IVYCIDE software. The Logged Activities module was implemented as direct text file exports on the local machine, allowing for future reference of the attack logs and providing any relevant information of the IVYCIDE analysis.

VII. PERFORMANCE EVALUATION

In this section, we evaluate the performance of IVYCIDE in detecting POISONIVY attacks. Specifically, we attempt to answer the following research questions.

RQ1 (Malicious Behavior Evaluation): How do different ML classification algorithms perform in detecting malicious activity based on individual network packets? (Section VII-B).

RQ2 (Malicious Activity Type): How effective is IVYCIDE in classifying between different POISONIVY attacks with signature-based detection? (Section VII-C).

A. Attack Data Collection

Based on the previously mentioned POISONIVY attacks, we performed the attacks as specified in Section IV.

To train IVYCIDE, we collected daily usage data from the E-IoT environment by performing expected operation with the E-IoT system as defined in Section V. Expected operation involved the use of the E-IoT environment for streaming media, volume control, menu navigation, and any use consistent with an expected smart system usage. Benign data were collected from the E-IoT environment over the span of two weeks, where the system was allowed to idle, turn on, turn off, and otherwise operate in a manner consistent with expected operation. Malicious data was captured as detailed in the POISONIVY attacks. In total, we collected 525 705 packets from the E-IoT system for testing and training. The collection resulted in a total of 60 data sets of attack data, 20 for each attack. Additionally, we recorded 20 data sets of expected network traffic from the E-IoT system as defined in

TABLE IV
MULTICLASS CLASSIFICATION OF MALICIOUS E-IoT TRAFFIC BEHAVIORS

		External Request							
Model	Class	TPR	TNR	FPR	FNR	ACC	PREC	REC	F1
DT	BEN	0.98	0.91	0.08	0.02	0.96	0.97	0.98	0.98
	CnC	0.92	0.97	0.01	0.09	0.98	0.97	0.92	0.94
	ACT	0.99	0.96	0.00	0.00	1.00	1.00	1.00	1.00
	UER	0.84	0.97	0.01	0.18	0.99	0.82	0.85	0.83
KNN	BEN	0.98	0.85	0.14	0.01	0.96	0.95	0.99	0.97
	CnC	0.84	0.98	0.00	0.19	0.96	0.98	0.84	0.91
	ACT	0.94	0.95	0.00	0.06	1.00	0.96	0.94	0.95
	UER	0.92	0.96	0.00	0.09	0.99	0.90	0.92	0.91
RF	BEN	0.99	0.92	0.08	0.01	0.97	0.97	0.99	0.98
	CnC	0.93	0.98	0.01	0.07	0.98	0.95	0.94	0.95
	ACT	0.99	0.97	0.00	0.01	1.00	0.99	0.99	0.99
	UER	0.85	0.98	0.01	0.17	0.99	0.87	0.85	0.86

Legend : DT = Decision Tree, KNN = Nearest Neighbors, RF = Random Forest
BEN = Benign, CnC = Command & Control, ACT = Activation, UER = Unauth.

Section V for a total of 80 data sets. To train the model, we followed a supervised learning approach, requiring labeled data for the training. We found that it was necessary to use supervised learning to properly categorize the four types of network activity from the E-IoT controller and evaluate IVYCIDE. All of the POISONIVY attacks were executed as highlighted in Section IV. For Attack 1. we performed a remote activation of DoS attack, disabling the E-IoT controller. For Attack 2, our attacker issued a CnC request to the malicious driver to perform unauthorized requests to a target webserver. Finally, for Attack 3, we issued a CnC request for the E-IoT controller to begin performing resource-intensive calculations on behalf of the attacker.

1) Performance Metrics: Performance metrics in our work follow the accepted parameters: accuracy, precision, *F*-score, recall, true-positive rate (TPR), true-negative rate (TNR), false-positive rate (FPR), and false-negative rate (FNR).

TPR: It denotes the total number of correctly identified benign traffic within the test environment.

TNR: It denotes the total number of correctly identified malicious traffic within the test environment.

FPR: It denotes the total number of cases where malicious traffic was mistaken as being benign.

FNR: It denotes the total number of cases where benign traffic is mistaken as malicious

$$\text{Recall Rate} = \frac{\text{TNR}}{\text{TNR} + \text{FPR}} \tag{1}$$

$$\text{Precision Rate} = \frac{\text{TPR}}{\text{TPR} + \text{FPR}} \tag{2}$$

$$\text{Accuracy} = \frac{\text{TPR} + \text{TNR}}{\text{TPR} + \text{TNR} + \text{FPR} + \text{FNR}} \tag{3}$$

$$F1 = \frac{2 * \text{Recall Rate} * \text{Precision Rate}}{\text{Recall Rate} + \text{Precision Rate}}. \tag{4}$$

B. IVYCIDE *Performance for Different Classifiers (RQ1)*

As part of *RQ1*, we evaluate different ML-based classifiers and their performance on individual network traffic packets. As highlighted in Section V, IVYCIDE may use the most effective classifier to classify between behavior types. For *RQ1* IVYCIDE evaluation, we implemented several classifiers, highlighting *Decision Tree*, *Nearest Neighbors*, and *Random Forest* with the best performance. We refer to Table IV for the performance of each classifier used with IVYCIDE. In these results we show how different classifiers perform against E-IoT network traffic in terms of accuracy and precision. For all of the covered classifiers, we observed accuracy and precision rates averaging higher than 90%.

TABLE V
SIGNATURE-BASED CLASSIFICATION OF MALICIOUS E-IoT ATTACKS

	Farming							
Class	TPR	TNR	FPR	FNR	ACC	PREC	REC	F1
BEN	1.0	1.0	0.00	0.00	1.0	1.0	1.0	1.0
BOT	1.0	0.98	0.03	0.00	0.98	0.91	1.0	0.95
DOS	0.75	1.00	0.00	0.25	0.94	1.00	0.75	0.86
MRF	0.95	0.95	0.05	0.05	0.95	0.86	0.95	0.90

Legend : BEN = Benign, BOT = Botnet, DOS = Denial-of-Service,
MRF = Resource

UER were particularly challenging to classify. In most cases, UER was misclassified as CnC attacks. This is possibly due to the fact that the internal programming functions to perform UER requests in the attack code are identical to CnC attacks. The IVYCIDE architecture (Section V) highlights that the multiclass classification is the first step for IVYCIDE. We note that perfect classification accuracy on individual packets is not required for effective signature classification because attack signatures have some matching tolerance given the rulesets. Furthermore, the configurable design of IVYCIDE, means that evaluating different classifiers yields to valuable information. For instance, some classifiers may have more success at classification on some E-IoT deployments and configuration than others. As such, since E-IoT systems are highly heterogeneous, IVYCIDE can be adapted with one or multiple classifiers to provide better accuracy and precision for individual deployments.

C. IVYCIDE *Signature-Based Classification Performance (RQ2)*

We refer to Table V for IVYCIDE's signature-based classification performance in terms of accuracy, precision, recall, and *F*1 metrics for each attack type. As such, we note that IVYCIDE achieved an overall accuracy of 97% and precision of 94%. More notable is that no malicious cases were classified as benign, as such, even if an attack is misclassified as another attack, the administrator will still be alerted of suspicious traffic. Specifically, we found that three DoS attacks were misclassified as malicious resource farming attacks. This may be due to both POISONIVY DoS (memory exhaustion on the controller) and resource farming attacks low network throughput.

In some cases, we found that the music streaming service TuneIn, caused false positives. IVYCIDE improperly classified some benign data from the streaming service as unauthorized requests. We believe that the addition of whitelisting to approved IP addresses may further improve the accuracy of IVYCIDE since attackers cannot spoof addresses using the driver API. However, even without whitelisting, the number of unauthorized requests in our proof-of-concept attacks were limited as legal limitations with the target Amazon Web Services hosted website do not allow for DDoS attacks. Specifically, Amazon Web Services explicitly prohibits any

type of DDoS testing what would put any stress on their servers. Traffic-based DoS attacks performed by attackers without legal concerns would create many more observable unexpected requests within a given timeframe from an E-IoT controller and, as such, become easier to identify using IVYCIDE.

We note that the classification performance was accomplished using only *black-box integration and with no modification to the E-IoT controller, drivers, or system code*. While some attacks were misclassified as other attacks, all malicious instances of attacks were detected as suspicious activities. Similarly, benign activity was properly classified in all cases, greatly reducing the number of false alarms by the signature-based classification. As such, in any system implementation, network administrators would have been alerted for all attacks, been able to investigate attacks further, and take action against an infected E-IoT controller.

D. Detection Time and Overhead

How quickly attacks are detected is dependent on the attacks and the attack payload through the network. For instance, a remote control (DoS) attack is much more noticeable in network traffic than a DoS attack. As such, the maximum time it would take for an attack to be detected is the time window given for IVYCIDE. We measured the CPU usage and memory consumption of IVYCIDE for each stage with 4.5% CPU usage and 11.6 MB of peak usage for the multiclass classification stage; and a 0.3% CPU and 19.2-MB peak usage for the signature classification stage. We must note that this overhead is only applied to the computer running the IVYCIDE system (16 GB RAM and i7-700 3.6 GHz) and not to the E-IoT system controller.

VIII. Related Work

Research and active attacks against smart devices have been an ongoing topic of research in recent years. Within the scope of smart homes, work by Abrishamchi *et al.* [3] summarized side-channel threats in smart home systems. Work presented by Koh *et al.* highlights how smart building applications are often over-privileged for their purpose. Defense mechanisms have been proposed in response to threats in smart buildings, homes and appliances [17], [20], [31], [40]. As early as 2013, some works highlight various threats in smart devices and note that attackers are in constant search of new methods to infect smart devices [5], [7], [8], [9], [10], [11], [12], [13], [18], [26], [32], [34], [35], [36], [48], [49]. Additionally, research in alternative attack vectors, such as HDMI, USB, and E-IoT buses exist, which can provide attackers with an expanded attack vector as E-IoT systems become more well-known to attackers [27], [39], [41], [43].

In terms of IoT, there have been numerous works covering IoT attacks and defense mechanisms in recent years. We do not go into details in this study but we refer readers to the surveys of Rondon *et al.* [44] for attacks, threats, and defenses against E-IoT systems, Aris *et al.* [6] for the intrusion detection and mitigation mechanisms applicable to 6LoWPAN-based IoT networks. For the existing ML-based and traditional network IDSs (NIDSs), we refer to the survey by Chaabouni *et al.* [19]. For research on attacks and defenses on wireless sensor networks, we refer to a survey by Butun *et al.* [14] on emerging sensor threats and security. Furthermore, for works on distributed IoT devices, attack techniques, and defenses are covered in a survey by Vishwakarma and Jain [50]. Finally, individual defense mechanisms such as the one proposed by Sforzin *et al.* [46] investigates the use of Snort on Raspberry Pis to create an IDS for IoT systems. Another notable example is Flowguard, an edge-defense mechanism proposed by Jia *et al.* [29] to mitigate against IoT DDoS attacks.

Our work differs from the previously discussed works as POISONIVY focuses on the insecurity of E-IoT system drivers, an attack vector which has been largely unexplored. Further. IVYCIDE targets this threat vector and offers a solution to an under-researched problem. Specifically, for this work we presented three threats possible through malicious E-IoT drivers: 1) DoS attacks on the host system; 2) remote control of a target E-IoT system; and 3) the malicious farming of system resources for unauthorized activities (e.g., bitcoin mining). To address these threats, we introduced IVYCIDE, a defense mechanism accounting for E-IoT system design and tailored specifically to E-IoT systems. Furthermore, IVYCIDE poses no modification or overhead to the original E-IoT system, and defends with a passive two-step network traffic defense framework.

IX. Conclusion

Recent years have seen a dramatic rise in IoT systems and applications that enabled billions of commodity IoT devices to empower smarter settings in buildings, offices, and homes. Although commodity IoT devices are employed by ordinary end users, more reliable, complex, customized, and robust E-IoT solutions are required for enterprise customers. With the higher price, customization, robustness, and scalability of E-IoT systems, they are commonly found in settings, such as smart buildings, government or private smart offices, academic conference rooms, luxury smart homes, and hospitality applications. With very little research investigating the security of E-IoT systems and their components, E-IoT systems in professional smart settings present an unexplored threat vector. In this work, we explored E-IoT system vulnerabilities and insecure development practices, specifically, the usage of drivers as an attack mechanism. We implemented an E-IoT system testbed in a smart building setting and introduced POISONIVY, a novel attack mechanism to show that it is possible for a malicious actor to easily attack and command E-IoT system controllers using malicious drivers. Specifically, with POISONIVY, we demonstrated that an attacker may cause DoS conditions, take control of E-IoT system controllers, and remotely abuse the resources of the such systems for illegal activities (e.g., bitcoin mining). To defend against these threats, we introduced IVYCIDE, a novel, configurable defense mechanism designed specifically for E-IoT systems. As IVYCIDE operates as a standalone framework, it provides no additional overhead to E-IoT systems. Finally, we evaluated the IVYCIDE performance on a realistic E-IoT system. Our analysis showed that IVYCIDE

achieved 97% in accuracy and 94% precision for attack-type identification.

ACKNOWLEDGMENT

The views are those of the authors only.

REFERENCES

[1] "Crestron application market: User upload guide." Jan. 2019. Accessed: Dec. 10, 2019. [Online]. Available: https://applicationmarket.crestron.com/user-upload-guide/

[2] "The number of smart homes in Europe and North America reached 45 million in 2017." Sep. 2018. Accessed: Dec. 10, 2019. [Online]. Available: https://cutt.ly/wpW2vLt

[3] M. A. N. Abrishamchi, A. H. Abdullah, A. D. Cheok, and K. S. Bielawski, "Side channel attacks on smart home systems: A short overview," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2017, pp. 8144–8149.

[4] J. A. Dev, "Usage of botnets for high speed MD5 hash cracking," in *Proc. 3rd Int. Conf. Innov. Comput. Technol. (INTECH)*, Aug. 2013, pp. 314–320.

[5] A. Arabo and B. Pranggono, "Mobile malware and smart device security: Trends, challenges and solutions," in *Proc. 19th CSCS*, May 2013, pp. 526–531.

[6] A. Aris, S. F. Oktuğ, and T. Voigt, "Security of Internet of Things for a reliable Internet of Services," in *Autonomous Control for a Reliable Internet of Services*. Cham, Switzerland: Springer, 2018. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-90415-3_13

[7] 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 ICC*, 2020, pp. 1–7.

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

[9] 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, 2020.

[10] L. Babun, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "Real-time analysis of privacy-(un)aware IoT applications," 2019, *arXiv:1911.10461*.

[11] L. Babun, A. K. Sikder, A. Acar, and A. S. Uluagac, "IoTDots: A digital forensics framework for smart environments," 2018, *arXiv:1809.00745*.

[12] L. Babun, H. Aksu, and S. A. Uluagac, "Detection of counterfeit and compromised devices using system and function call tracing techniques," U.S. Patent 10 027 697 B1, Jul. 2018.

[13] L. Babun, H. Aksu, and S. A. 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 B1, Mar. 2019.

[14] I. Butun, P. Österberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 616–644, 1st Quart., 2020.

[15] "Control4 driver programming." C4Drivers. Oct. 2014. Accessed: Dec. 10, 2019. [Online]. Available: https://c4drivers.wordpress.com/2014/10/13/hello-world/

[16] "Control4 forums files download." C4Forums. Accessed: Jan. 23, 2020. [Online]. Available: https://www.c4forums.com/files/

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

[18] Z. B. Celik *et al.*, "Sensitive information tracking in commodity IoT," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 1687–1704.

[19] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019.

[20] A. B. Chen, M. Behl, and J. L. Goodall, "Trust me, my neighbors say it's raining outside: Ensuring data trustworthiness for crowdsourced weather stations," in *Proc. BuildSys*, 2018, pp. 25–28.

[21] "Control4 driver search." Control4. Jan. 2019. Accessed: Dec. 10, 2019. [Online]. Available: https://drivers.control4.com/solr/drivers/browse

[22] "Control4 operating system OS release notes." Control4. 2010. Accessed: Jun. 20, 2020. [Online]. Available: https://silo.tips/download/control4-operating-system-os-release-version-releasenotes-copyright-2010-contro

[23] "Press release: Four years in a row, Control4 named leading whole-house automation brand in CEPro brand analysis." Control4. 2018. Accessed: Jun. 20, 2020. [Online]. Available: https://www.control4.com/press_releases/2018/07/05/four-years-in-a-row-control4-named-leading-whole-house-automation-brand-in-2018-cepro-brand-analysis/

[24] "Getting started with composer pro." Control4. Jun. 2010. [Online]. Available: https://www.yumpu.com/en/document/view/35663397/composer-pro-getting-started

[25] "Crestron technical institute." Crestron. Accessed: Dec. 10, 2019. [Online]. Available: https://www.crestron.com/Training-Events/Training

[26] K. Denney, L. Babun, and A. S. Uluagac, "USB-watch: A generalized hardware-assisted insider threat detection framework," *J. Hardw. Syst. Security*, vol. 4, pp. 136–149, Mar. 2020.

[27] K. Denney, E. Erdin, L. Babun, M. Vai, and S. Uluagac, "USB-watch: A dynamic hardware-assisted USB threat detection framework," in *Proc. SecureComm*, 2019, pp. 126–146.

[28] "Control4 drivers." Drivercentral. 2020. Accessed: May 20, 2020. [Online]. Available: https://drivercentral.io/platforms/control4-drivers

[29] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9552–9562, Oct. 2020.

[30] "Attack signature." Kaperski. 2021. Accessed: Jan. 15, 2022. [Online]. Available: https://encyclopedia.kaspersky.com/glossary/attack-signature/

[31] A. Karapetyan, S. C.-K. Chau, K. Elbassioni, M. Khonji, and E. Dababseh, "Smart lighting control using oblivious mobile sensors," in *Proc. BuildSys*, 2018, pp. 158–167.

[32] 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 ICC*, 2018, pp. 1–6.

[33] K. Lancaster. "Control4 delivers high-resolution audio and homeowner personalization enhancements to elevate the smart home experience." Sep. 2018. Accessed: Dec. 10, 2019. [Online]. Available: https://cutt.ly/8pW1b2Q

[34] J. Lopez, L. Babun, H. Aksu, and A. Uluagac, "A survey on function and system call hooking approaches," *J. Hardw. Syst. Security*, vol. 1, pp. 114–136, Sep. 2017.

[35] 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)*, 2019, pp. 1–6.

[36] 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*, 2020, pp. 1–9.

[37] *Viruses Frame PC Owners for Child Porn*, CBS News, New York, NY, USA, Nov. 2009. Accessed: Dec. 10, 2019.

[38] Pinkoos. "Apple TV tvOS 13 killed my remote programming." 2019. Accessed: May 20, 2020. [Online]. Available: https://www.c4forums.com/topic/32727-psa-apple-tv-tvos-13-killed-my-remoteprogramming/

[39] L. C. P. Rondon, L. Babun, K. Akkaya, and A. S. Uluagac, "HDMI-watch: Smart intrusion detection system against HDMI attacks," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2060–2072, Jul.–Sep. 2021.

[40] H. Rashid, N. Batra, and P. Singh, "R_{IMOR}: Towards identifying anomalous appliances in buildings," in *Proc. BuildSys*, 2018, pp. 33–42.

[41] L. P. Rondon, L. Babun, K. Akkaya, and A. S. Uluagac, "HDMI-walk: Attacking HDMI distribution networks via consumer electronic control protocol," in *Proc. 35th ACSAC*, 2019, pp. 650–659.

[42] 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. BuildSys*, New York, NY, USA, 2020, pp. 130–139.

[43] L. P. Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac, "LightningStrike: (in)secure practices of E-IoT systems in the wild," in *Proc. 14th ACM Conf. Security Privacy Wireless Mobile Netw.*, New York, NY, USA, 2021, pp. 106–116.

[44] L. P. Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac, "Survey on enterprise Internet-of-Things systems (E-IoT): A security perspective," *Ad Hoc Netw.*, vol. 125, Feb. 2022, Art. no. 102728.

[45] S. Seth. "What is botnet mining?" 2019. Accessed: Jan. 23, 2020. [Online]. Available: https://www.investopedia.com/tech/what-botnet-mining

[46] A. Sforzin, F. G. Mármol, M. Conti, and J.-M. Bohli, "RPiDS: Raspberry Pi IDs—A fruitful intrusion detection system for IoT," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput. Adv. Trusted Comput. Scalable Comput. Commun. Cloud Big Data Comput. Internet People Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, 2016, pp. 440–448.

[47] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thSense: A context-aware sensor-based attack detector for smart devices," in *Proc. 26th USENIX Security*, 2017, pp. 397–414.

[48] A. K. Sikder, L. Babun, H. Aksu, and A. S. Uluagac, "Aegis: A context-aware security framework for smart home systems," in *Proc. ACSAC*, 2019, pp. 28–41.
[49] A. K. Sikder *et al.*, "Kratos: Multi-user multi-device-aware access control system for the smart home," in *Proc. 13th ACM WiSec*, 2020, pp. 1–12.
[50] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," *Telecommun. Syst.*, vol. 73, no. 1, pp. 3–25, 2020.
[51] Zaphod. "Why is Lua used for Control4 driver programming." May 2017. [Online]. Available: https://www.c4forums.com/topic/24086-why-is-lua-used-for-control4-driverprogramming/

Luis Puche Rondon received the bachelor's degree in computer science and the master's degree in cybersecurity from Florida International University, Miami, FL, USA, in 2016 and 2017, respectively.

He is a member of the Cyber-Physical Systems Security Lab, Department of Electrical and Computer Engineering, Florida International University, and a CyberCorps Scholarship for Service Fellow. He has over a decade of experience in audio/video, smart homes, professional smart systems, and CCTV installations. His research interests focus on the security of enterprise Internet of Things systems, and the security implications of threats under-researched threat vectors.

Leonardo Babun (Member, IEEE) received the first master's degree in electrical engineering from Florida International University, Miami, FL, USA, in 2015, and the second master's degree in computer engineering and the Doctoral degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, Florida International University, in 2019 and 2020, respectively.

He is currently a member of the Cyber-Physical Systems Security Lab and a CyberCorps Scholarship for Service Alumni with the Department of Electrical and Computer Engineering, Florida International University. His research interests are focused on the security and privacy of cyber–physical systems and Internet of Things.

Ahmet Aris received the B.Sc. degree in computer engineering from Bahcesehir University, Istanbul, Turkey, in 2009, and the M.Sc. and Ph.D. degrees in computer engineering from the Graduate School of Science, Engineering and Technology, Istanbul Technical University, Istanbul, 2012 and 2019, respectively.

He is a Research Assistant Professor with the Department of Electrical and Computer Engineering, Florida International University, Miami, FL, USA, and is conducting research in Cyber-Physical Systems Security Lab under the supervision of Dr. A. S. Uluagac. He also worked with the Medianova CDN R&D Center, Istanbul, as an R&D Analyst. He conducted research with the Networked Embedded Systems Group, Swedish Institute of Computer Science, Kista, Sweden, as a Visiting Researcher. His research interests include IoT security, network security, Web security, and malware.

Kemal Akkaya (Senior Member, IEEE) received the B.S. degree in computer engineering and information sciences from Bilkent University, Ankara, Turkey, in 1997, the M.S. degree in computer engineering from Middle East Technical University, Ankara, in 1999, and the Ph.D. degree in computer science from the University of Maryland Baltimore County, Baltimore, MD, USA, in 2005.

He leads the Advanced Wireless and Security Lab, Florida International University, Miami, FL, USA. His research areas span various challenges of mobile and wireless networks, Internet of Things, and cyber–physical systems, such as security, privacy, quality of service, topology control, and mobility management.

Dr. Akkaya is the Area Editor of the *Ad Hoc Networks* (Elsevier) and serves for the editorial board of the IEEE COMMUNICATION SURVEYS AND TUTORIALS. He has been a guest editor for various journals and serves for the organizing committees of leading IEEE communication conferences, such as IEEE LCN, ICC, Globecom, WCNC, and SmartGridComm. He is a member of the IEEE Computer Society and IEEE Technical Committees on Communication, Cybersecurity, Smart Cities, and Online Social Networks.

A. Selcuk Uluagac received the first M.Sc. degree in electrical and computer engineering (ECE) from Carnegie Mellon University, Pittsburgh, PA, USA, in 2002, the second M.Sc. degree in information security from the School of Computer Science, The Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA, in 2009, and the Ph.D. degree with a concentration in information security and networking from the School of ECE, Georgia Tech in 2010.

He is currently an Associate Professor with the Department of ECE, Florida International University (FIU), Miami, FL, USA. Before joining FIU, he was a Senior Research Engineer with the School of ECE, Georgia Tech. Prior to Georgia Tech, he was a Senior Research Engineer with Symantec, Mountain View, CA, USA. The focus of his research is on cybersecurity topics with an emphasis on its practical and applied aspects. He is interested in and currently working on problems pertinent to the security of cyber–physical systems and Internet of Things.

Dr. Uluagac received the Faculty Early Career Development (CAREER) Award from the U.S. National Science Foundation in 2015 and the Summer Faculty Fellowship from the University of Padua, Padua, Italy, in 2016. He was awarded the U.S. Air Force Office of Sponsored Research's 2015 Summer Faculty Fellowship in 2015. He is currently the Area Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING and *Computer Networks* (Elsevier), and serves for the editorial board of the IEEE COMMUNICATION SURVEYS AND TUTORIALS as the Network Security Track Lead. He is also an active member ACM and ASEE and a regular contributor to national panels and leading journals and conferences in the field. More information can be obtained from: http://web.eng.fiu.edu/selcuk.