The Truth Shall Set Thee Free: Enabling Practical Forensic Capabilities in Smart Environments

Leonardo Babun, Amit Kumar Sikder, Abbas Acar, and A. Selcuk Uluagac
Cyber-Physical Systems Security Lab
Department of Electrical & Computer Engineering, Florida International University
Email: {lbabu002, asikd003, aacar001, suluagac}@fiu.edu

***Abstract*—In smart environments such as smart homes and offices, the interaction between devices, users, and apps generate abundant data. Such data contain valuable forensic information about events and activities occurring in the smart environment. Nonetheless, current smart platforms do not provide any digital forensic capability to identify, trace, store, and analyze the data produced in these environments. To fill this gap, in this paper, we introduce VERITAS, a novel and practical digital forensic capability for the smart environment. VERITAS has two main components: Collector and Analyzer. The Collector implements mechanisms to automatically collect forensically-relevant data from the smart environment. Then, in the event of a forensic investigation, the Analyzer uses a First Order Markov Chain model to extract valuable and usable forensic information from the collected data. VERITAS then uses the forensic information to infer activities and behaviors from users, devices, and apps that violate the security policies defined for the environment. We implemented and tested VERITAS in a realistic smart office environment with 22 smart devices and sensors that generated 84209 forensically-valuable incidents. The evaluation shows that VERITAS achieves over 95% of accuracy in inferring different anomalous activities and forensic behaviors within the smart environment. Finally, VERITAS is extremely lightweight, yielding no overhead on the devices and minimal overhead in the backend resources (i.e., the cloud servers).**

I. INTRODUCTION

Modern technology has quickly evolved as a network of Internet-enabled (i.e., smart) devices. These smart devices, such as smart lights and smart thermostats, communicate with each other and interact with the users' day-to-day activities through sensors (e.g., motion and temperature sensors) to enable the concept of the *smart environment*. Such an environment improves the quality of life of the people while handling a new set of data previously untapped [1]–[3] and with tremendous forensic value [4]–[7]. For instance, in a smart office setup containing motion sensors, forensic evidence extracted from the state of the sensors may reveal the presence of individuals at unauthorized hours. On a larger scale, the analysis performed on data collected from multiple devices may reveal substantially more forensic details about the activities occurring in the smart environment at any time. In fact, in some countries, insurance companies have started giving incentives to customers if they install measures like smart lighting, smart energy management systems, smart fire, or water monitoring devices [8].

Nonetheless, *current smart home/office platforms do not provide any means for forensic analysis.* Indeed, the limitation of available computing resources in the majority of the smart devices [9]–[14] and the distinctive cloud-based architecture of most smart home/office platforms [15] make it very challenging to store data inside these devices [16]–[19], rendering traditional forensic analysis unsuitable for the smart environment. Additionally, the most popular platforms (e.g., Samsung SmartThings, Apple HomeKit) do not provide any mechanisms to access and indefinitely store smart data in the cloud [20]. Prior research works have proposed logging techniques to collect data from smart apps and devices [21], [22] to implement, for instance, IoT data provenance analysis [23]. However, these works either (1) do not specifically focus on utilizing the data collected from the smart environment to implement forensic solutions; (2) assume trusted devices which may be unrealistic nowadays [24]; or (3) do not directly consider forensically-relevant activities and behaviors in their threat models. In addition, traditional forensic solutions targeting smart systems only focus on specific data formats (e.g., video streaming extracted from a smart camera) to perform their analysis [25], which are not always available in every forensic evaluation.

To overcome these limitations, we introduce VERITAS[1], a novel and practical forensic analysis framework that uses data collected from the smart environment to support traditional forensic investigations. With VERITAS, we aim to fill the gap between current digital forensics and the unique data-rich characteristics of modern smart environments, providing a forensic solution that considers data from every device and app used in the smart environment for a more comprehensive analysis. VERITAS has two main components: the VERITAS Collector (VC) and the VERITAS Analyzer (VA). The VC automatically analyzes and modifies smart

[1]In Roman mythology, VERITAS is the goddess of truth.

Network and Distributed Systems Security (NDSS) Symposium 2022
24-28 April 2022, San Diego, CA, USA
ISBN 1-891562-74-6
https://dx.doi.org/10.14722/ndss.2022.24133
www.ndss-symposium.org

applications to flag and collect forensically-relevant data at runtime. Then, in the event of a forensic investigation, the VA applies a First Order Markov Chain models on the collected data to infer the forensically-relevant activities and behaviors that occurred in the smart environment. VERITAS considers these activities and behaviors, and the security policies defined for the smart environment to detect security violations from users, devices, or smart apps. We evaluate VERITAS in a real-life smart environment with 22 smart devices and sensors that generated 84209 forensically-valuable incidents. Our experimental results demonstrate that VERITAS achieves over 95% accuracy in inferring anomalous activities and detecting malicious forensic behaviors. Finally, VERITAS yields minimal overhead to smart devices, apps, and the cloud servers.

Summary of Contributions. This work has both conceptual and methodological novel contributions:

- *Conceptual*: We propose an end-to-end forensic framework, VERITAS, that matches the activities and behaviors of users, devices, and apps within a smart environment against security policies to support traditional forensic investigations.
- *Methodological-1*: We design and implement a source code analysis tool that identifies and extracts, via automatic app instrumentation, forensically-relevant data from smart apps (i.e., VERITAS Collector). We made the Collector freely available online at https://iotdots-modifier.appspot.com/.
- *Methodological-2*: We build a First Order Markov-Chain model that analyzes the data collected from the smart environment to infer the activities and behaviors that potentially violate the security policies.
- *Methodological-3*: We evaluate the effectiveness of VERITAS in a realistic smart office environment with 22 devices and sensors. VERITAS is effective in inferring anomalous user activities with a minimum 95.4% accuracy. Also, the framework detects forensic behaviors that violate the security policies in place in at least 95.3% of the cases.

Organization. In Section II, we present the background information. In Section III, we introduce the problem and the threat model. Section IV details the architecture of VERITAS and the analysis techniques used to extract forensic information from the smart environment. The evaluation results are detailed in Section V and Section VI discusses security aspects of VERITAS. The related work is discussed in Section VII and Section VIII concludes the paper.

II. FROM TRADITIONAL TO SMART FORENSICS

Traditional digital forensics refers to the investigation and the uncovering of any digital evidence (i.e., data) from electronic devices like computers and cellphones [26]–[32]. The most basic application of digital forensic investigations is to either support or oppose the hypothesis proposed in a criminal case before a court. Frequently, digital forensics require the analysis of physical evidence directly related to people's presence in the crime scene such as an audio or video file. In other cases, a more in-depth investigation of scientific evidence such as network data or mobile device data may be required. Complex cases may require a combination of both types of analysis. Although there are different approaches, a standard digital forensic process can be structured in five different stages [33], [34]: (1) *Identification*, (2) *Preservation*, (3) *Recovery*, (4) *Analysis*, and (5) *Presentation*. During the identification stage, the investigator marks and identifies the relevant digital media (e.g., computer hard drive) containing the forensic data to be retrieved. Then, she takes the necessary steps to guarantee that such data is preserved and no action is performed on the selected media that could jeopardize the relevant forensic data. Further, during recovery, different techniques are applied to the media to extract the data from the digital media without modifying it. Finally, the extracted data is analyzed and the forensic findings are reported using predetermined report formats. Depending on the type of device analyzed and the source of the evidence, the methodology used and the type of evidence extracted may differ. For instance, while a mobile device is a good source of information for tracking the location of a person, a computer may have the capability to provide more detailed and varied intelligence.

Smart Forensics. We use the term *smart forensics* to refer to the traditional forensic analysis performed on data extracted from a smart environment. With the growth of smart home and smart office solutions, digital forensics is not anymore limited to storage devices like hard drives, USB drives, computers, or smartphones [35]. Now, the data from devices like smart locks and motion sensors may also be used for forensic purposes [36], [37]. These devices generate and handle a very diverse set of heterogeneous data from their interaction with users' day-to-day activities and other devices. For example, a smart lock can reveal when someone entered the building, or a smart speaker may reveal the exact location of its user at the time of a forensic incident. However, the use of data from smart devices is not a straightforward exercise due to some unique challenges present in the smart environment:

Limited Computing Resources: Most of the smart devices are both computational and storage resource-limited [10], [11], [16]. These restrictions limit the storing and processing of data inside the devices for forensic purposes. In fact, most smart platforms offer cloud-based solutions [38]–[40] and no data is ever stored inside the devices.

High Divergence and Diversity of Data: In traditional forensic investigations, the data is typically collected from specific devices of interest. However, in smart forensics, a single case may require the collection of data from a diverse set of devices [41]. Different data sources may have diverse impacts and meanings during a forensic investigation. For instance, even though generated from the same activity (e.g., a user walking through a restricted area), the data collected from a motion sensor and a smart thermostat may have different format and size. Therefore, it is required to have a dedicated and reliable forensic mechanism that can successfully handle and analyze the data collected from different types of smart devices.

Lack of Forensic Analysis Support: Even though smart data may contain a high forensic value, smart platforms do

not offer any means to support forensic investigations. For instance, as devices do not normally store data, the smart home/office platforms do not provide specific mechanisms to collect such data from the smart environment at runtime. One solution could be to monitor the devices' traffic passively and collect information as needed. However, such a procedure might be impractical as most of the smart environment communication is encrypted [42], [43]. Also, as forensic analysis in the smart home is a novel approach, traditional mechanisms of data preservation and chain of custody must be enabled for the very distributed smart data.

III. Definitions and Threat Model

A. Problem Definition

This work assumes that there exists an office O. The office has deployed several devices to create a fully equipped smart environment. The topology of the smart environment in O includes devices like smart thermostats, locks, lights, presence sensors, security cameras, and smoke detectors. We also assume that the general manager, Bob, is the only person in O with administrative rights to handle the apps that control the smart devices. By policy, these apps are the only ones authorized to manage the devices inside O. Bob, however, is not authorized to modify the security policies in place for the smart environment. Finally, the security policies of O prohibit the presence of any person between 8:00 pm to 7:00 am from Monday through Friday and anytime during the weekends. At some point, a fire incident inside O has caused the loss of sensitive information along with important economic consequences. After the incident, the insurance company requests a forensic investigation.

We propose VeritaS as a novel framework that utilizes the data extracted from the smart apps to perform the forensic analysis of the events in O. Indeed, VeritaS can be used in conjunction with traditional forensic analysis tools and techniques to hold the person, smart app, or device (if any) accountable if a case of negligence or deliberate violation of security policies is detected. By using VeritaS, we can answer several forensic questions: (1) What was happening inside O right before the fire incident had occurred (e.g., right before the smart smoke detector started sensing the smoke presence)? (2) Was anyone inside the room (e.g., presence sensor state changed)? (3) Was the door opened/closed anytime before the fire incident (e.g., smart lock state changed)? Further, the proposed framework would be able to evaluate the different states of connected devices and the overall state of the smart environment. Then, this information can be matched with the security policies in place to detect any (intentional or involuntary) violation of the security policies. Some of these violations could be: workers accessing the office at night time (i.e., careless unauthorized activities), Bob tampering the security camera to avoid video recording (i.e., device tampering), or smart apps running malicious code to modify the values of the presence sensor.

B. Assumptions and Definitions

We make the following realistic assumptions and define the terms to introduce VeritaS's threat model.

- *Assumption 1 – Smart Environment Security Policies:* We assume there is a dedicated security policy (e.g., Enterprise Information Security Policy (EISP)) that regulates physical activities within the smart environment (e.g., unauthorized presence in physical spaces, approved work schedules, etc.). Also, the EISP defines the types of devices and applications approved for use within the corporate environment. We believe this is a realistic assumption because many governments and organizations use EISP's guidelines to define their security posture [44], [45]. For instance, the U.S. government currently works with vendors to define baseline standards for the use of IoT devices and apps in government agencies and smart offices [46]. For VeritaS, we assume that there is a corporate smart environment (devices and apps) that is centrally-managed by well-defined EISPs, and users are not authorized to change its pre-defined settings with their own apps without upper management's approval. For instance, the configuration settings of thermostats and smart locks cannot be changed unless the management office requests and approves the reconfiguration. With this, we expect that the security policies regulate both the access level and settings of the smart environment. Such an assumption imposes a finer granularity to traditional EISPs, as they will need to clearly match, for instance, a specific set of apps with their approved user(s). Note that this will not impact VeritaS's adoption considering its overall benefits to the enterprise security efforts.
- *Definition 1 – Forensically-relevant Data:* It defines the smart apps events, device's states, user-defined inputs, device's information, and time and location information that VeritaS extracts/logs from the smart environment. This information typically reflects on the state of the different smart devices and sensors as a function of the users' and smart apps' activities.
- *Definition 2 – Unauthorized User:* It defines a user that carelessly try to change the pre-defined settings of the smart environment. Also, unauthorized users are those who perform activities and access the smart environment locations during not-permitted hours. Depending on the specific time, date, the context of the activity, and the smart environment policies, authorized users may act in an unauthorized manner.
- *Definition 3 – Attacker (insider or outsider):* We consider the *attacker* as an insider or outsider that maliciously tries to disrupt or take control over the smart environment or any of its entities (i.e., devices and sensors) to learn user behaviors, steal sensitive information, gain access to the systems, or even interrupt the normal operations of VeritaS.
- *Definition 4 – Malicious App:* This is a smart app with malicious code intended to leak sensitive information from users and the smart environment to attackers, change/replace legitimate VeritaS logs with the intention to hide malicious behavior, or perform any other malicious activity inside the smart setup (e.g., side channel attacks).
- *Definition 5 – Tampered Device:* It defines a smart entity that is forced to change its expected state (based on the user activity) by a different one that does not reflect the real overall state of the smart environment. Also, this

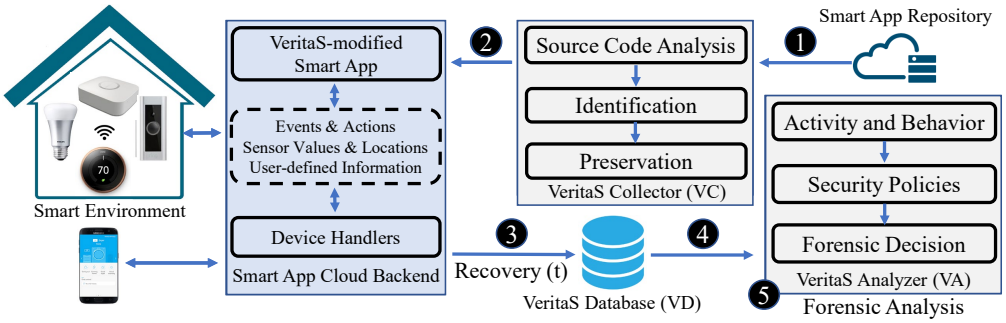


Figure 1: The architecture of the VERITAS. The VC logs smart data to the VD at runtime. Later, the VA analyzes the data and infers activities and forensic behaviors.

group includes devices that are deliberately relocated, disabled, or turned off without the required authorization.

C. Threat Model

We consider two different sets of forensic threats: (1) *anomalous user activities* and (2) *malicious behavior from users, smart apps, and devices*.

Anomalous User Activities. Anomalous user activity includes any careless or unintentional action performed by an *unauthorized user* inside the smart environment that is considered a violation of the security policies in place. For instance, an authorized user trying to access a restricted area during the non-permitted hours. Here, we consider threats related to user activities based on their time dependency; that is, we explicitly consider time-based actions to evaluate VERITAS in detecting anomalous user activities.

Forensic Behavior. We consider forensic behavior as any intentional action performed by users, smart apps, and smart devices that can be considered a threat to the overall state of the smart environment. These forensic behaviors may include *impersonation attacks*, *false data injection attacks*, *replay attacks*, *Denial-of-Service*, and *side channel attacks*.

We provide details of specific activities and behaviors used to evaluate the performance of VERITAS in Section V. For every activity and behavior, we specify the attack method utilized, time dependency, and specific examples in the context of the problem considered by this work. Later, these activities and behaviors are considered to evaluate the efficacy and performance of VERITAS.

IV. VERITAS ARCHITECTURE

Figure 1 depicts the general architecture of VERITAS. First, the user downloads the original smart app source code from one of the freely available online repositories ❶. Then, the VERITAS Collector (VC) automatically analyzes the app and enables the collection of forensically-relevant data to a secure database (VD) ❷. With this, the VC implements the first three stages of a traditional forensic analysis (Section II). Specifically, the VC processes include the *identification* (via source code analysis) of forensically-relevant data within the smart apps that monitor and control the smart environments. Also, the VC supports *preservation* of the smart data via implementing secure collection mechanisms that log the smart data upon app's utilization (i.e., runtime data collection). Finally, the collected data is stored into a secure VERITAS's database where the data is properly labeled and time-stamped for further analysis (i.e., *recovery*) ❸. Later, during the event of a forensic investigation, the Analyzer (VA) performs data processing and applies a First Order Markov Chain model on the VERITAS-collected data ❹. The purpose of this *analysis* is to extract forensically-relevant information from the VERITAS logs. Analyzing this data allows learning the state of the smart environment during specific incidents and provides insights about the users', apps', and devices' actions. Finally, VERITAS correlates these actions with the security policies defined for the smart environment to *uncover* anomalous user activities and potential malicious behaviors from users, smart apps, and devices to the investigator ❺.

A. Forensically-valuable Features in VERITAS

We describe smart app features that potentially contain relevant information for forensic purposes. We consider *events* since they define physical changes in the smart environment setup based on the user's activities (e.g., a door's state was "unlocked" at an unauthorized time). Also, our architecture considers *actions* extracted from smart apps because they potentially contain valuable timing information that can define changes in forensic timelines (e.g., a fire event occurred "after" a door was unlocked). Additionally, we consider *user-defined inputs* in our architecture. Changes in user-defined inputs are critical during forensics analysis because unauthorized modifications directly impact the execution of the apps' events, actions, and notification mechanisms (e.g., the notification recipients of the smoke detector app are modified right before a fire event). Other features considered by VERITAS are *time* and *location* information. These features provide valuable timing information about forensic actions. Also, they prevent reply attacks from malicious users trying to inject fake activity data into the smart environment (e.g., it prevents an attacker from using old benign logs to hide current potentially dangerous activities). Finally, VERITAS tracks smart apps communications to correlate the logged data with information obtained from notifications. For instance, the device state obtained through specific VERITAS logs can be confirmed or disproved by checking the notifications sent by the smart app. This process may help to identify tampered devices or attackers trying to

inject fake device state data into the system to disrupt the normal operations of VERITAS.

B. VERITAS *Collector (VC)*

The VERITAS Collector analyzes the source code of original smart applications to identify forensically-relevant data. Then, it automatically inserts code (i.e., app instrumentation) to enable the secure collection of the data. The app instrumentation process of VERITAS works on the source code of the smart apps. VERITAS takes advantage of the existence of 1000+ open-source apps in the market today. Recent data collected from IoT vendors estimate that these open-source apps control over 4000 different smart devices from most major vendors including SmartThings, openHAB, Windows IoT, and AWS IoT [47]. Considering this, the potential impact of VERITAS is significant, and the approach of using open-source apps makes its contribution realistic. We also envision the adoption of VERITAS in the closed-source market. As we make VERITAS fully open source, closed vendors can use its technology as is, or adapt it to their specific needs, so they can enable full out-of-the-box forensics support in their smart products and take advantage of the benefits offered by VERITAS framework.

Algorithm 1: Steps in the VERITAS Collector (VC).

```
1: appSC ← app source code
   Analysis:
2: AST ← generateAST(appSC)
3: ICFG ← createICFG(AST)
4: if Exists ICFG then
5:     for nodes in ICFG do
6:         forensicPT ← forensic-relevant points
7:     end for
8: end if
   Instrumentation:
9: if forensicPT then
10:    for points in forensicPT do
11:        Insert VERITAS Logs
12:    end for
13: end if
```

1) Identification Stage: Algorithm 1 details the processes performed by the VC to analyze and modify a smart app. During *Analysis*, the app's source code is loaded into the VC client (Line 1). The first step towards analyzing the smart app source code is to create an intermediate representation (IR) of the app and model the application's structure [48]. Different smart home platforms may utilize different programming languages to code their smart apps. For instance, Samsung SmartThings uses Groovy [49] while openHAB apps are written in a Java-based Domain Specific Language (DSL). Thus, the IR facilitates the design of a generalized solution that integrates apps and devices from multiple smart home/office platforms into the VERITAS's analysis. Furthermore, modeling the smart app permits the extraction of smart apps' entry points, events, and control the flow of data. Also, it allows for identifying the data sources and exfiltrations methods (i.e., sink functions) that VERITAS uses to define (1) the source of the forensically-relevant information and (2) how this information is sent out from the apps and its recipients, respectively. The VC extracts an Abstract Syntax Tree (AST) representation of the smart apps for building

Listing 1: A sample code of a real smart app instrumented by the VeritaS Collector.

```
1 /* A section of a code block of a VeritaS−instrumented smart app */
2
3 section("Via a push notification and/or an SMS message") {
4     input("recipients", "contact", title: "Send notifications
          to") {
5         input "phone", "phone", title: "Enter a phone number
              to get SMS", required: false}
6     }
7     sendValue ("New recipient defined: $phone") //VeritaS log
8 }
```

algorithms to find forensically-relevant points inside the source code's IR (Line 2). The advantage of the AST is that considering the simplicity of the smart apps, it permits the extraction of app constructs that are relevant to the forensic analysis like "Events", "Actions", and "User-defined Inputs" (see Section II). Then, the source code analysis starts by constructing an Inter-procedural Control Flow Graph (ICFG) of the apps, and by extracting the specific nodes that define events handlers, actions, and user inputs (Line 3). Once the ICFG is obtained, the VC recursively visits all the ICFG nodes and flags all forensic-relevant distinctiveness in Line 6. These distinctiveness can be summarized as:

App Permissions. In smart apps, permissions include the device information, its capabilities, and the user inputs defined at install time. The VC flags nodes from the ICFG that potentially contain information related to permissions.

Entry Points. Smart apps define multiple entry points (i.e., event handlers) to subscribe to events occurring in the smart environment. Hence, these handlers can be utilized to extract events occurring in the smart environment and the specific device's response to those events (i.e., device actions). Thus, the VC flags the ICFG nodes that define the event handlers to extract the different device's actions from the smart app.

Handler-dependent Methods. Once VERITAS flags the event handlers, the VC visits all the methods invoked by every specific handler. With this, the VC constructs the multiple call graph branches of the ICFG.

Sink Functions. VERITAS flags the sink functions used to send information out of the apps. With this, VERITAS confirms that notifications are sent to the authorized recipients when forensic events occur in the smart environment. Additionally, the sink's information can be used to correlate device states collected via VERITAS logs. For instance, the logs from a smart app controlling a motion sensor can be modified to hide user presence. However, by analyzing the notification received by the user, VERITAS detects malicious behavior if the content from the sink function does not match the recorded log. Also, sink information may reveal unauthorized app settings (e.g., modification of the sink's recipients). Finally, in case of missing sink logs (e.g., notifications never received), VERITAS analysis may reveal devices that were disabled or turned off.

Once the app's relevant data is identified, the VC process in Algorithm 1 concludes. At this point, the *Instrumentation* process of VERITAS customizes the app by inserting the tracing logs (Line 11 in Algorithm 1) to capture the forensic data in real time. Listing 1 shows a sample code of a real

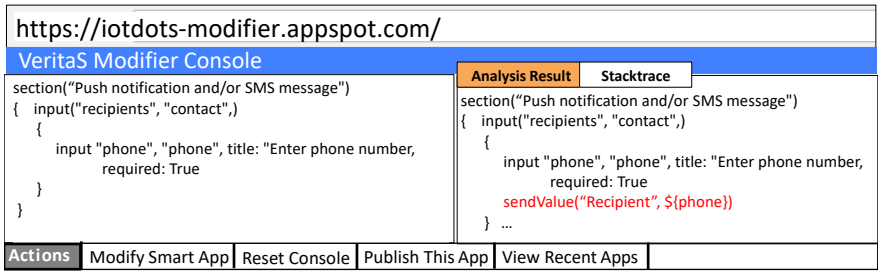


Figure 2: The VERITAS Collector is freely available online.

SmartThings app that have been instrumented by VERITAS. The VC identifies that a new phone number has been set to receive notifications from the app. Then, a `sendValue` function is inserted to collect this specific app distinctiveness in real time.

2) Preservation Stage: VERITAS solves the challenge of securely collecting relevant forensic data from smart devices and sensors via app instrumentation. Specifically, the VC visits the ICFG nodes and detects potentially forensically-relevant data such as user-defined inputs and device actions. Then, the VC automatically insert logs to collect the forensic data. Figure 2 illustrates the instrumentation process of VERITAS. On the left side, the user simply types or paste the original source code of the app that needs to be modified to enable VERITAS. Then, on the right panel, the tool returns the modified app capable of logging all the forensically-relevant data from the app. In this example, the VC flags the user-defined input `recipients`. This specific input defines the phone number to which all the notification from the smart apps will be sent to. As explained earlier, these types of inputs are critical for forensic purposes. Careless or malicious users can modify the smart data anytime which can negatively impact the final purpose of tracking the events and actions executed by apps. Going back to the previous example, after labeling the user input, VERITAS modifies the app by inserting a forensic log. Here, the `sendValue()` function defines an `HTTPS` request that securely pushes the `phone` information to the remote VD. We note that the use of the VC web app does not require technical background or any level of coding experience from the users. The goal of the VC is to automatically analyze the apps and perform the required instrumentation to enable the collection of forensic-relevant data. We made the VERITAS available online at:

`https://iotdots-modifier.appspot.com/`

3) Recovery Stage: During the recovery process, VERITAS must guarantee the authenticity and integrity of the collected data. As explained in Section III-A, we expect network administrators to utilize the VERITAS-modified apps to control devices within their smart environments. With this, they can take advantage of VERITAS's capabilities to support forensic analysis on data extracted from the smart environment. We do not expect individuals using other than the VERITAS-enabled apps within the smart environments. In fact, doing so would be a direct violation of the security policies in places where VERITAS is implemented. To prevent the VERITAS apps from being modified, network administrators may verify that they have installed the correct app version via the use of SHA checksum hashes [50]. While the VERITAS-enable apps are utilized in real time, the apps send forensically-relevant data generated by the smart devices to the secure VERITAS database (VD). We protect the data in motion from apps to the database with the use of encrypted HTTPS calls via Transport Layer Security (TLS). Also, we digitally sign the messages containing the sensitive information with digital certificates and methods approved by the smart platforms (more details in Section V-A). Finally, we consider that the VD is secure and protects the stored data (i.e., data at rest) from being modified/leaked by external entities. Note that attacks to the server or server-related threats are not considered in this work.

C. VERITAS *Analyzer (VA)*

The VERITAS Database (VD) stores data logs obtained from smart apps at runtime using VC. These logs are fed later to a prediction model that VERITAS uses infer activities and forensic behaviors occurring in a smart environment. For successfully extracting forensics information from the logs, VA performs the following operations on the VD data:

Time-stamping. As we consider that the smart environment could be compromised, we cannot trust the temporal information of the collected smart data. Thus, we timestamp every data instance collected by VERITAS upon arrival to the database. We assume that the latency added by VERITAS from the time in which the action occurs to the time in which the log is properly timed and stored has negligible effect in the forensic analysis (see Section V-G).

Labeling. The VA labels the collected data based on timing information, the specific device and app generating and handling the data, and their location.

Inference. For simpler smart environments with a low number of devices (e.g., 5 to 10), data provenance and cross-app analysis of the collected data could provide enough forensic evidence. However, more complex scenarios with a higher number of devices (e.g., enterprise environment with tens or even hundreds of devices) or environments that potentially include tampered devices may require a more in-depth analysis. The VA uses intelligence to analyze the VD-labeled data and infer forensic information from all data logs. Specifically, VERITAS utilizes Markov Chain-based prediction mechanisms, which we describe in Section IV-C2.

Algorithm 2 details the steps executed by the VA to analyze the forensic data. In Line 1, $verLogs$ variable is

initialized with the content of the VD. The initialization step also includes updating the variable $policy$ with the security policies defined for the smart environment at the time in which the logs were collected. Then, in Line 5 the VERITAS logs are organized and labeled based on the timing information, the type of devices that generated the logs, and the location information of the devices. With this information, a prediction model is applied on the data to (1) detect user actions in the smart environment (Line 9) and (2) detect behaviour of users and smart apps, all depending on the security policies established at the time the logs were acquired (Line 10). Finally, if a forensic violation is detected, the flag *anomaly* is set to TRUE.

Algorithm 2: Steps in the VERITAS Analyzer (VA).

```
1: verLogs ← VD
2: policy ← smart environment security policies
3: anomaly ← FALSE
   Labeling:
4: for each Log in verLogs do
5:     label Log by Time, Device, Location
6:     MLdata ← labels
7: end for
   Detection:
8: for data in MLdata do
9:     Evaluates user activity userAct
10:    ML ← MLanalyzer(userAct, policy)
11:    if Anomaly detected in ML then
12:        anomaly ← TRUE
13:    end if
14: end for
```

In the following, we describe the characteristics of the data collected and the analysis techniques used by VERITAS to identify activities and behaviors in a smart environment.

1) Data Characterization in VERITAS*:* As noted earlier, the VC collects and stores data from a smart environment in a secure VERITAS Database (VD). The data includes timing information, sensor information, device state, location, and log's timestamps. VERITAS collects such a diverse dataset from different sources. For instance, some data may be collected via instrumenting the app functions that directly handle data received from devices and sensors (e.g., device states). And, some other data may be collected via instrumenting portions of the apps that contain information defined by users or developers (e.g., location, device settings) [42].

For a specific time slot t, the collected data can be represented by:

$$Data\ array, E_t = \{T, S, D, M, L, t\}, \tag{1}$$

where T represents the timing features, S represents the set of sensors' features, D is the set of device features, M are the features extracted from the controlling devices (smartphone/tablet), L is the set of location features of the controlling devices, and t represents the VERITAS log's timestamp information. We describe the characteristics of these features below.

Timing features (T). A smart environment consists of several sensors and devices that change their states based on different user activities and controlling commands. In this context, some devices perform instantaneous tasks (e.g., switching lights on and off based on motion) while some devices perform a task over certain period of time (e.g., controlling the room temperature). VERITAS considers this timing information as a feature to infer the overall state of the smart environment at a specific time.

Sensor features (S). Sensors in a smart environment trigger different actions. A smart environment may contain several different sensors (e.g., motion sensor, temperature sensor, presence sensor) attached to multiple devices. These sensors sense the changes in devices' proximity and help the devices to take autonomous decisions such as switching lights on motion or triggering fire alarm after smoke is detected. Depending on the nature of the sensors, sensor data can be both logical (active/inactive) or numerical values. For VERITAS, we collect both numerical values and logical states of the sensors and create the state of the smart environment at a specific time. We represent the change in both logical states and a numerical value of a sensor as a binary input (1 if active/change and 0 otherwise) to create a forensic data matrix to train the detection algorithm (see Section IV-C3).

Device features (D). A smart environment supports different devices that may be or may be not connected to a smart hub (Section II), to different sensors, or other devices. These devices can perform multiple tasks as standalone (e.g., smart thermostat) or as ad-hoc entities (e.g., automatic door controlled by smart lock, camera, and presence sensor). For different user activities and input commands, these devices change their states (active/inactive) autonomously. VERITAS considers the device state data as part of its analysis.

Controller device features (M). In a smart environment, users can use smartphones or tablets to control devices (i.e, controller devices) from the associated smart apps. VERITAS collects the control command generated from controller devices to understand the smart environment settings and the user intended operations in the smart environment at any moment.

Location features (L). The smart environment allows controlling the connected devices from both inside and outside of a specific location. Location information from where a given command was executed has very high forensic value, and is critical to infer illegal activities. VERITAS considers the location of both the controller devices and the smart devices as a feature to understand activities occurring in the smart environment. We use app instrumentation to collect data, such as location, that is not directly provided by smart devices. Specifically, location information is part of the app's permission block and is often defined by the user or the app's developer. The permission block of an app defines device-related settings (e.g., location, value ranges, and notification recipients). As noted in Section IV-B, we included the app permissions block as part of the forensically-relevant information that VERITAS collects.

Timestamps (t). VERITAS considers the data timestamps to accurately infer the exact date and time of occurrence of the events. The data instances of type timestamps define the time interval in which VERITAS predicts the activities and forensic events within the smart environment. As explained in Section IV-C2, these intervals are minimized by the use

of a prediction model of first order. Hence, timestamps represent the prediction instances with high precision. Additionally, the use of timestamps prevents adversaries from performing reply attacks on the VERITAS's data logs.

2) The Markov Chain Model: VERITAS uses a First Order Markov Chain model [51] to understand the chain of events occurring in a smart environment over time and identify any unexpected or malicious incident that violates the security policies of an organization. The high statistical correlation between different smart environment states supports the use of Markov Chain approaches in VERITAS. In most cases, a smart environment contains smart devices with similar sensors that generate a highly redundant dataset. Also, as we detailed in previous sections, the extraction of Markov states from the environment constitute a very straightforward process via smart app instrumentation.

We use the Discrete-Time Markov Chain library written in Python, `PyDTMC` [52], to implement VERITAS's Markov Chain model. This library offers the flexibility of choosing a specific number of dependent states, which is very helpful to infer quasi-instantaneous activities (e.g., opening a door). Also, PyDTMC offers automatic feature extraction from datasets, which is suitable for smart environments with a diverse set of devices and also makes VERITAS easily scalable.

Benefits of a First Order Markov Chain Model. Note that the benefits of implementing a Markov Chain model of first order to support the VERITAS's analysis are two-fold. First, a Markov Chain stochastic model permits using device state information collected from the smart apps to create an array of Markov states. These states characterize the behavior of entities (users, devices, etc.) within the smart environment. We apply this stochastic model to the set of device states to predict the overall future state of the environment. Then, based on the probability distribution of the Markov states, the model infers the expected state of every device and identify violations of the security policies within the environment even in the presence of tampered devices (see Section IV-D) [53]. Second, because we use a first order model, exactly the previous (t_{-1}) Markov State is considered to infer the current (t_0) overall state of the environment (the model considers the previous dependent state). This makes the VERITAS's analysis highly adaptable and resilient to changes in the smart environment configuration (e.g., add/remove devices, change the device settings, etc.). To explain this, assume that an organization wants to change the position of some presence sensors. This modification impacts the overall state of the environment as the new positions of the sensors change the statistical correlation between device states. We call this interval a *transition time* in which VERITAS does not depend on backwards data of previous configurations of the environment to successfully predict the new state. Instead, since VERITAS uses a first order model, its analyzer can quickly adapt to the new environment state, and start predicting the new Markov states with the fresh data instantly, minimizing the duration of the transition time and hence the time interval in which VERITAS can predict with high accuracy. For n^{th} order models, VERITAS would need to collect several states instances before being able to perform an accurate prediction. In general, the use of a first order model guarantees the shortest time frame in which VERITAS can perform a valid prediction. Finally, the use of a First Order Markov Model aligns with the characteristics of the smart devices and sensors whose states do not propagate over time, but are mostly dependent on the previous state (e.g., the light is now on, was off before).

3) Markov Chain States in VERITAS*:* In addition to considering the previous state to perform the prediction, we use binary Markov states to characterize a smart environment. The rationale behind such a design decision is supported by the fact that most smart sensors and devices have binary outputs (e.g., door open, presence, light on, light off, etc.). In fact, only a few cases (e.g., temperature) provide outputs of type numerical data. Although numeric sensor values may provide additional cases of analysis and finer granularity, only those values that lead to a forensic case (e.g., temperature value over certain threshold that violates a security policy) are of real interest of VERITAS. Also, multi-level Markov states are mostly valuable for implementing real-time monitoring systems, which is not the purpose of the VERITAS framework. Hence, we convert numerical device data into binary states and keep the Markov Chain model simpler and homogeneous.

For this, we implemented a *binarizer* to convert numerical data into binary values that can be directly interpreted by the Markov Chain model. We found that for the types of devices and sensors utilized in our evaluation, only a few cases of sensors and devices outputs explicitly required *binarization*. In these specific cases, we correlated user-defined settings extracted from the smart apps with the security policies (i.e., EISPs, see Section III-B) to define threshold values that permit VERITAS determine cases with potential forensic interest. For instance, for a temperature sensor, VERITAS may log as "active" (e.g., "1") any temperature values exceeding the upper bound limit set by the EISPs, and "inactive" (e.g., "0") otherwise. As we combine user-defined settings and EISPs to binarize numerical values, we minimize the occurrence of potential quantification errors due to wrongly determining the "active" and "inactive" state of the multi-state devices.

4) Analytical Approach used in VERITAS*:* VERITAS correlates Markov states derived from the data collected from similar sensors in one location and at any given time to (1) determine the validity of the state, (2) infer if one specific state was or not compromised, and (3) determine the probability that specific events and actions occurring are either anomalous or malicious. The VA collects the smart environment data and creates a binary state array (1 for active status and 0 for inactive status) to represent the state of the smart environment at any specific time t. Thus, we represent the state of the smart environment as a n-bit binary number, where n is the number of features extracted from the logs. Thus, the total number of possible states of a smart environment with n number of features (sensors' states, devices' states, controller devices, and locations) would be $m = 2^n$. VERITAS utilizes the timed binary state of the smart environment to train a First Order Markov Chain model to detect forensically-valuable behavior from users,

smart apps, and devices. The Markov Chain model benefits from two main assumptions: (1) the occurrence probability of a specific state s_i only depends on the previous state s_{i-1} and (2) the transition between two consecutive states is independent of time and does not depend on any previous condition. Based on these assumptions, we represent the Markov Chain model with the following equation [54].

$$\begin{aligned} P(X_{t+1} = x | X_1 = x_1, X_2 = x_2 ..., X_t = x_t) = \\ P(X_{t+1} = x | X_t = x_t), \\ when,\ P(X_1 = x_1, X_2 = x_2 ..., X_t = x_t) > 0, \end{aligned} \tag{2}$$

where X_t and X_{t+1} denotes state of the smart environment at time t and $t+1$, respectively. Lets assume the smart environment has the state i at time t and j at time $t+1$. If P_{ij} illustrates the transition probability between state i to j, the state transition matrix for m number of states of a smart environment can be represented by the following matrix.

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \dots & \dots & P_{1m} \\ P_{21} & P_{22} & P_{23} & \dots & \dots & P_{2m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{m1} & P_{m2} & P_{m3} & \dots & \dots & P_{mm} \end{bmatrix}, \tag{3}$$

To calculate each element of the transition matrix, we assume that the smart environment has $X_0, X_1, \dots, X_T$ states at a given time $t = 0, 1, \dots, T$. Then, each element of the transition matrix can be represented by the following equation [54]:

$$P_{ij} = \frac{N_{ij}}{N_i}, \tag{4}$$

where N_{ij} is the total number of transition from X_t to X_{t+1} over a certain period. From the state transition matrix, the Markov Chain model calculates the probability of occurring a state or sequence of states. To predict the probability of occurring a state, we assume the initial probability distribution of the Markov Chain model as follows:

$$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \dots & \dots & q_m \end{bmatrix}, \tag{5}$$

where, q_m is the probability that the model is in state *m* at time 0. The probability of observing a sequence of states can be calculated by the following equation:

$$P(X_1, X_2, \dots, X_T) = q_{x1} \prod_{2}^{T} P_{X_{t-1} X t}. \tag{6}$$

D. State Inference via Device Cooperation

VERITAS infers tampered devices based on the analysis of collected logs from multiple devices. That is, the Markov Chain model analyses the state of all the devices in the smart environment at any given time and detects malicious or unexpected states by comparing data from similar devices that share similar contexts (e.g., nearby locations). We call this process *device cooperation*. During device cooperation analysis, if one device is compromised or tampered, the information collected from other trusted devices inside the environment is used to detect the one reporting fake or unexpected data. For instance, consider a smart light controlled by a motion sensor. During normal operations, if the sensor detects any motion, the light is turned on. However, in the event of a compromised state of the smart light, the light may not operate appropriately nor follow the motion sensor states. In this case, a third device (e.g., a smart thermostat that also includes a presence sensor) can act as an *adjudicator authority* to decide which device is misreporting states. If both the motion sensor and the thermostat achieve similar states, then the smart light is deemed as tampered. On the other hand, if the smart light successfully follows the smart thermostat actions, the motion sensor is flagged as tampered. In summary, our classification model relies on third-party entities to resolve conflicts between device states inside the smart environment.

E. Suitability of the Markov Chain Approach

VERITAS is not a real-time prediction tool, but a forensic framework that analyzes data after the events have occurred. Since VERITAS does not infer activities and events occurring in real time, organizations have the flexibility of adding/removing devices, changing the smart environment settings and configurations (e.g., new maximum allowed thermostat temperature value), and upgrading/adding new security policies (e.g., additional presence restrictions for users) without the need of re-training a pre-built Markov Chain model. We demonstrate in Section V that the amount of data collected from a completely new environment over a period of seven days is enough to infer activities and behaviors with high accuracy with VERITAS. Also, as explained in Section IV-C2, the characteristics of the data collected and the type of prediction model used impact the time interval in which events or activities can be predicted with VERITAS. A smart environment contains several devices with similar sensors, hence, the data collected tends to be highly redundant. Data redundancy and the use of a stochastic First Order Markov Chain model to correlate the data permits learning the Markov states of the environment with finer granularity and with the least amount of data. Therefore, VERITAS is capable of capturing all the events and activities occurring in the smart environment regardless of the frequency or time intervals in which they occur.

F. Chain of Custody in VERITAS

As VERITAS supports traditional forensics, it must abide to strict *Chain of Custody* (CoC) rules. However, as VERITAS introduces automation, the CoC rules has to be implemented across the different VERITAS architectural stages. For instance, as explained in Section IV-B2, network administrators start the CoC process by using the right forensics-enabled application generated by VERITAS via the web application (Section IV-B). With this, they guarantee that the logging process integrated to the apps does not maliciously modify the forensic data during the preservation stage. Further, we also integrate the authenticity and integrity of the collected smart data by protecting the transfer process between apps and the VD residing in the VERITAS server. As noted in Section IV-B3, we use encrypted `HTTPS` calls and digital signatures to (1) avoid passive observers from having access to the sensitive data during transmission [43] and (2) verify that the forensic data have not been modified before analysis. In cases where the digital certificates can not be verified, the data is rejected for analysis. We also assume the VERITAS server secure so the classification process of

Device Type	Model	Count
Smart Home Hub	Samsung SamrtThings Hub	1
Smart Light	Philips Hue Light Bulb	5
Smart Lock	Yale B1L Lock with Z-Wave Push Button Deadbolt	2
Fire Alarm	First Alert 2-in-1 Z-Wave Smoke Detector and Carbon Monoxide Alarm	1
Smart Monitoring System	Arlo by NETGEAR Security System	2
Smart Thermostat	Ecobee 4 Smart Thermostat	1
Motion Sensor Light Sensor Temperature Sensor	Fibaro FGMS-001 ZW5 Motion Sensor with Z-Wave Plus Multi-sensor	6
Door Sensor	Samsung Multipurpose Sensor	4
Total		**22**

Table I: List of smart devices and sensors used during the data collection stage in VERITAS's evaluation.

VERITAS and its results are not compromised. In addition, we carefully date and time (i.e., labeling) every collected log so we can follow a chronological order of the collected data. As we consider that the smart environment could be compromised, we add the timelines to the data upon arrival to the VD. We use the same chronological information to establish temporal dependencies (i.e., data provenance) between events and actions generated withing the smart setting during the Markov Chain analysis.

V. PERFORMANCE EVALUATION

With the performance evaluation of VERITAS, we aim to answer the following research questions:

- **RQ1: Anomalous Activities.** What is the performance of VERITAS in inferring anomalous activities within a smart environment? (Section V-C).
- **RQ2: Forensic Behavior from Users.** Can VERITAS effectively detect forensic behaviors from the users that explicitly violate the corporate security policies? Can the VERITAS analysis detect, via device cooperation, the presence of tampered devices within the smart environment? (Section V-D).
- **RQ3: Forensic Behavior from Devices and Apps.** What is the performance of VERITAS in detecting forensic behaviors from malicious apps? (Section V-E)
- **RQ4: Overhead.** What is the overall overhead introduced by VERITAS in terms of physical storage and latency? (Section V-G).

Note that we obtained Institutional Review Board (IRB) approval to test VERITAS in real scenarios. We implemented a smart office environment with specific security rules enforced in the system (e.g., time-restricted location access for users and restricted device re-configuration) where real users performed regular activities. Further, we consider a group of users that carelessly violate the security rules of the environment by performing anomalous activities such as accessing office locations at an unauthorized time or changing the configuration and topology of the smart environment. Finally, we considered specific forensic behavior like poisoning data from a specific number of devices using malicious apps. We built the Markov Chain-based detection method and trained our model with data collected by VERITAS from the real-life smart office setup.

A. VERITAS *Implementation*

The current implementation of VERITAS provides solutions for the Samsung SmartThings platform. With this, we focus on a smart platform that (1) defines the highest amount of different smart devices and apps in the market [55]; (2) it is open-source, so the apps' source code is freely available online; and (3) it provides extensive documentation to developers [39]. To protect the communication between the SmartThings apps and the secure VD, we used the *asynchttpv1* class of Samsung SmartThings [39], which allows for asynchronous and encrypted `HTTPS` calls. Also, we used the SmartThings x.509 certificates to verify integrity of the data. Specifically, we fetched the public keys generated for every app from the `https://key.smartthings.com + <aKeyId>` account, which is assigned the first time the VERITAS-enabled app is uploaded to the Samsung SmartThings account online.

1) Training Environment Setup: For training purposes, we built a real-life smart office environment with 22 different and popular smart devices and their apps available in the SmartThings App Market (Table I). Then, we designed generic forensically-valuable activities that the users typically perform in most smart environments. The set of considered activities included the following scenarios:

Time-dependent access. We allowed a single user to access all the office locations during scheduled hours and observed her activity patterns inside the smart environment.

Restricted access. We repeated the previous activities while enforcing restricted access policies for specific locations in the smart environment.

Multi-user environment. We combined the two previous scenarios with up to ten users interacting within the smart setting, which emulated more realistic setups.

Reconfiguration rule. We allowed configuration of the devices and apps in a timely and supervised manner only. Any unauthorized reconfiguration is considered anomalous. We provided time-specific rules to collect data while reconfiguring the smart environment.

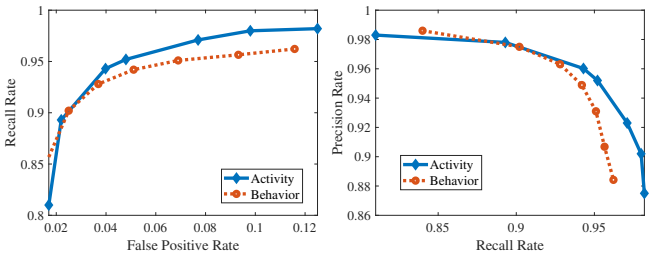
2) Data Collection: During data collection, we used the VC available online to automatically instrument the SmartThing apps and insert the forensic log code. Then, while utilizing the apps in real setups, the VERITAS-collected data was sent to the VERITAS datbase (VD). For evaluation purposes, we categorized the user activities as *time-independent* and *time-dependent*. In the first group, we included all the user actions whose execution times were irrelevant to VERITAS. The second category grouped actions whose execution times were strictly regulated by the security policies of the smart environment (e.g., the presence of a user in a restricted area would constitute a violation between 8:00 pm and 7:00 am). We explicitly considered scenarios where 10 different authorized users were freely performing regular smart office activities within a smart environment consisting of 22 smart devices and sensors (Table I) for seven days. The

Threat	Time Dependency	Attack Method	Specific Attack Example
Activity-1	TI	Tampered device	Bob changes the orientation of the presence sensor to fit in a new equipment.
Activity-2	TI	Careless unauthorized user	Bob manually increases the temperature of smart thermostat from home, the night before of an important meeting with the stakeholders
Activity-3	TD	Careless unauthorized user	Bob is inside a restricted area (servers room) at 8:45 pm.
Activity-4	TD	Careless unauthorized user	Bob is getting into the office at 8:45 pm.
Activity-5	TD	Careless unauthorized user	Bob leaves the smart door unlocked.
Behavior-1	TI	Tampered device	Bob disables the smart camera to stop recording while he is performing unauthorized activities inside the office O.
Behavior-2	TI	Impersonation Attack	Alice gets access to the office O using the smart lock pin that she obtained through a malicious app that leaks information.
Behavior-3	TI	False Data Injection Reply Attack	The smart fire detector reports inverted fire alarm states to VERITAS.
Behavior-4	TI	Denial-of-Service Attack	The smart thermostat is shut down via a malicious app.
Behavior-5	TI	Side Channel Attack	The smart light app disable the compromised smart camera by creating a specific light on/off pattern.

Table II: We used these specific activities and behaviors to evaluate the efficacy of VERITAS.

activities and behaviors included in our evaluation were not unique to our testing environment, but represented actions that users can perform in any similar smart office scenario (e.g., opening a door). Thus, our evaluation results can be generalized to other smart environments with similar types of devices, sensors, and apps.

We first asked the users to perform their regular office activities while strictly avoiding violating the security policies. Then, for the anomalous activities, we asked the users to implicitly violate the security policies defined for the smart environment. Specifically, the users performed five specific anomalous types of activities. These anomalous activities were related to: (*Activity-1*) careless tampering of smart devices (e.g., relocation, disabling), (*Activity-2*) negligent modification of device settings (e.g., temperature values), (*Activity-3*) inadvertent user presence into restricted office locations, (*Activity-4*) careless and unauthorized access to restricted locations, and (*Activity-5*) negligent and unauthorized use of smart office devices. We further collected data related to forensically-relevant behavior from the users, smart apps, and devices. Forensic behaviors from users' actions considered a user that tries to modify, tamper, or remove the devices in the smart environment. We define a *Behavior-1* intended to explicitly and maliciously change the original configuration of the smart setting to prevent VERITAS from sending incriminating logs to the VD. We allowed the users to tamper (e.g., disable, relocate) and remove smart devices to recreate this scenario. On the other hand, for forensic behaviors related to the smart apps, we installed the modified version of malicious smart apps collected from the IoTBench [56] project. We defined four additional behaviors as follows. For *Behavior-2*, we created two different apps to control a smart lock that leak the secure pin to an attacker. For *Behavior-3*, we built an app that injects forged data in a fire alarm and triggers the alarm maliciously. For *Behavior-4*, we developed an app that can power down the smart thermostat after setting a specific input temperature, creating a Denial-of-Service (DoS) situation. Finally, for *Behavior-5*, we created an app through which a specific light pattern can maliciously trigger the smart camera. To collect the



(a) ROC (b) PRC

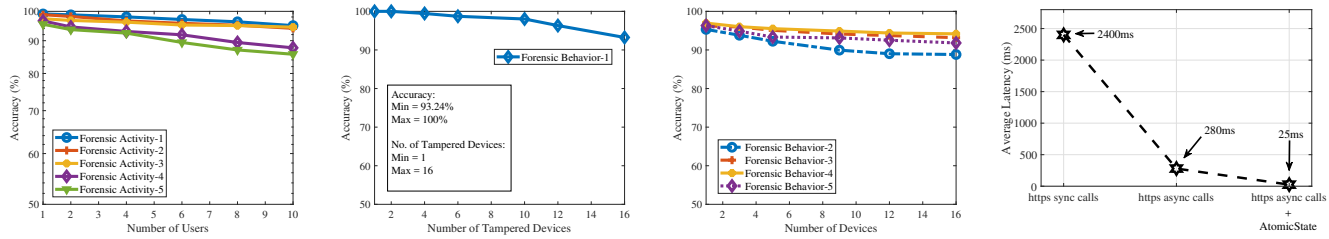
Figure 3: Impact of data imbalance in VERITAS.

behavior-related data, we performed the attack scenarios multiple times. We also considered multi-attacker scenarios where multiple attackers perform different attacks at once.

We provide details of the specific anomalous activities and forensic behaviors considered in our implementation and evaluation of VERITAS in Table II. For every activity and behavior, we specify the attack method utilized, the time dependency, and provide specific examples in the context of the problem considered by this work. These activities and behaviors were utilized to test the efficacy and performance of VERITAS in realistic smart office environments.

Our regular user activity dataset comprised 84209 forensically-valuable incidents. We also collected 4500 instances of anomalous activities and 7500 data instances corresponding to forensic behaviors. The data collected from regular user activities represented the 87.5% of the total data. Meanwhile, the 7.8% and the 4.7% of the data were obtained from forensic behaviors and anomalous activities, respectively. We applied some techniques to avoid data imbalance and over-fitting issues in our evaluation. First, we under-sampled the set of regular activity data until we obtained a new evaluation dataset with a proportion of 50% of the data corresponding to regular activities and 50% of the data corresponding to both anomalous activities and forensic behaviors (50-50 rule). We used 75% of the new dataset to train the Markov Chain model and the remaining 25% to evaluate its performance, which is a typical data split used in other research studies [42], [54], [57]. Finally, we used k-fold cross validation to generalize our Markov model and avoid overfitting of specific activities or behaviors during training.

3) Model Training and Optimization: To determine the effectiveness of our data imbalance correction process, we considered both the Receiver Operation Characteristics (ROC) and Precision-Recall Curve (PRC) metrics. We used these metrics to evaluate the base-rate fallacy of the training data [58] after applying the techniques to avoid data imbalance and over fitting issues. Figures 3(a) and 3(b) illustrate the ROC and PRC for user activities and forensic behaviors. We used the `trapz` function in `MATLAB` to calculate the area under the curve for both Figures 3(a) and 3(b). The ROC curves resulted in an average area under the curve of 85.1%, indicating very low sensitivity to data imbalance in VERITAS [59]. The PRC evaluation achieved an area under the curve of 82.7%, indicating excellent performance of VERITAS under imbalanced dataset [60].



(a) User Activities (b) Tampered Devices (c) Forensic Behaviors (d) Overhead

Figure 4: Performance of VERITAS in: (a) inferring forensic activities in multi-user scenarios, (b) detecting forensic behavior related to tampered smart devices, and (c) detecting forensic behavior of compromised smart apps. Finally, (d) details the average latency imposed by VERITAS to smart apps' execution times.

B. Performance Metrics

We chose six different performance metrics to evaluate the effectiveness of VERITAS: True Positive rate or Recall rate (TPR), False Negative rate (FNR), True Negative rate or Specificity (TNR), False Positive rate (FPR), Accuracy, and F-score. We define success and failure as follow:

- *True Positive (TP):* Refers to the total number of correctly identified regular activities.
- *True Negative (TN):* Specifies the number of correctly detected anomalous activities or forensic behaviors.
- *False Positive (FP):* Defines the number of instances when an anomalous activity or a forensic behavior is deemed as a regular activity.
- *False Negative (FN):* States the number of regular activities that are categorized as anomalous activities or forensic behaviors.

C. Inferring User Activities

The state of the interconnected devices inside the smart environment depends on the on-going user activities. For example, while a user moves from one place to another, several devices and sensors become active. The changes in device states can be an instantaneous event (a specific event at a specific time such as switching on a light) or a combination of subsequent events over a certain period (motion from one place to another).

Table III shows the performance of VERITAS while inferring user activities. One can observe that, for time-independent activities (i.e., Activity-1 and Activity-2), VERITAS achieved accuracy (i.e., ACC) and F-score values of over 98% and 96%, respectively. True Positive Rate (TPR) and True Negative Rate (TNR) are also high (over 99% and 94%, respectively). On the other hand, for time-dependent activities (i.e., Activity-3, Activity-4, and Activity-5), VERITAS achieved accuracy and F-score values of over 95% and 91%, respectively. In summary, VERITAS obtained average accuracy values of over 95% for detecting different forensically-relevant user activities.

In the case of a multi-user smart environment, the fact that the users may perform different tasks at once directly impacted the accuracy of VERITAS. Figure 4(a) shows the accuracy of VERITAS in inferring user activities in multi-user scenarios. As explained, one can observe how the accuracy values decreased with the number of users. For time-independent activities (i.e., Activity-1 and Activity-2), VERITAS achieved accuracy in the range of 98% to 95%. For time-dependent activities (i.e., Activity-3, Activity-4, and Activity-5), the accuracy of VERITAS varied from 96% to 86% as the number of users increased.

User Activity	TPR	FNR	TNR	FPR	ACC	F-Score
Activity-1	0.9926	0.0074	0.9562	0.0438	0.9907	0.9740
Activity-2	0.9904	0.0096	0.9438	0.0562	0.9880	0.9665
Activity-3	0.9860	0.0140	0.8623	0.1377	0.9739	0.9197
Activity-4	0.9721	0.0279	0.8614	0.1386	0.9664	0.9133
Activity-5	0.9584	0.0416	0.8861	0.1139	0.9547	0.9208

Table III: Performance of VERITAS in inferring user activities.

D. Forensic Behaviors from Users

Devices installed in a smart environment can be vulnerable to tampering, which can lead to malicious events. Tampered devices impact VERITAS performance as the framework relies on legitimate Markov States to infer actions and activities occurring in the smart environment. Although VERITAS is not a tool to detect compromised devices, it is possible to infer device tampering by analyzing the data collected from similar sensors and devices in a location. For instance, if a motion sensor is tampered with, the data collected from that sensor would contradict states from similar devices at the same location. Figure 4(b) depicts the accuracy of VERITAS in detecting tampered devices in a smart environment. One can observe that VERITAS achieved near 100% of accuracy in cases with 2 tampered devices in the environment. As expected, the accuracy of VERITAS decreased as the number of tampered devices increased in the system. However, even with a maximum number of 16 tampered devices, VERITAS performed very well with over 90% accuracy.

E. Forensic Behaviors from Devices and Apps

Smart home/office platforms offer customized apps to control smart devices. In recent years, researchers have reported several malicious apps that can perform malicious activities in the smart environments [48], [55], [56]. We evaluated the efficacy of VERITAS in detecting behaviors in

Behavioral Model	TPR	FNR	TNR	FPR	ACC	F-Score
Behavior-2	0.9612	0.0388	0.8652	0.1348	0.9533	0.9106
Behavior-3	0.9651	0.0349	0.9289	0.0711	0.9621	0.9466
Behavior-4	0.9730	0.0270	0.9317	0.0683	0.9696	0.9518
Behavior-5	0.9687	0.0313	0.9012	0.0988	0.9631	0.9336

Table IV: Performance of VERITAS in detecting forensic behaviors from smart apps.

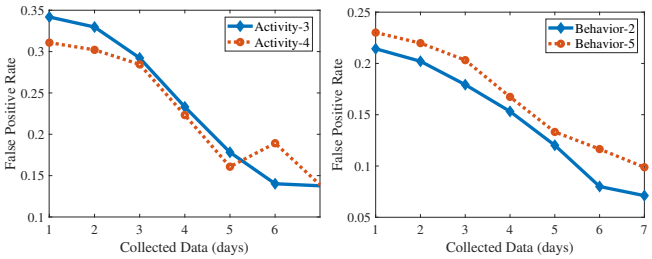
a smart environment caused by malicious apps installed in the system. As noted earlier, we considered four different scenarios to evaluate app behavior in VERITAS (Section III). To evaluate these scenarios, we installed malicious VERITAS-modified apps in a real-life smart environment (smart office). Table IV shows the evaluation results of VERITAS in detecting the app's behavior in the smart environment. One can observe that VERITAS achieved accuracy and F-score for the cases above of 95% and 91%, respectively.

Figure 4(c) details the accuracy of VERITAS for different forensic behaviors when the number of devices being controlled by malicious apps increases in the system. VERITAS achieved the highest accuracy of 97% for Behavior-4 and the lowest accuracy of 95% for Behavior-2, for the case of only one device in the system. As the number of devices increased, the accuracy decreased to 95% and 89% for Behavior-4 and Behavior-2, respectively. The accuracy of detecting Behavior-3 and Behavior-5 varied between 96% to 94% and 96% to 92%, respectively.

F. False Positive Rate Evaluation

We observed some cases of relatively high FPR and FNR during our evaluation of time-dependent activities (i.e., Activity-3, Activity-4, and Activity-5) in Table III and forensic behaviors in Table IV (Behavior-2 and Behavior-5). Although 22 devices is a very good representative setup of realistic smart environments, we attribute these results to (1) the number of devices and sensors used during our evaluation and (2) the amount of data collected. We observed that the inference of time-dependent activities such as the user motion were highly dependent with the amount of data used during training in VERITAS. We noticed that VERITAS had more difficulties to infer user motion as different motion patterns from users during different days of the week caused the Markov Chain states to substantially differ.

We investigated the activities and behaviors with the highest FPR (i.e., Activity-3, Activity-4, Behavior-2, and Behavior-5) to verify the root cause. Specifically, we re-trained the Markov Chain model using the cumulative amount of data corresponding to those specific activities and behaviors and observed the FPR performance over time. Figure 5(a) and 5(b) depicts the FPR against training data for selected activities and behaviors. For all four activities and behaviors, we can observe that the FPR decreased with the number of training days. However, in Figure 5(a), the FPR suddenly increased in the 6^{th} day for Activity-4. We manually checked and found that this was caused by the differences between activities performed during weekdays



(a) FPR vs. Training (Activities) (b) FPR vs. Training (Behaviors)

Figure 5: Impact of the training data on VERITAS's FPR.

and weekend schedules (Activity-4 depends on user presence and time). The FPR decreased again after we trained VERITAS with data from the 7^{th} day. We can conclude that the current performance of VERITAS can be further improved with additional training data. Since VERITAS is a forensic framework that uses a first order model and not a real-time analysis tool, we can realistically assume that the cumulative amount of collected evidence over time in real scenarios would be enough to improve these preliminary results easily.

G. Overhead Analysis

We evaluate overhead based on (1) the amount of physical storage occupied by the VERITAS-modified apps in the cloud servers and (2) the extra latency imposed on the apps.

Physical Storage. On average, the VC added 110 new lines of code to the original smart apps, which represents an increase of around 5KB of physical memory space per app. Additionally, we evaluated the physical memory space used to store VERITAS's logs into the VD. We acquired data from three different types of experiments: regular user activities, anomalous activities, and forensic behavior from users and apps. In total, the datasets from 7 days of activities occupied a total size of 20.55 MB of memory.

Latency. We define *latency* as the additional delay imposed to the VERITAS-modified apps compared to the execution time of original apps. In general, the latency depends on (1) the number of times that the VERITAS sends logs to the VD and (2) the VERITAS logging function's execution time. The higher latency (2.4 sec) was observed when VERITAS used synchronous `HTTPS` requests every time a log was sent to the VD. To reduce latency, we implemented asynchronous `HTTPS` calls using the $asynchttp\_v1$ API in Samsung SmartThings [39]. Further, we utilized the `AtomicState` variable in SmartThings to queue and map several logs together before communicating with the VD. Finally, with all these modifications, the average latency overhead imposed by VERITAS decreased to around 30ms for every `HTTPS` call sent to the VD (Figure 4(d)).

VI. DISCUSSION

A. Privacy and Ethic Considerations of VERITAS

VERITAS has access and stores sensitive information. Thus, we apply well-known data protection schemes (i.e., restrictive access control, encryption) to secure the collected logs at rest, in motion, and in use to prevent adversaries from

compromising the data. In addition, the VERITAS's analysis does not make any effort to fingerprint user behavior, nor collect Protected Personal Information (PPI), nor implement mechanisms that can be used for user surveillance. VERITAS is not a surveillance tool nor a tool to monitor the users in real time. Instead, VERITAS is intended to be used in highly critical environments, where the users performs duties related to their work environment only and not activities related with their personal lives.

B. Security Analysis of VERITAS

Assume there is an attacker, Mallory, trying to bypass the VERITAS analysis or modify its results to her benefits.

False Data Injection. Mallory may have the capability to inject false data into the VERITAS Database (VD) and skew the forensic analysis results. VERITAS uses digitally signed requests (Section IV-F and V-A) to send data from the instrumented apps to the VD while protecting the integrity of the data and preventing external attackers from injecting false data. As an example, Samsung SmartThings provides mechanisms to digitally sign `HTTPS` requests following the IETF HTTP Signatures Draft 3 specification. The legitimacy of the logs sent via SmartThings APIs can be verified via digital signatures, minimizing the impact of the false data injection attacks in VERITAS [39], [61].

Unauthorized App Instrumentation. Mallory may use the freely available Collector to instrument compromised apps and use those app to control the smart environment. To do so, Mallory would need to compromise the credential information of the Samsung SmartThings account under which the legitimate instrumented apps and devices are registered. VERITAS relies on the security measures implemented by SmartThings to protect the user credentials. These measures include the use of encryption to protect the credential information of users, hashed passwords, and Multi-Factor Authentication (MFA) [62]. Also, inside the SmartThings account, corporate administrators may verify that they are using the allowed version of the apps via the use of SHA checksum hashes [50] (Section IV-B3).

Tampering Devices or Inserting Compromised Devices. Mallory tries to tamper the devices and sensors in the smart environment to modify the forensic data collected and change the outcome of the forensic analysis (e.g., disable a presence sensor). In a similar attack, she may try to insert compromised devices that generate fake data. Our evaluation results in Section V-D demonstrate that VERITAS is resilient and resistant to this type of attack via the use of device cooperation. We note that device cooperation is effective when the environment is redundant enough so the tampered data can be compensated with information collected from similar devices and sensors. As the number of tampered devices increase, the accuracy of VERITAS decreases, hence, Mallory may succeed in this attack if she can successfully tamper a high number of devices. Device cooperation does not stop the attack but makes VERITAS more resistant to it and also makes it more difficult for Mallory to implement the attack. Our evaluation results in Section V-D demonstrate that VERITAS can achieve over 90% accuracy even when the number of compromised devices represent over 50% of the total number of devices in the environment.

Tool Name	Cross App Analysis	Consider Devices	No Platform Modification	Freely Available Online	Consider Tampered Devices	App Data Analysis
FlowFence [77]	●	○	○	●	○	○
ContextIoT [78]	○	○	●	○	○	○
SaINT [48]	○	●	●	●	○	○
ProvThings [23]	●	●	●	○	○	○
IoTWATCH [42]	○	○	●	●	●	●
IRULER [79]	●	○	●	●	●	●
VERITAS	●	●	●	●	●	●

Table V: VERITAS vs. other smart data analysis tools.

Adversarial ML Attacks. Mallory may use adversarial machine learning to tamper with the Markov Chain model and modify the forensic analysis results. We consider this attack out of scope, as VERITAS' model is built and trained on demand only when a forensic investigation is requested. Although the attack is possible, we consider the forensic process secure and outside of the scope of VERITAS.

VII. RELATED WORK

Forensic Data Collection. Previous approaches that collect data from the smart environment and devices only focus on vendor-specific devices [63], present general methods to only collect data without any future analysis [64], [65], or have proposed models to collect data for forensic purposes using smart devices [66]–[68]. Kebande et al. proposed a generic approach, DFIF-IoT, to analyze digital forensic data in Internet of Things (IoT) settings [69]. However, the authors do not offer any real-life implementation or evaluation. Zawoad et al. proposed FAIoT, a forensic-aware eco-system to collect forensic data from smart platforms [70]. FAIoT collects data systematically but it does not offer data analysis capabilities. Chung et al. proposed a forensic framework to collect and analyze forensic data in an IoT eco-system [71]. However, this solution is limited to Amazon Alexa with only one device-specific solution.

Forensic Analysis in Smart Settings. Previous research proposed digital forensic frameworks for the IoT. The work in [72] proposes a forensic framework that analyzes network data to detect anomalous IoT activities. In [73] the authors propose an holistic forensic model that define baselines for IoT forensics. Finally, the authors in [74], [75] propose a in-depth study about challenges, approaches, and open issues in IoT forensics analysis.

Smart Data Logging. Prior works have used logging to access smart app data, which is allowed in some Ismart home/office platforms [76]. ProvThings [23] proposes a platform-centric approach that looks at activities from smart apps for data provenance purposes. However, the analysis is limited only to consider the temporal relationship between devices and apps events. Lastly, this work assumes trusted devices which, despite considered in other works [55], [77], [78], may be unrealistic in several scenarios.

Information Flow Analysis. Recently, information flow analysis (IFA) has moved from traditional domains like Android to IoT. Table V summarizes the major differences

between FlowFence [77], ContextIoT [78], ProvThings [23], SaINT [48], IoTWatcH, iRuler, and VeritaS. In general, solutions prior to VeritaS are limited to policy enforcement, data provenance, or privacy analysis in IoT. In summary, VeritaS achieves the capabilities of all the considered tools with high accuracy and low overhead. In addition, the proposed framework also offers the capability of applying deep data analysis to solve a comprehensive forensic model that includes challenges related to careless users, malicious users, malicious apps, and tampered devices.

VeritaS vs. Prior Works. VeritaS presents a lightweight solution that automatically analyses smart apps source code and applies app instrumentation to collect forensically-relevant data from a smart environment. Then, the framework analyzes the collected data to infer anomalous activities and malicious behavior from users, devices, and smart apps. VeritaS flags the activities and forensic behaviors that potentially violate the security policies defined for the smart environment.

VIII. Conclusion

Devices and sensors deployed in smart environments have access to data with high forensic value. Nonetheless, current smart home/office platforms do not provide any digital forensic capability to keep track and analyze such data. As a result, current forensic analysis do not use information from smart settings to perform their investigations. In this work, we introduce VeritaS, a novel framework that instrumented smart apps and extracted forensically-relevant logs from the smart environment. Then, the framework used a First Order Markov Chain model to automatically analyze the data for forensic purposes. We tested VeritaS in a realistic smart office environment with a total of 22 devices that produced 84209 forensically-valuable incidents. VeritaS achieved over 95% accuracy in inferring anomalous activities and forensic behaviors from users, devices, and smart apps. VeritaS's performance yielded very low overhead to the cloud server and smart apps.

Acknowledgments

This work was partially supported by the U.S. National Science Foundation (Awards: NSF-CAREER-CNS-1453647, NSF-1663051). The views expressed are those of the authors only, not of the funding agencies.

References

[1] S. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman, and R. Boreli, "An Experimental Study of Security and Privacy Risks with Emerging Household Appliances," in *Communications and Network Security (CNS)*, 2014.

[2] H. Aksu, L. Babun, M. Conti, G. Tolomei, and A. S. Uluagac, "Advertising in the IoT Era: Vision and Challenges," *IEEE Communications Magazine*, 2018.

[3] 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," *arXiv preprint arXiv:2102.10695*, 2021.

[4] B. L. R. Stojkoska and K. V. Trivodaliev, "A Review of Internet of Things for Smart Home: Challenges and Solutions," *Journal of Cleaner Production*, vol. 140, 2017.

[5] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger, and A. S. Uluagac, "A Survey on Sensor-Based Threats and Attacks to Smart Devices and Applications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, 2021.

[6] A. K. Sikder, L. Babun, H. Aksu, and A. S. Uluagac, "Aegis: a Context-aware Security Framework for Smart Home Systems," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019.

[7] A. K. Sikder, L. Babun, and A. S. Uluagac, "Aegis+ A Context-aware Platform-independent Security Framework for Smart Home Systems," *Digital Threats: Research and Practice*, vol. 2, no. 1, 2021.

[8] Control4: Can Smart Home Technology Reduce Home Insurance Rates?, https://www.control4.com/blog/440/can-smart-home-technology-reduce-home-insurance-rates, [Online; accessed 10-March-2020].

[9] 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 *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2020, pp. 1–9.

[10] L. Babun, H. Aksu, and A. S. Uluagac, "Identifying Counterfeit Smart Grid Devices: A Lightweight System Level Framework," in *IEEE International Conference on Communications (ICC)*, 2017.

[11] L. Babun and H. Aksu and A.S. Uluagac, "A System-level Behavioral Detection Framework for Compromised CPS Devices: Smart-grid Case," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 2, 2019.

[12] 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 *IEEE International Conference on Communications (ICC)*, 2020.

[13] L. Babun, H. Aksu, and A. S. Uluagac, "CPS Device-Class Identification via Behavioral Fingerprinting: From Theory to Practice," *IEEE Transactions on Information Forensics and Security*, vol. 16, 2021.

[14] A. Cosson, A. K. Sikder, L. Babun, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "Sentinel: A Robust Intrusion Detection System for IoT Networks Using Kernel-Level System Information," in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021.

[15] 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, 2019.

[16] C. Kaygusuz, L. Babun, H. Aksu, and A. S. Uluagac, "Detection of Compromised Smart Grid Devices with Machine Learning and Convolution Techniques," in *IEEE International Conference on Communications (ICC)*, 2018.

[17] K. Denney, E. Erdin, L. Babun, M. Vai, and S. Uluagac, "USB-watch: A Dynamic Hardware-assisted USB Threat Detection Framework," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2019.

[18] F. Naseem, L. Babun, C. Kaygusuz, S. Moquin, C. Farnell, A. Mantooth, and A. S. Uluagac, "CSPoweR-Watch: A Cyber-Resilient Residential Power Management System," in *IEEE Green Computing and Communications (GreenCom)*, 2019.

[19] A. K. Sikder, L. Babun, Z. B. Celik, A. Acar, H. Aksu, P. McDaniel, E. Kirda, and A. S. Uluagac, "Kratos: Multi-user Multi-device-aware Access Control System for the Smart Home," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020.

[20] S. Chitnis, N. Deshpande, and A. Shaligram, "An Investigative Study for Smart Home Security: Issues, Challenges and Countermeasures," *Wireless Sensor Network*, p. 61, 2016.

[21] P. Sundaravadivel, K. Kesavan, L. Kesavan, S. P. Mohanty, and E. Kougianos, "Smart-Log: A Deep-Learning Based Automated Nutrition Monitoring System in the IoT," *IEEE Transactions on Consumer Electronics*.

[22] M. A. B. Ahmadon, S. Yamaguchi, S. Saon, and A. K. Mahamad, "On Service Security Analysis for Event Log of IoT System Based

on Data Petri Net," in *Inter. Symp. on Consumer Electronics*, Nov 2017.

[23] Q. Wang, W. U. Hassan, A. J. Bates, and C. Gunter, "Fear and Logging in the Internet of Things," in *Network and Distributed Systems Symposium (NDSS)*, Feb 2018.

[24] SmartThings Logging, Matt J Frank, https://github.com/krlaframboise/SmartThings/blob/master/smartapps/krlaframboise/simple-event-logger.src/simple-event-logger.groovy, [Online; accessed 10-March-2020].

[25] L. P. Rondon, L. Babun, K. Akkaya, and A. S. Uluagac, "HDMI-walk: Attacking HDMI Distribution Networks Via Consumer Electronic Control Protocol," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019.

[26] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu, "Hercule: Attack Story Reconstruction via Community Discovery on Correlated Log Graph," in *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, 2016.

[27] B. Saltaformaggio, Z. Gu, X. Zhang, and D. Xu, "{DSCRETE}: Automatic Rendering of Forensic Information from Memory Images via Application Logic Reuse," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014.

[28] A. Ali-Gombe, S. Sudhakaran, A. Case, and G. G. Richard III, "DroidScraper: A Tool for Android in-memory Object Recovery and Reconstruction," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, 2019.

[29] N. Lewis, A. Case, A. Ali-Gombe, and G. G. Richard III, "Memory Forensics and the Windows Subsystem for Linux," *Digital Investigation*, vol. 26, 2018.

[30] M. Graziano, A. Lanzi, and D. Balzarotti, "Hypervisor Memory Forensics," in *International Workshop on Recent Advances in Intrusion Detection*, 2013.

[31] A. Reina, A. Fattori, F. Pagani, L. Cavallaro, and D. Bruschi, "When Hardware Meets Software: A Bulletproof Solution to Forensic Memory Acquisition," in *Proceedings of the 28th annual computer security applications conference*, 2012.

[32] L. Martignoni, A. Fattori, R. Paleari, and L. Cavallaro, "Live and Trustworthy Forensic Analysis of Commodity Production Systems," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2010.

[33] R. Hegarty, D. J. Lamb, and A. Attwood, "Digital Evidence Challenges in the Internet of Things," in *INC*, 2014, pp. 163–172.

[34] Y. Yusoff, R. Ismail and Z. Hassan, "Common Phases of Computer Forensics Investigation Model," *International Journal of Computer Science & Information Technology (IJCSIT)*, 2011.

[35] K. Denney, L. Babun, and A. S. Uluagac, "USB-Watch: a Generalized Hardware-Assisted Insider Threat Detection Framework," *Journal of Hardware and Systems Security*, vol. 4, no. 2, 2020.

[36] Why You Need Forensics in an IoT World, Raj Udeshi, https://www.guidancesoftware.com/blog/digital-forensics/2017/10/03/why-you-need-forensics-in-an-iot-world, [Online; accessed 10-March-2020].

[37] How can IoT help digital forensics?, Cognixia, https://www.cognixia.com/blog/how-can-iot-help-digital-forensics/, [Online; accessed 10-December-2020].

[38] A. I. Newaz, A. K. Sikder, M. A. Rahman, and A. S. Uluagac, "A Survey on Security and Privacy issues in Modern Healthcare Systems: Attacks and Defenses," *ACM Transactions on Computing for Healthcare*, vol. 2, no. 3, pp. 1–44, 2021.

[39] SmartThings Official Developer Documentation, Samsung, https://developer-preview.smartthings.com/, [Online; accessed 10-November-2021].

[40] OpenHAB: Open Source Automation Software for Home, https://www.openhab.org/, [Online; accessed 10-March-2020].

[41] Smart Forensics for the Internet of Things (IoT), Usama Salama, https://securityintelligence.com/smart-forensics-for-the-internet-of-things-iot/, [Online; accessed 10-March-2020].

[42] L. Babun, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "Real-time Analysis of Privacy-(un)aware IoT Applications," in *21st Privacy Enhancing Technologies Symposium (PETS)*, 2021.

[43] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: I See Your Smart Home Activities, even Encrypted!" in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020.

[44] 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 *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2020.

[45] Rondon, Luis P. and Babun, Leonardo and Aris, Ahmet and Akkaya, Kemal and Uluagac, A. Selcuk, "LightningStrike: (In)Secure Practices of E-IoT Systems in the Wild," ser. WiSec '21, 2021, p. 106116.

[46] Microsoft, Cybersecurity Policy for the Internet of Things, https://www.microsoft.com/en-us/cybersecurity/content-hub/cybersecurity-policy-for-iot, [Online; accessed 20-November-2021].

[47] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A Survey on IoT Platforms: Communication, Security, and Privacy Perspectives," *Computer Networks*, vol. 192, 2021.

[48] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. McDaniel, and A. S. Uluagac, "Sensitive Information Tracking in Commodity IoT," in *27th USENIX Security Symposium*, 2018.

[49] GroovyCodeVisitor: An Implementation of the Groovy Visitor Patterns, http://docs.groovy-lang.org/docs, [Online; accessed 10-March-2020].

[50] Generate SHA or MD5 File Checksum Hash in Java, https://howtodoinjava.com/java/io/how-to-generate-sha-or-md5-file-checksum-hash-in-java/, [Online; accessed 10-March-2020].

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

[52] PyDTMC 6.10.0, https://pypi.org/project/PyDTMC/, [Online; accessed 20-November-2021].

[53] A. R. Cassandra, "Exact and Approximate Algorithms for Partially Observable Markov Decision Processes," 1998.

[54] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thsense: A Context-aware Sensor-based Attack Detector for Smart Devices," in *26th USENIX Security Symposium*, 2017.

[55] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications," in *IEEE Security and Privacy Symposium (S&P)*, 2016.

[56] IoTBench Repository, L. Babun, Z. Berkay Celik and A. Kumar Sikder, https://github.com/IoTBench, [Online; accessed 10-March-2020].

[57] A. K. Sikder, H. Aksu, and A. S. Uluagac, "A Context-aware Framework for Detecting Sensor-based Threats on Smart Devices," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 245–261, 2019.

[58] S. Roy, J. DeLoach, Y. Li, N. Herndon, D. Caragea, X. Ou, V. P. Ranganath, H. Li, and N. Guevara, "Experimental Study with Real-world Data for Android App Security Analysis using Machine Learning," in *Proceedings of the 31st Annual Computer Security Applications Conference*, 2015.

[59] C. G. Weng and J. Poon, "A New Evaluation Measure for Imbalanced Datasets," in *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, 2008.

[60] T. Saito and M. Rehmsmeier, "The Precision-recall Plot is More Informative than the ROC Plot when Evaluating Binary Classifiers on Imbalanced Datasets," *PloS one*, vol. 10, no. 3, 2015.

[61] GitHub: Joyent node-http-signature, [Online; accessed 10-November-2021]. [Online]. Available: https://github.com/joyent/node-http-signature

[62] How Do I Activate the Two-Step Verification on my Samsung Account?, https://www.samsung.com/nz/support/mobile-

devices/activate-two-step-verification/, [Online; accessed 10-November-2021].

[63] H. Chung, J. Park, S. Lee, and C. Kang, "Digital Forensic Investigation of Cloud Storage Services," *Digital Investigation*, 2012.

[64] E. Oriwoh, D. Jazani, G. Epiphaniou, and P. Sant, "Internet of Things Forensics: Challenges and Approaches," in *Collaborative Computing: Networking, Applications and Worksharing*, 2013.

[65] S. Watson and A. Dehghantanha, "Digital Forensics: the Missing Piece of the Internet of Things Promise," *Computer Fraud & Security*, 2016.

[66] S. Perumal, N. M. Norwawi, and V. Raman, "Internet of Things (IoT) Digital Forensic Investigation Model: Top-down Forensic Approach Methodology," in *Digital Information Processing and Communications*, 2015.

[67] M. Harbawi and A. Varol, "An Improved Digital Evidence Acquisition Model for the Internet of Things Forensic: A Theoretical Framework," in *Digital Forensic and Security*, 2017.

[68] F. Bouchaud, G. Grimaud, and T. Vantroys, "IoT Forensics: Identification and Classification of Evidence in Criminal Investigations," in *13th Int. Conf. on Availability, Reliability and Security*, 2018.

[69] V. R. Kebande and I. Ray, "A Generic Digital Forensic Investigation Framework for Internet of Things (IoT)," in *IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2016.

[70] S. Zawoad and R. Hasan, "FaIoT: Towards Building a Forensics Aware Eco System for the Internet of Things," in *Services Computing*, 2015.

[71] H. Chung, J. Park, and S. Lee, "Digital Forensic Approaches for Amazon Alexa Ecosystem," *Digital Investigation*, vol. 22, pp. S15–S25, 2017.

[72] N. Koroniotis, N. Moustafa, and E. Sitnikova, "A New Network Forensic Framework based on Deep Learning for Internet of Things Networks: A Particle Deep Framework," *Future Generation Computer Systems*, vol. 110, 2020.

[73] L. Sadineni, E. Pilli, and R. B. Battula, "A Holistic Forensic Model for the Internet of Things," in *IFIP International Conference on Digital Forensics*. Springer, 2019.

[74] J. Hou, Y. Li, J. Yu, and W. Shi, "A Survey on Digital Forensics in Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 1, 2019.

[75] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, 2020.

[76] SmartThings Official Logging, Samsung, http://docs.smartthings.com/en/latest/tools-and-ide/logging.html, [Online; accessed 10-March-2020].

[77] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical Data Protection for Emerging IoT Application Frameworks," in *25th USENIX security symposium*, 2016.

[78] Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z. M. Mao, A. Prakash, and S. J. Unviersity, "ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms," in *Network and Distributed Systems Symposium (NDSS)*, 2017.

[79] Q. Wang, P. Datta, W. Yang, S. Liu, A. Bates, and C. A. Gunter, "Charting the Attack Surface of Trigger-Action IoT Platforms," in *2019 ACM Conference on Computer and Communications Security*, 2019.