6thSense: A Context-aware Sensor-based Attack Detector for Smart Devices

Amit Kumar Sikder, Hidayet Aksu, and A. Selcuk Uluagac, *Florida International University*

6thSense: A Context-aware Sensor-based Attack Detector for Smart Devices

Amit Kumar Sikder, Hidayet Aksu, A. Selcuk Uluagac

Cyber-Physical Systems Security Lab,
Electrical and Computer Engineering Department,
Florida International University.
{*asikd003, haksu, suluagac*}*@fiu.edu*

Abstract

Sensors (e.g., light, gyroscope, accelerometer) and sensing enabled applications on a smart device make the applications more user-friendly and efficient. However, the current permission-based sensor management systems of smart devices only focus on certain sensors and any App can get access to other sensors by just accessing the generic sensor API. In this way, attackers can exploit these sensors in numerous ways: they can extract or leak users' sensitive information, transfer malware, or record or steal sensitive information from other nearby devices. In this paper, we propose 6thSense, a context-aware intrusion detection system which enhances the security of smart devices by observing changes in sensor data for different tasks of users and creating a contextual model to distinguish benign and malicious behavior of sensors. 6thSense utilizes three different Machine Learning-based detection mechanisms (i.e., Markov Chain, Naive Bayes, and LMT) to detect malicious behavior associated with sensors. We implemented 6thSense on a sensor-rich Android smart device (i.e., smartphone) and collected data from typical daily activities of 50 real users. Furthermore, we evaluated the performance of 6thSense against three sensor-based threats: (1) a malicious App that can be triggered via a sensor (e.g., light), (2) a malicious App that can leak information via a sensor, and (3) a malicious App that can steal data using sensors. Our extensive evaluations show that the 6thSense framework is an effective and practical approach to defeat growing sensor-based threats with an accuracy above 96% without compromising the normal functionality of the device. Moreover, our framework costs minimal overhead.

1 Introduction

Smart devices such as smartphones and smartwatches have become omnipresent in every aspect of human life. Nowadays, the role of smart devices is not limited to making phone calls and messaging only. They are integrated into various applications from home security to health care to military [18, 60]. Since smart devices seamlessly integrate the physical world with the cyber world via their sensors (e.g., light, accelerometer, gyroscope, etc.), they provide more efficient and user-friendly applications [37, 41, 85, 55, 48].

While the number of applications using different sensors [38] is increasing and new devices offer more sensors, the presence of sensors have opened novel ways to exploit the smart devices [76]. Attackers can exploit the sensors in many different ways [76]: they can trigger an existing malware on a device with a simple flashlight [28]; they can use a sensor (e.g., light sensor) to leak sensitive information; using motion sensors such as accelerometer, and gyroscope, attackers can record or steal sensitive information from other nearby devices (e.g., computers, keyboards) or people [10, 87, 26, 42]. They can even transfer a specific malware using sensors as a communication channel [76]. Such *sensor-based* threats become more serious with the rapid growth of Apps utilizing many sensors [6, 2].

In fact, these sensor-based threats highlight the flaws of existing sensor management systems used by smart devices. Specifically, Android sensor management system relies on permission-based access control, which considers only a few sensors (i.e., microphone, camera, and GPS)[1]. Android asks for access permission (i.e., with a list of permissions) only while an App is being installed for the first time. Once this permission is granted, the user has no control over how the listed sensors and other sensors (not listed) will be used by the specific App. Moreover, using some sensors is not considered as a violation of security and privacy in Android. For instance, any App is permitted to access to motion sensors by just accessing the *sensor* API. Access to motion sensors is not controlled in Android.

---

[1] IOS, Windows, and Blackberry also have permission-based sensor management systems. In this work, we focus on Android.

Existing studies have proposed enhanced access control mechanisms for some of the sensors, but these enhancements do not cover all the sensors of a smart device [69]. Some proposed solutions introduced trusted paths on top of the existing security mechanism for controlling information flow between sensors and Apps, but these are also App-specific solutions and depend upon explicit user consent [32, 61]. Thus, introducing additional permission controls for sensors of a smart device will not mitigate the risk of all sensor-based threats as they are App specific and address only data leakage risks. Some attacks may not abuse sensors directly, instead, they may use sensors as side channels to activate another malware [34]. Albeit useful, existing security schemes overlook these critical threats which directly impact the security and privacy of the smart device ecosystem. Moreover, although sensors on smart devices seem to work independently from each other, a task or activity on a smart device may activate more than one sensor to accomplish the task. Hence, it is necessary to secure all the different sensors [5] on a smart device and consider the context of the sensors in building any solution against sensor-based threats.

In order to address the sensor-based threats, in this paper, we present a novel intrusion detection (IDS) framework called *6thSense*, a comprehensive security solution for sensor-based threats for smart devices. The proposed framework is a *context-aware IDS* and is built upon the observation that for any user activity or task (e.g., texting, making calls, browsing, driving, etc.), a different, but a specific set of sensors becomes active. In a context-aware setting, the 6thSense framework is aware of the sensors activated by each activity or task. 6thSense observes sensors data in real time and determines the current use context of the device according to which it concludes whether the current sensor use is malicious or not. 6thSense is context-aware and correlates the sensor data for different user activities (e.g., texting, making calls, browsing, driving, etc.) on the smart device and learns how sensors' data correlates with different activities. As a detection mechanism, 6thSense observes sensors' data and checks against the learned behavior of the sensors. In 6thSense, the framework utilizes several different Machine Learning-based detection mechanisms to catch sensor-based threats including Markov Chain, Naive Bayes, and LMT. In this paper, we present the design of 6thSense on an Android smartphone because of its large market share [7] and its rich set of sensors. To evaluate the efficiency of the framework, we tested it with data collected from real users (50 different users, nine different typical daily activities [3]). We also evaluated the performance of 6thSense against three different sensor-based threats and finally analyzed its overhead. Our evaluation shows that 6thSense can detect sensor-based attacks with an accuracy and F-Score over 96%. Also, our evaluation shows a minimal overhead on the utilization of the system resources.

Contributions: In summary, the main contributions of this paper are threefold—

- *First*, the design of 6thSense, a context-aware IDS to detect sensor-based threats utilizing different machine learning based models from Markov Chain to Naive Bayes to LMT.

- *Second*, the extensive performance evaluation of 6thSense with real user experiments over 50 users.

- *Third*, testing 6thSense against three different sensor-based threats.

Organization: The rest of the paper is organized as follows: we give an overview of sensor-based threats and existing solutions in Section 2. In section 3, we briefly discuss the *Android's* sensor management system. Adversary model and design facts and assumptions for 6thSense are briefly discussed in Section 4. Different detection techniques used in our framework are described in Section 5. In Sections 6 and 7, we provide a detailed overview of 6thSense including its different components and discuss its effectiveness by analyzing different performance metrics. Finally, we discuss features and limitations and conclude this paper in Sections 8 and 9, respectively.

2 Related Work

Sensor-based threats [76] on mobile devices have become more prevalent than before with the use of different sensors in smartphones such as user's location, keystroke information, etc. Different works [73] have investigated the possibility of these threats and presented different potential threats in recent years. One of the most common threats is keystroke inference in smartphones. Smartphones use on-screen QWERTY keyboard which has specific position for each button. When a user types in this keyboard, values in smartphone's motion sensor (i.e., accelerometer and gyroscope) change accordingly [16]. As different keystrokes yield different, but specific values in motion sensors, typing information on smartphones can be inferred from an unauthorized sensor such as motion sensor data or motion sensor data patterns collected either in the device or from a nearby device can be used to extract users' input in smartphones [9, 66, 52]. The motion sensor data can be analyzed using different techniques (e.g., machine learning, frequency domain analysis, shared-memory access, etc.) to improve the accuracy of inference techniques such as [12, 53, 81, 46, 58, 47]. Another form of

keystroke inference threat can be performed by observing only gyroscope data. Smartphones have a feature of creating vibrations while a user types on the touchpad. The gyroscope is sensitive to this vibrational force and it can be used to distinguish different inputs given by the users on the touchpad [51, 15, 44]. Recently, ICS-CERT also issued an alert for accelerometer-based attacks that can deactivate any device by matching vibration frequency of the accelerometer [2, 1, 70]. Light sensor readings also change while a user types on the smartphone; hence, the user input in a smartphone can be inferred by differentiating the light sensor data in normal and typing modes [71]. The light sensor can also be used as a medium to transfer malicious code and trigger message to activate malware [28, 76]. The audio sensor of a smartphone can be exploited to launch different malicious attacks (e.g., information leakage, eavesdropping, etc.) on the device. Attackers can infer keystrokes by recording tap noises on touchpad [24], record conversation of users [63], transfer malicious code to the device [73, 76], or even replicate voice commands used in voice-enabled different Apps like *Siri*, *Google Voice Search*, etc. [21, 39]. Modern smartphone cameras can be used to covertly capture screenshot or video and to infer information about surroundings or user activities [68, 43, 67]. GPS of a smartphone can be exploited to perform a false data injection attack on smartphones and infer the location of a specific device [75, 19].

Solutions for sensor-based threats: Although researchers identified different sensor-based threats in recent years, no complete security mechanism has been proposed that can secure sensors of a smart device. Most of the proposed security mechanisms for smart devices are related to anomaly detection at the application level [78, 74, 80, 22] which are not built with any protection against sensor-based threats. On the other hand, different methods of intrusion detection have been proposed for wireless sensor networks (WSN) [72, 30, 86, 23, 59], but they are not compatible with smart devices. Xu et al. proposed a privacy-aware sensor management framework for smartphones named *Semadroid* [82], an extension to the existing sensor management system where users could monitor sensor usage of different Apps and invoke different policies to control sensor access by active Apps on a smartphone. Petracca et al. introduced *AuDroid*, a SELinux-based policy framework for smartphones by performing behavior analysis of microphones and speakers [57]. AuDroid controls the flow of information in the audio channel and notifies users whenever an audio channel is requested for access. Jana et al. proposed *DARKLY*, a trust management framework for smartphones which audits applications of different trust levels with different sensor access permissions [31]. Darkly scans for vulnerability in the source code of an
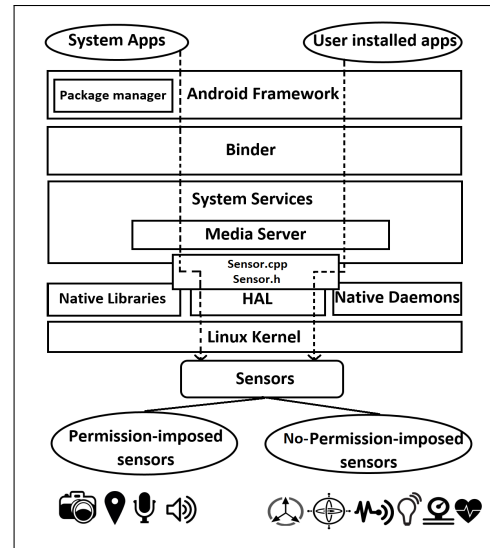

Figure 1: Android Sensor Management Architecture

application and try to modify the run-time environment of the device to ensure the privacy of sensor data.

Differences from the existing solutions: Though there is no direct comparable work to compare 6thSense with, differences between existing solutions and our framework can be noted as follows. The main limitation of Semadroid [82] is that the proposed solution is only tested against a similar type of attack scenario (information leakage by a background application). Semadroid also does not provide any extensive performance evaluation for the proposed scheme. Finally, this work depends on user permissions to fully enforce an updated policy on the sensor usage which is vulnerable as users might unknowingly approve the sensor permissions for malicious Apps. In another prior work Darkly [31], the proposed framework is not tested against any sensor-based threats. More recent work Audroid presented a policy enforced framework to secure only the audio channels of a smart device. Albeit useful, similar to the others, this work does not consider other sensor-based threats, either. *Compared to these prior works, 6thSense provides a comprehensive coverage to all the sensors in a smart device and ensures security against three different types of sensor-based threats with high accuracy.*

3 Background: Sensor Management in Smart Devices

Present versions of Android, iOS, or Blackberry do not comprise of any security mechanism to manage the information flow from sensors or among them. For example, any App can get access to motion sensors by just accessing *sensor* API. One task may need more than one sensor,

but protecting only one sensor is not a viable design. The lack of ability to secure the information flow between the sensors and Apps and a holistic view into the utilization of sensors can lead to different malicious scenarios like information leakage, eavesdropping, etc.

In our work, we focus on Android because of its open-source nature. In Figure 1, we present how Android handles access to different sensors by Apps (installed by the user) and system Apps (installed automatically by Android). Apps access to sensors by sending requests via Software Development Kit (SDK) API platform which then registers the App to a corresponding sensor [45]. If more than one App tries to access the same sensor, the SDK API runs a multiplexing process which enables different Apps to be registered in the same sensor. Hardware Abstraction Layer (HAL) works as an interface to bind the sensor hardware with the device drivers in Android. HAL has two parts: *Sensors.h* works as HAL interface and *Sensors.cpp* works as the HAL implementation. Through the HAL library, different applications can communicate with the underlying Linux kernel to read and write files associated with sensors. For most of the sensors, no permission is needed to access these files. For *permission-imposed sensors* (i.e., camera, microphone, and GPS), a permission is explicitly needed from the user to ensure file access to a specific App. This user permission is declared inside the *AndroidManifest.xml* file of an App and once the user accepts the permission, that App can have access to the corresponding sensor and other *no-permission imposed sensors* even without any explicit approval from the users. This lack of security in sensor access can lead to different malicious attacks on a device.

4 Adversary Model and Assumptions

In this section, we discuss different threats that may use sensors to execute malicious activities on a smart device. Different design assumptions are also explained in this section.

4.1 Adversary Model

For this work, we consider the following sensor-based threats similar to [76]:

- *Threat 1*-**Triggering a malicious App via a sensor.** A malicious App can exist in the smart device which can be triggered by sending a specific sensory pattern or message via sensors.

- *Threat 2*-**Information leakage via a sensor.** A malicious App can exist in the device which can leak information to any third party using sensors.

- *Threat 3*-**Stealing information via a sensor.** A malicious App can exist in the device which can exploit the sensors of a smart device and start stealing information after inferring a specific device mode (e.g., sleeping).

In this paper, we cover these three types of malicious sensor-based threats. We also note that to build our adversary model, we consider any component on a smart device that interacts with the physical world as a sensor [57]. In section 7, we show how 6thSense defends against these threats.

4.2 Design Assumptions and Features

In designing a comprehensive security scheme like 6thSense for sensor-based threats, we note the following design assumptions and features:

- *Sensor co-dependence:* A sensor in a smart device is normally considered as an independent entity on the device. Thus, one sensor does not know what is happening in another sensor. However, in this work, we consider sensors as co-dependent entities on a device instead of independent entities. The reason for this stems from the fact that for each user activity or task on a smart device, a specific set of sensors remains active. For example, if a user is walking with a phone in hand, motion sensors (i.e., gyroscope, accelerometer), the light sensor, GPS will be active. On the contrary, if the user is walking with the phone in the pocket or bag, instead of the light sensor, the proximity sensor will remain active. Thus, a co-dependent relationship exists between sensors while performing a specific task. Each activity uses different, but specific set of sensors to perform the task efficiently. Hence, one can distinguish the user activity by observing the *context of the sensors* for a specific task. 6thSense uses the context of all the sensors to distinguish between normal user activities and malicious activities. In summary, *sensors in a smart device are individually independent, but per activity-wise dependent* and 6thSense considers the context of the activities in its design.

- *Adaptive sensor sampling:* Different sensors have different sampling frequencies. To monitor all the sensor data for a specific time, a developed solution must consider and sample the sensor data correctly. Our proposed framework considers sampling the sensor data over a certain time period instead of individual sensor frequencies which mitigates any possible error in processing of data from different sensors.

- *Faster computation:* Modern high precision sensors on smart devices have high resolution and sampling rate. As a result, sensors provide large volume of data even for a small time interval. A solution for sensor-based threats should quickly process any large data from different sensors in real time while ensuring a high detection rate. To address this, we use different machine learning algorithms which are proven simple and fast techniques [11, 62].

- *Real-time monitoring:* 6thSense provides real-time monitoring to all the sensors which mitigates the possibility of data tempering or false data injection on the device.

5 Detection Techniques: Theoretical Foundation

In this section, we describe the details of the detection techniques used in 6thSense from a theoretical perspective.

For the context-aware IDS in 6thSense, we utilize several different machine learning-based techniques including Markov Chain [13], Naive Bayes [50] and alternative set of ML algorithms (e.g., PART, Logistic Function, J48, LMT, Hoeffding Tree, and Multilayer Perception) to differentiate between normal behavior from malicious behavior on a smart device. The main advantage of using Markov Chain model is that it is easy to build the model from a large dataset and computational requirements are modest which can be met by resource-limited devices. As smart devices have less processing speed, a Markov Chain-based approach can work smoothly in the context of sensor data analysis. On the other hand, Naive Bayes technique is chosen for its fast computation rate, small training dataset requirement, and ability to modify it with new training data without rebuilding the model from scratch. Other ML techniques are also common in malware detection because of higher accuracy rate. A brief discussion of these approaches in the context of 6thSense is given below. The efficacy of these different approaches utilized in 6thSense is analyzed in Section 7.

5.1 Markov Chain-Based Detection

A Markov Chain-based detection model can be described as a discrete-time stochastic process which denotes a set of random variables and defines how these variables change over time. Markov Chain can be applied to illustrate a series of events where and what state will occur next depends only on the previous state. In 6thSense, a series of events represents user activity and state represents sensor conditions (i.e., sensor values, on/off status)

of the sensors in a smart device. We can represent the probabilistic condition of Markov Chain as in Equation 1 where X_t denotes the state at time t [35]:

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

In 6thSense, we observe the changes of the conditions of a set of sensors as a variable which changes over time. The condition of a sensor indicates whether the sensor value is changing or not from a previous sensor value in time. As such, S denotes a set which represents current conditions of n number of sensors. So, S can be represented as follows.

$$
\begin{aligned}
S = \{S_1, S_2, S_3, ..., S_n\}, \\
S_1, S_2, S_3, ..., S_n = 0 \ or \ 1
\end{aligned} \quad (2)
$$

For 6thSense, we use a modified version of the general Markov Chain. Here, instead of predicting the next state, 6thSense determines the probability of a transition occurring between two states at a given time. In 6thSense, the Markov Chain model is trained with a training dataset collected from real users and the transition matrix is built accordingly. Then, 6thSense determines conditions of sensors for time t and $t+1$. Let us assume, a and b are a sensor's state in time t and $t+1$. 6thSense looks up for the probability of transition from state a to b which can be found by looking up in the transition matrix, P and calculating $P(a,b)$. As the training dataset consists sensor data from benign activities, we can assume that, if transition from state a to b is malicious, the calculated probability from transition matrix will be zero. Details of this Markov Chain-based detection model in 6thSense are given in Appendix A1.

5.2 Naive Bayes Based Detection

Naive Bayes model is a simple probability estimation method which is based on Bayes' method. The main assumption of the Naive Bayes detection is that the presence of a particular sensor condition in a task/activity has no influence over the presence of any other feature on that particular event. The probability of each event can be calculated by observing the presence of a set of specific features.

6thSense considers users' activity as a combination of n number of sensors. Assume X is a set which represents current conditions of n number of sensors. We consider that conditions of sensors are conditionally independent (See Section 4.2), which means a change in one sensor's working condition (i.e., on/off states) has no effect over a change in another sensor's working condition.

As explained earlier, the probability of executing a task depends on the conditions of a specific set of sensors. So, in summary, although one sensors' condition does not control another sensor's condition, overall the probability of executing a specific task depends on all the sensors' conditions. As an example, if a person is walking with his smartphone in his hand, the motion sensors (accelerometer and gyroscope) will change. However, this change will not force the light sensor or the proximity sensor to change its condition. Thus, sensors in a smartphone change their conditions independently, but execute a task together. We can have a generalized model for this context-aware detection [49] as follows:

$$p(X|c) = \prod_{i=1}^{n} p(X_i|c) \qquad (3)$$

Detailed description of this Naive Bayes model in 6thSense is given in Appendix A2.

5.3 Alternative Detection Techniques

In addition to Markov Chain and Naive Bayes models above, there are other machine learning algorithms (such as PART, Logistic Function, J48, LMT, Hoeffding Tree, and Multilayer Perception) that are very popular for anomaly detection frameworks because of their faster computation ability and easy implementation feature. In the alternative detection techniques, we used four types of ML-based classifier to build an analytical model for 6thSense. The following briefly discusses these classifiers and our rationale to include them.

Rule-based Learning. Rule-based ML works by identifying a set of relational rules between attributes of a given dataset and represents the model observed by the system [25]. The main advantage of the rule-based learning is that it identifies a single model which can be applied commonly to any instances of the dataset to make a prediction of outcome. As we train 6thSense with different user activities, the rule-based learning provides one model to predict data for all the user activities which simplifies the framework. For 6thSense, we chose, *PART* algorithm for the rule-based learning.

Regression Model. Regression model is widely used in data mining for its faster computation ability. This type of classifier observes the relations between dependent and independent variables to build a prediction model [20, 79]. For 6thSense, we have a total 11 attributes where we have one dependent variable (device state: malicious/benign) and ten independent variables (sensor conditions). Regression model observes the change in the dependent variable by changing the values of the independent variables and build the prediction model. We use the logistic regression model in 6thSense, which per-
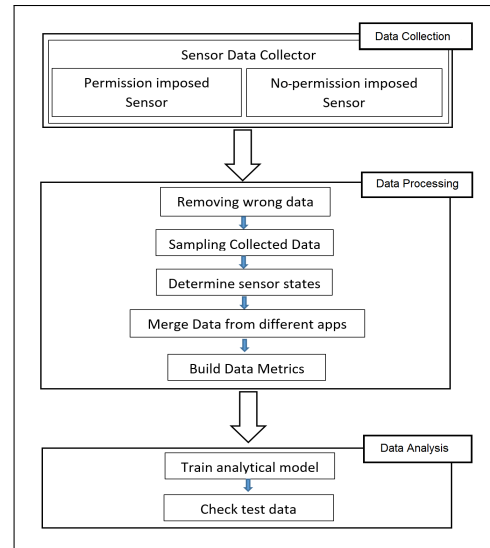

Figure 2: Overview of 6thSense.

forms with high accuracy against conventional Android malware [65].

Neural Network. Neural network is another common technique that is being adapted by researchers for malware detection. In neural network techniques, the relation between attributes of dataset is compared with the biological neurons and a relation map is created to observe the changes for each attribute [40]. We chose *Multilayer Perceptron* algorithm for training the 6thSense framework as it can distinguish relationships among non-linear dataset.

Decision Tree. Decision tree algorithms are predictive models where decision maps are created by observing the changes in one attribute in different instances [84]. These types of algorithms are mostly used in a prediction model where output can have a finite set of values. For 6thSense, we utilized and tested three different decision tree algorithms (*J48*, *LMT (Logistic Model Tree)*, and *Hoeffding tree*) to compare the outcome of our framework.

6 6thSense Framework

In this section, we provide a detailed overview of our proposed contextual behavior IDS framework, 6thSense, for detecting sensor-based threats on smart devices. As illustrated in Figure 2, 6thSense has three main phases: (1) data collection, (2) data processing, and (3) data analysis. In the data collection phase, we use a custom Android application to collect the sensor data for different user activities and the collected sensor data are then processed in the data processing phase. Note that in 6thSense some sensors provide discrete values as data (e.g., accelerometer, gyroscope, light sensor, etc.) while

other sensors provide their on-off state as sensor data (e.g., microphone, speaker, etc.). In phase 3, the collected data is fed into detection models and the end result indicates whether the current state of the device is malicious or not. The following sub-sections briefly describe these three phases.

6.1 Data Collection Phase

In this phase, 6thSense collects data from different sensors of a smart device. There can be multiple sensors in a smart device. We chose nine sensors in total to identify different user activities using a sensor-rich Android device. The sensors selected are accelerometer, gyroscope, light sensor, proximity sensor, GPS, audio sensor (microphone and speaker), camera, and headphone. 6thSense does not consider all the other sensors available in the device because all typical user activities do not affect all the sensor values. For example, the gravity sensor value does not change effectively while talking or walking with the phone. The chosen sensors are then categorized into two following categories.

- *No-permission-imposed sensors:* No-permission-imposed sensors can be defined as sensors that do not need any user permission explicitly to be accessed by an App. For 6thSense, we chose four no-permission imposed sensors (i.e., accelerometer, gyroscope, light, proximity sensors). We can also refer these sensors as data-oriented sensors in the context of 6thSense because values provided by these sensors need to be observed to infer user activities. For example, accelerometer's and gyroscope's values change with motion and they give values on *X, Y,* and *Z* axes. These values change along with the motion in different axes. To detect whether a sensor is activated or not for a specific activity, one needs to observe values of these sensors.

- *Permission-imposed sensors:* Permission-imposed sensors are those which need user permission to be accessed by an App. For 6thSense, we chose five permission-imposed sensors to build the context-aware model (camera, microphone, GPS, speaker, and headset). The conditions of these sensors can be represented by their logical states (on/off status) for different user activities. Hence, we also referred to these sensors as logic-oriented sensors in the context of 6thSense. For example, camera has only two values to identify users' activity: on and off. So, it can be represented with 0 or 1 to detect if the camera is on or off correspondingly.

To collect the data and logical values from sensors, we built a custom Android App and 6thSense used this in the data collection phase. In Android, this App uses *sensoreventlistener* API to log numerical values of the data-oriented sensors. On the other hand, the App determines the state of the sensor and logs 0 or 1 if the sensor is on or off, respectively. This App uses the user permission access to use the microphone, GPS, and camera to record the working condition of these sensors. For GPS, we consider two datasets - either GPS is turned on or not and either location is changing or not. In total, six different logic state information for five aforementioned permission-imposed sensors are collected by this App.

Note that we chose different typical daily human activities [4] that involve the smart device to build our contextual model. These activities include walking (with phone in hand and pocket), talking, interacting (playing games, browsing, listening to music), video calling, driving (as driver and passenger). Furthermore, the number of activities is configurable in 6thSense and is not limited to aforementioned examples. In the evaluation of 6thSense, we chose a total of nine typical daily activities as they are considered as common user activities for a smart device [4]. We collect these data using the App for different users to train the 6thSense framework which is then used to distinguish the normal sensor behavior from the malicious behavior. In summary, the aforementioned App collects data from nine different sensors for nine typical user activities. We observe sensor state (combination of working conditions (i.e., values, on/off status) of nine different sensors) in a per second manner for each user activity. Each second of data for user activity corresponds to 1024 state information from nine different sensors.

6.2 Data Processing Phase

After the data collection, in the second phase of the framework, we organize the data to use in the proposed IDS framework. As different sensors have different frequencies on the smart device, the total number of readings of sensors for a specific time period is different. For example, the accelerometer and gyroscope of *Samsung Galaxy S5* have a sampling frequency of approximately 202 Hz while the light sensor has a sampling frequency of 5.62 Hz. Thus, the data collected in Phase 1 needs to be sampled and reorganized. 6thSense observes the change in the sensor condition in each second to determine the overall state of our device and from this per second change, 6thSense determines the activity of users. For this reason, 6thSense takes all the data given by a single sensor in a second and calculates the average value of the sensor reading. This process is only applicable for the data oriented sensors as mentioned earlier. Again, the data collected from the App is numerical value given by the sensor. However, for the detection model, we only

Sensor type	Name	Model	Specification
No-permission imposed sensors	Accelerometer	MPU6500 Acceleration Sensor	19.6133 m/s^2, 203.60 Hz, 0.25 mA
	Gyroscope	MPU6500 Gyroscope Sensor	8.726646 rad/s, 203.60 Hz, 6.1 mA
	Light Sensor	TMG399X RGB Sensor	600000 lux, 5.62 Hz, 0.75 mA
	Proximity Sensor	TMG399X proximity sensor	8V, 0.75 mA
Permission-imposed sensors	Camera	Samsung S5K2P2XX	12 megapixels, 30 fps, 4.7 mA
	Microphone	Qualcomm Snapdragon 801 Processor built in microphone	86 dB, .75 mA
	Speaker	Qualcomm Snapdragon 801 Processor built in speaker	110 dB, 1 mA

Table 1: Sensor list of Samsung Galaxy S5 Duo used in experiment.

consider the condition of the sensors. 6thSense observes the data collected by the aforementioned App and determines whether the condition of sensors is changing or not. If the sensor value is changing from the previous value in time, 6thSense represents the sensor condition as 1 and 0 otherwise. The logic state information collected from the sensors need to be reorganized, too as these data are merged with the data collected from the collected values from the other sensors to create an input matrix. The sampling frequency of the logical state detection is 0.2 Hz which means in every five seconds the App generates one session of dataset. We consider the condition of the sensors to be the same over this time period and organize the data accordingly. The reorganized data generated from the aforementioned App are then merged to create the training matrices.

6.3 Data Analysis Phase

In the third and final phase, 6thSense uses different machine learning-based detection techniques introduced in the previous section to analyze the data matrices generated in the previous phase.

For the Markov Chain-based detection, we use 75% of the collected data to train 6thSense and generate the transition matrix. This transition matrix is used to determine whether the transition from one state to another is appropriate or not. Here, state refers to generic representation of all the sensors' conditions on a device. For testing purposes we have two different data set — basic activities or trusted model and malicious activities or threat model. The trusted model consists of 25% of the collected data for different user activities. We test the trusted model to ensure the accuracy of the 6thSense framework in detecting benign activities. The threat model is built from performing the attack scenarios mentioned in Section 4. We calculate the probability of a transition occurring between two states at a given time and accumulate the total probability to distinguish between normal and malicious activities.

To implement the Naive Bayes-based detection tech-

nique, we use the training sessions to define different user activities. In 6thSense, we have nine typical user activities in total as listed in Table 2. We use groundtruth user data to define these activities. Using the theoretical foundation explained in Section 5, we calculate the probability of a test session to belong to any of these defined activities. As we consider one second of data in each computational cycle, we calculate the total probability up to a predefined configurable time interval (in this case five minutes). This calculated probability is used to detect malicious activities from normal activities. If the computed probability for all the known benign activities is not over a predefined threshold, then it is detected as a malicious activity.

For the other alternative machine-learning-based detection techniques, we used WEKA, a data mining tool which offers data analysis using different machine learning approaches [64, 27]. Basically, WEKA is a collection of machine learning algorithms developed at the University of Waikato, New Zealand, which can be directly applied to a dataset or can be integrated with a framework using JAVA platform [56]. WEKA offers different types of classifier to analyze and build predictive model from given dataset. We use 10 fold cross-validation method to train and test 6thSense with different ML techniques in Section 7.

7 Performance Evaluation of 6thSense

In this section, we evaluate the efficiency of the proposed context-aware IDS framework, 6thSense, in detecting the sensor-based threats on a smart device. We test 6thSense with the data collected from different users for benign activities and adversary model described in Section 4. As discussed earlier, 6thSense considers three sensor-based threats: (1) a malicious App that can be triggered via a light or motion sensors, (2) a malicious App that can leak information via audio sensor, and (3) a malicious App that steals data via camera. Furthermore, we measured the performance impact of 6thSense on the device and present a detailed results for the efficiency of

the 6thSense framework. Finally, we discuss the performance overhead of the framework in this section.

7.1 Training Environment

In order to test the effectiveness of 6thSense, we implemented it on a sensor-rich Android-based smartphone. However, our framework would also efficiently work in another smart device such as smartwatch. In the evaluations, we used *Samsung Galaxy S5 Duos* as a reference Android device to collect sensor data for different typical user activities. We chose this Android device as *Samsung* currently holds approximately 20.7% of total marketshare of smartphone [8] and provides a rich set of sensors. A list of sensors of *Samsung Galaxy S5 Duos* is given in Table 1. As discussed earlier, we selected 9 different typical user activities or tasks to collect user data. These are typical basic activities with smartphones that people usually do in their daily lives [3]. The user activities/tasks are categorized in two categories as generic activities and user related activities.

Generic activities are the activities in which the sensor readings are not affected by the smartphone users. Sleeping, driving with the phone using GPS as a navigator, and driving with phone in pocket are three generic activities that we considered in this work. Basically, in the generic activities, sensors' data are not affected by different users since the smart phone is not in contact with the user or user is not directly interacting with the phone. For *user-related activities*, in which the sensor readings may be affected by the device user, we identified six different activities including walking with the phone in hand, playing games, browsing, and making voice calls and video calls.

6thSense was tested by 50 different individuals aged from 18 to 45 while the sensor data was collected. We note that our study with human subjects was approved by the appropriate Institutional Review Board (IRB) and we followed all the procedures strictly in our study. Each participant received some monetary compensation for participating in our experiments. To ensure privacy and anonymity, we used fake user IDs rather than any personal information. We collected 300 sets of data for six user-related activities where each dataset comprised of 5 minutes long data from the selected nine sensors mentioned in Section 6. We also collected three sets of data for each general activity. We asked the different users to perform the same activity to ensure the integrity for different tasks. Note that each five minute of data collected for user related and generic activities corresponds to 300 events with 1024 different states. Here, states represent a combination of conditions (i.e., values, on/off status) of nine different sensors and events represent user activities per second. So, a total of 307,200 different event-state

information were analyzed by 6thSense.

For the malicious dataset, we created three different attack scenarios considering the adversary model mentioned in Section 4. For Threat 1, we developed two different Android Apps which could be triggered using the light sensor and motion sensors on the smartphone. To perform the attack described in Threat 2, we developed a malware that could record conversations as audio clips and playback after a specific time to leak the information. This attack scenario included both the microphone and speaker on the smartphone. For Threat 3, we developed a malicious App that could scan all the sensors and if none of the sensors were changing their working con-

Task Category	Task Name
Generic Activities	1. Sleeping
	2. Driving as driver
	3. Driving as passenger
User-related Activities	1. Walking with phone in hand
	2. Walking with phone in pocket/bag
	3. Playing games
	4. Browsing
	5. Making phone calls
	6. Making video calls

Table 2: Typical Activities of Users on Smart Device [3].

ditions, the malicious App could open up the camera and record videos surreptitiously. We collected 15 different datasets from these three attack scenarios to test the efficacy of 6thSense against these adversaries.

7.2 Dataset

In order to test 6thSense, we divided the collected real user data into two sections as it is a common practice [77]. 75% of the collected benign dataset was used to train the 6thSense framework and 25% of the collected data along with malicious dataset were used for testing purposes. For the Markov Chain-based detection technique, the training dataset was used to compute the state transitions and to build the transition matrix. On the other hand, in the Naive Bayes-based detection technique, the training dataset was used to determine the frequency of sensor condition changes for a particular activity or task. As noted earlier, there were nine activities for the Naive Bayes technique. We split the data according to their activity for this approach. For the analysis of the other ML-based approaches, we define all the data in benign and malicious classes. The data were then used to train and test 6thSense using 10-fold cross validation for different ML algorithms.

Threshold (Number of consecutive malicious states)	Recall rate	False negative rate	Precision rate (specificity)	False positive rate	Accuracy	F-score
0	0.62	0.38	1	0	0.6833	0.7654
1	0.86	0.14	1	0	0.8833	0.9247
2	0.96	0.04	1	0	0.9667	0.9796
3	0.98	0.02	1	0	0.9833	0.9899
5	1	0	0.9	0.1	0.9833	0.9474
6	1	0	0.8	0.2	0.9667	0.8889
8	1	0	0.6	0.4	0.9333	0.75
10	1	0	0.5	0.5	0.9167	0.6667
12	1	0	0.5	0.5	0.9167	0.6667
15	1	0	0.3	0.7	0.8833	0.4615

Table 3: Performance evaluation of Markov Chain based model.

7.3 Performance Metrics

In the evaluation of 6thSense, we utilized the following six different performance metrics: Recall rate (sensitivity or True Positive rate), False Negative rate, Specificity (True Negative rate), False Positive rate, Accuracy, and F-score. True Positive (TP) indicates number of benign activities that are detected correctly while true negative (TN) refers to the number of correctly detected malicious activities. On the other hand, False Positive (FP) states malicious activities that are detected as benign activities and False Negative (FN) defines number of benign activities that are categorized as malicious activity. F-score is the performance metric of a framework that reflects the accuracy of the framework by considering the recall rate and specificity. These performance metrics are defined as follows:

$$Recall\ rate = \frac{TP}{TP+FN}, \quad (4)$$

$$False\ negative\ rate = \frac{FN}{TP+FN}, \quad (5)$$

$$Specificity = \frac{TN}{TN+FP}, \quad (6)$$

$$False\ positive\ rate = \frac{FP}{TN+FP}, \quad (7)$$

$$Recall\ rate = \frac{TP}{TP+FN}, \quad (8)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (9)$$

$$F-score = \frac{2*Recall\ rate*Precision\ rate}{Recall\ rate+Precision\ rate} \quad (10)$$

In addition to the aforementioned performance metrics, we considered Receiver Operating Characterstic (ROC) curve as another performance metric for 6thSense.

7.4 Evaluation of Markov Chain-Based Detection

In the Markov Chain-based detection technique, we question whether the transition between two states (sensors' on/off condition in each second) is expected or not. In the evaluations, we used 65 testing sessions in total, among which 50 sessions were for the benign activities and the rest of the sessions were for the malicious activities. A session is composed of a series of sensory context conditions where a sensory context condition is the set of all available sensor conditions (on/off state) for different sensors. As discussed earlier in Section 6, a sensor condition is a value indicating whether the sensor data is changing or not. In this evaluation, the sensory context conditions were computed every one second. We observed that in real devices sometimes some sensor readings would be missed or real data would not be reflected probably due to hardware or software imperfections. And, real malicious Apps would cause consecutive malicious states on the device. Therefore, to overcome this, we also keep track of number of consecutive malicious states and use it as a threshold after which the session is considered as malicious. Table 3 displays the evaluation results associated wit the Markov Chain-based detection technique. When the threshold for consecutive malicious states is 0, i.e., when no threshold is applied, the accuracy is just 68% and FNR is as high as 38%. With increasing the threshold value, the accuracy first increases up to 98% then start decreasing.

The possible cut-off threshold can be three consecutive malicious occurrences which has both accuracy and F-score over 98%. In Table 3, different performance indicators for Markov Chain based detection are presented. We can observe that FN and TN rates of Markov Chain-based detection decrease as the threshold of consecutive malicious states increases. Again, both accuracy and F-

Threshold Probability	Recall rate	False negative rate	Precision rate (specificity)	False positive rate	Accuracy	F-score
55%	1	0	0.6	0.4	0.9333	0.75
57%	1	0	0.7	0.3	0.95	0.8235
60%	1	0	0.7	0.3	0.95	0.8235
62%	1	0	0.7	0.3	0.95	0.8235
65%	0.94	0.06	0.7	0.3	0.9	0.8024
67%	0.88	0.12	0.7	0.3	0.85	0.7797
70%	0.7	0.3	0.8	0.2	0.7167	0.7467
72%	0.7	0.3	0.9	0.1	0.7333	0.7875
75%	0.66	0.34	0.9	0.1	0.7	0.7616
80%	0.66	0.34	0.9	0.1	0.7	0.7615

Table 4: Performance evaluation of Naive Bayes model.

score reach to a peak value with the threshold of three consecutive malicious states on the device. From Figure 3, we can see that FP rate remains zero while TP rate increases at the beginning. The highest TP rate without introducing any FP case is over 98%. After 98%, it introduces some FP cases in the system which is considered as a risk to the system. In summary, Markov Chain-base detection in 6thSense can acquire accuracy over 98% without introducing any FP cases.
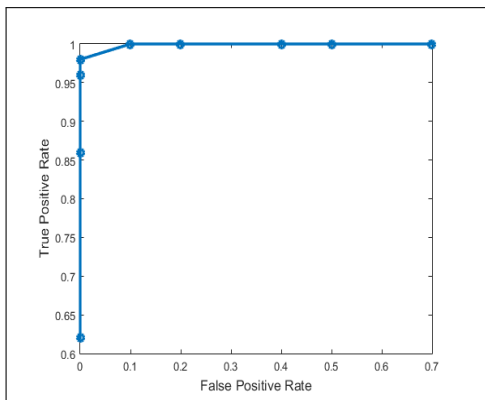

Figure 3: ROC curve of Markov Chain-based detection.

7.5 Evaluation of Naive Bayes-based Detection

In the Naive Bayes-based detection technique, 6thSense calculates the probability of a session to match it with each activity defined in Section 7.1. Since all the activities are benign and there is no malicious activity (i.e., ground-truth data), 6thSense checks calculated probability of an activity from dataset against a threshold to determine the correct activity. If there is no match for a certain sensor condition with any of the activities, 6thSense detects the session as malicious. Table 4 shows the evaluation results.

For a threshold value of 55%, FN rate is zero. However, FPR is too high, which lowers F-score of the framework. For a threshold of 60%, FPR decreases while FNR is still zero. In this case, accuracy is 95% and F-score is 82%. If the threshold is increased over 65%, it reduces the recall rate which affects accuracy and F-score. The evaluation indicates that the threshold value of 60% provides an accuracy of 95% and F-score of 82%.

From Figure 4, one can observe the relation between FPR and TPR of Naive Bayes-based detection. For FPR larger than 0.3, TPR becomes 1.
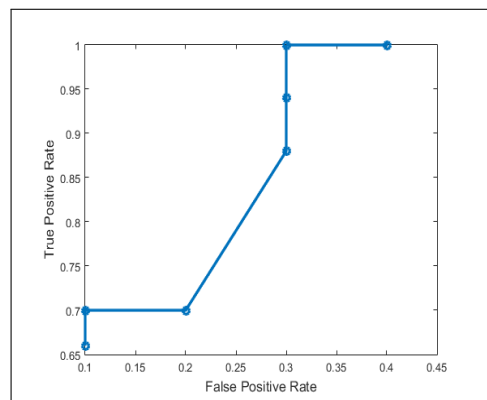

Figure 4: ROC curve of Naive Bayes-based detection.

7.6 Evaluation of Alternative Detection Techniques

In alternative detection techniques, we used other supervised machine learning techniques to train the 6thSense framework. For this, we utilized WEKA and it provides three types of analysis - split percentage analysis, cross-validation analysis, and supplied test set analysis. We chose 10 fold cross-validation analysis to ensure that all the data was used for both training and test. Thus, the

Algorithms	Recall rate	False negative rate	Precision rate	False positive rate	Accuracy	F-score
PART	0.9998	0.0002	0.6528	0.3472	0.99	0.7899
Logistic Function	0.9997	0.0003	0.2778	0.7222	0.998	0.4348
J48	0.9998	0.0002	0.6528	0.3472	0.99	0.7899
LMT	0.9998	0.0002	0.9306	0.0694	0.9997	0.964
Hoeffding Tree	1	0	0.0556	0.9444	0.9978	0.1053
Multilayer Perceptron	0.9998	0.0002	0.6944	0.3056	0.9991	0.8196

Table 5: Performance of other different machine learning based-detection techniques tested in 6thSense.

error rate of the predictive model would be minimized in the cross validation. In Table 5, a detailed evaluation of different machine learning algorithms is given for 6thSense. For *Rule Based Learning*, 6thSense has the best result for *PART* algorithm, which has an accuracy of 0.99 and F-score of 0.7899. On the other hand, for *Regression Analysis*, we use the logistic function which has high FPR (0.7222) and lower F-score (0.4348). *Multilayer Perceptron* algorithm gives an accuracy of 0.9991 and F-score of 0.8196, which is higher than previously mentioned algorithms. However, FPR is much higher (0.3056), which is actually a limitation for intrusion detection frameworks in general. Compared to these algorithms, *Linear Model Tree (LMT)* gives better results in detecting sensor-based attacks. This evaluation indicates that *LMT* provides an accuracy of 0.9997 and F-score of 0.964.

7.7 Comparison

In this subsection, we give a comparison among the different machine-learning-based detection approaches tested in 6thSense for defending against sensor-based threats. For all the approaches, we select the best possible case and report their performance metrics in Table 6. For Markov Chain-based detection, we choose three consecutive malicious states as valid device conditions. On the other hand, in Naive Bayes approach, the best performance is observed for the threshold of 60%. For other machine learning algorithms tested via WEKA, we choose LMT as it gives highest accuracy among other machine learning algorithms. These results indicate that LMT provides highest accuracy and F-score compared to the other two approaches.

On the contrary, Naive Bayes model displays higher recall rate and less FNR than other approaches. However, the presence of FPR in IDS is a serious security threat to the system since FPR refers to a malicious attack that is identified as a valid state, which is a threat to user privacy and security of the device. Both Markov Chain and LMT has lower FPR. In summary, considering F-score and accuracy of all these approaches, we conclude that

LMT performs better than the others.

Performance Metrics	Markov Chain	Naive Bayes	LMT
Recall rate	0.98	1	0.9998
False Negative Rate	0.02	0	0.0002
Precision rate	1	0.7	0.9306
False positive rate	0	0.3	0.0694
Accuracy	0.9833	0.9492	0.9997
F-Score	0.9899	0.8235	0.964
auPRC	0.947	0.686	0.91

Table 6: Comparison of different machine-learning-based approaches proposed for 6thSense (i.e., Markov Chain, Naive Bayes, and LMT).

7.8 Performance Overhead

As previously mentioned, 6thSense collects data in an Android device from different sensors (permission and no-permission imposed sensors). In this sub-section, we measure the performance overhead introduced by 6thSense on the tested Android device in terms of CPU usage, RAM usage, file size, and power consumption and Table 7 gives the details of the performance overhead. For no-permission-imposed sensors, the data collection phase logs all the values within a time interval which causes an increased usage of RAM, CPU and Disc compared to permission- imposed or logic-oriented sensors. For the power consumption, we observe that no-permission-imposed sensors use higher power than permission-imposed sensors. This is mainly because logic-oriented sensors have lower sampling rate, which reduces its resource needs.

The overall performance overhead is as low as 4% of CPU, less then $40MB$ RAM space, and less than $15MB$ disc space. Thus, its overhead is minimal and acceptable for an IDS system on current smartphones. One of the main concerns of implementing 6thSense on Android device is the power consumption. Table 7 also shows the power consumption of the Android app used in 6thsense. For one minute, 6thsense consumes 16.62 mW power which increases

upto 178.33mW for ten seconds. The main reason of this high power consumption is that all the sensors are kept on for the data collection and all the data are saved on device for later analysis. However, in practical settings, the data would not be saved on device rather a real time analysis would be done, which indeed will decrease the power consumption. Without saving the data, the power consumption significantly becomes smaller. From Table 7, we can observe that the power consumption of 6thSense becomes 72.35 mW which is almost 2.5 times lower than otherwise. Also, all the sensors do not have to remain on for the analysis part. Data can be observed if the smart device is in unlocked status. Also, a suitable interval can be chosen for the data analysis by estimating average time of an attack. This is one of the possible future research directions for 6thSense.

Parameters	Time	No-permission imposed sensors	Permission imposed sensors
CPU Usage	N/A	3.90%	0.3%
RAM Usage	N/A	23 MB	14 MB
Disc Usage	For 1 min	6.5 MB	1 KB
	For 5 min	9 MB	2 KB
	For 10 min	12 MB	3 KB
Power Consumption	1 min	13.5 mW	3.12 mW
	5 min	96.67 mW	27.4 mW
	10 min	133.33 mW	45 mW
Power Consumption (without datafile)	1 min	2.68 mW	0.23 mW
	5 min	23.4 mW	9.63 mW
	10 min	55.35 mW	17 mW

Table 7: Performance Overhead of Android Apps.

8 Discussion and Limitations

- **Features and Benefits-** Compared to the existing solutions, 6thSense differentiates itself by considering a context-aware model to detect sensor-based threats. As sensors provide continuous data to the apps, security schemes must handle real-time data rather than stored data in the system. While most of the existing solutions work with the stored data or the data used by Apps [14, 29], 6thSense offers real-time sensor monitoring. On the other hand, modern high precision sensors on-board have higher frequency and sensitivity. These sensors can detect slight changes in the smart device's ambiance which reflects on sensor values. To overcome frequent changes in sensor values, 6thSense considers average values over one second, which mitigates the effect of changes in sensor values caused by the device ambiance. For example, if a person walks by a smartphone, the light sensor and motion sen-

sors values will be changed for that instance. However, if one considers the average value over one second, it will be compensated by other readings recorded over one second. Another unique feature of 6thSense is that instead of considering the individual sensor data accessed by the Apps, user activities are monitored, which forms the basis of the contextual model for the 6thSense framework. 6thSense observes changes in sensors for different user activities. As more than one sensor remain active to perform a task, attackers need to learn the pattern of all the sensors for user activities to outperform 6thSense. If an attacker targets one specific sensor, an attack scenario will differ from normal user activity which can be easily detected by 6thSense. Thus, the context of user activities is very important to detect malicious activities in 6thSense. Moreover, 6thSense considers all the sensors' conditions as one device state, which provides easy monitoring of the sensors by one framework. Finally, 6thSense can work with all the sensors on a smart device extending the security beyond the traditional permission-imposed sensors (i.e., GPS, microphone, and camera).

- **Application Level Detection-** One of the promising practical applications of 6thSense is to combine the sensor level detection with an application level intrusion detection. 6thSense focuses on detecting malicious activities by observing working conditions of sensors rather than individual App behaviors. However, some prior works [82, 57, 78] also show that it is possible to achieve good accuracy when detecting malicious activities by observing sensor usage in the application level. The combination of application and sensor level detection might be one promising way to further improve the performance of 6thSense. Another interesting application of 6thSense would be to combine it with an online training method to eliminate the necessity of offline training.

- **Sensor-based threats in real-life settings -** One limitation of 6thSense is the adversaries (sensor-based attacks) used in the evaluation were constructed in a lab environment. Note that as of this writing, there are no real sensor-based malware in the wild. However, recently, many independent researchers have confirmed the feasibility of sensor-based threats for smart devices [57, 17, 1, 70]. Indeed, more recently, ICS-CERT also warned the vendors and the wider communities about the possibility of exploiting the sensors of a device to alter sensors' output in a controlled way to perform malicious behavior in the device [2]. Although there

are different limited security schemes to mitigate these attacks, there is no comprehensive contextual solution to secure smart devices from the sensor-based threats. Furthermore, we note that even locking down the sensor API with explicit permissions at the OS level would not surpass the sensor-based threats as users are not aware of these threats yet and can allow malicious Apps to use sensors unknowingly. For all these points, we built the prof-of-concept versions of the sensor-based threats discussed in Section 4. We also note that to ensure the reliability of the lab-made malware (i.e., a specific malicious App) for three threats described in the Adversary Model Section, we checked how they perform with respect to the real malicious software scanners. For this, we uploaded our lab-made malware on *VirusTotal* and tabulated the results of the performance of 60 different malware scanners available at the VirusTotal website in Table 8. As seen in this table, the sensor-based threats are not recognized by the different scanners. Only 2 out of 60 reported that they could detect, but these two only reported risks without clearly identifying any explicit malicious behaviour. Hence, it is difficult to

Adversary Model	Detection Ratio
Threat-1	2/60
Threat-2	2/60
Threat-3	3/62

Table 8: *VirusTotal* scan result for the adversary models.

detect the sensor-based threats mentioned in this paper by existing security schemes. Moreover, some security schemes only provide security to the specific sensors [57]. 6thSense covers several sensors as opposed to other existing existing security schemes without alerting the device. Also, existing sensor management systems of Android depends on explicit user-permission only for specific sensors (e.g., microphone, camera, speaker). As users are not aware of sensor-based threats yet, they can allow malicious Apps to use sensors unknowingly. Additionally, 6thSense also covers no-permission-imposed sensors (e.g., motion sensors, light sensor, etc.) in its design.

- **Context-aware Malicious App-** One compelling case is that how 6thSense can defend against a malicious App which can learn and imitate a user's behavior. As described earlier, Threat 3, described in Section 4.1, can observe the working conditions of all the sensors and detect, for instance, a sleeping activity that records videos stealthily. 6thSense can even detect this powerful context-aware malware

successfully. In summary, to outperform 6thSense, a malicious App must behave like a benign App all the time in a device, which limits its malicious purposes. Any incompatible behavior in the sensors of a smart device can be easily detected by 6thSense.

9 Conclusion

Wide utilization of sensor-rich smart devices created a new attack surface namely sensor-based attacks. Accelerometer, gyroscope, light, etc. sensors can be abused to steal and leak sensitive information or malicious Apps can be triggered via sensors. Security in current smart devices lacks appropriate defense mechanisms for such sensor-based threats. In this paper, we presented 6thSense, a novel context-aware task-oriented sensor-based attack detector for smart devices. We articulated problems in existing sensor management systems and different sensor-based threats for smart devices. Then, we presented the design of 6thSense to detect sensor-based attacks on a sensor-rich smart device with low-performance overhead. 6thSense utilized different machine learning (ML) techniques to distinguish malicious activities from benign activities on a device. To the best of our knowledge, 6thSense is the first comprehensive context-aware security solution against sensor-based threats. We evaluated 6thSense on real devices with 50 different individuals. 6thSense achieved over 95% of accuracy with different ML algorithms including Markov Chain, Naive Bayes, and LMT . We also evaluated 6thSense against three different sensor-based threats, i.e., information leakage, eavesdropping, and triggering a malware via sensors. The empirical evaluation revealed that 6thSense is highly effective and efficient at detecting sensor-based attacks while yielding minimal overhead.

Future Work: While 6thSense detects different sensor-based threats with high accuracy, we will expand 6thSense in our future work as follows: We will study other performance metrics such as Precision Recall Curve (PRC). We will evaluate the efficacy of 6thSense in other smart devices such as smartwatches and analyze all of its phases in its operations. Moreover, due to limited resources of the smart devices, trade-off between power consumption and effectiveness is a prime concern of any intrusion detection framework. Hence, we will study frequency-accuracy trade-off, battery-accuracy trade-off, and battery-frequency trade-off of 6thSense in different smart devices.

References

[1] Hacking sensors. `https://www.usenix.org/conference/enigma2017/conference-program/presentation/kim`.

Accessed: 2017-5-30.

[2] Mems accelerometer hardware design flaws (update a). https://ics-cert.us-cert.gov/alerts/ICS-ALERT-17-073-01A. Accessed: 2017-5-30.

[3] U.s. smartphone use in 2015. http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/, April 2015.

[4] A week in the life analysis of smartphone users. http://www.pewinternet.org/2015/04/01/, April 2015.

[5] Analyzing the power consumption of mobile antivirus software on android devices. http://drshem.com/2015/11/08/, August 2016.

[6] Android antivirus protection: Security steps you should take. http://us.norton.com/Android-Anti-Virus-Protection/article, sep 2016.

[7] Smartphone os market share, 2016 q2. http://www.idc.com/prodserv/smartphone-os-market-share.jsp, August 2016.

[8] Smartphone vendor market share, 2016 q2. http://www.idc.com/prodserv/smartphone-market-share.jsp, August 2016.

[9] AL-HAIQI, A., ISMAIL, M., AND NORDIN, R. Keystrokes inference attack on android: A comparative evaluation of sensors and their fusion. *Journal of ICT Research and Applications 7*, 2 (2013), 117–136.

[10] ASONOV, D., AND AGRAWAL, R. Keyboard acoustic emanations. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on* (May 2004), pp. 3–11.

[11] AVILÉS-ARRIAGA, H., SUCAR-SUCCAR, L., MENDOZA-DURÁN, C., AND PINEDA-CORTÉS, L. A comparison of dynamic naive bayesian classifiers and hidden markov models for gesture recognition. *Journal of applied research and technology 9*, 1 (2011), 81–102.

[12] AVIV, A. J., SAPP, B., BLAZE, M., AND SMITH, J. M. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference* (New York, NY, USA, 2012), ACSAC '12, ACM, pp. 41–50.

[13] BROOKS, S., GELMAN, A., JONES, G., AND MENG, X.-L. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[14] BUGIEL, S., DAVI, L., DMITRIENKO, A., HEUSER, S., SADEGHI, A.-R., AND SHASTRY, B. Practical and lightweight domain isolation on android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* (2011), ACM, pp. 51–62.

[15] CAI, L., AND CHEN, H. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX Conference on Hot Topics in Security* (Berkeley, CA, USA, 2011), HotSec'11, USENIX Association, pp. 9–9.

[16] CAI, L., AND CHEN, H. *On the practicality of motion based keystroke inference attack*. Springer, 2012.

[17] CARLINI, N., MISHRA, P., VAIDYA, T., ZHANG, Y., SHERR, M., SHIELDS, C., WAGNER, D., AND ZHOU, W. Hidden voice commands. In *25th USENIX Security Symposium (USENIX Security 16), Austin, TX* (2016).

[18] CHAN, M., CAMPO, E., ESTÈVE, D., AND FOURNIOLS, J.-Y. Smart homes—current features and future perspectives. *Maturitas 64*, 2 (2009), 90–97.

[19] COFFED, J. The threat of gps jamming: The risk to an information utility. *Report of EXELIS, Jan. Chicago* (2014).

[20] DAHL, G. E., STOKES, J. W., DENG, L., AND YU, D. Large-scale malware classification using random projections and neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (2013), IEEE, pp. 3422–3426.

[21] DIAO, W., LIU, X., ZHOU, Z., AND ZHANG, K. Your voice assistant is mine: How to abuse speakers to steal information and control your phone. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices* (New York, NY, USA, 2014), SPSM '14, ACM, pp. 63–74.

[22] ENCK, W., GILBERT, P., HAN, S., TENDULKAR, V., CHUN, B.-G., COX, L. P., JUNG, J., McDANIEL, P., AND SHETH, A. N. Taintdroid: An information-flow tracking system for real-time privacy monitoring on smartphones. *ACM Trans. Comput. Syst. 32*, 2 (June 2014), 5:1–5:29.

[23] FAROOQI, A. H., KHAN, F. A., WANG, J., AND LEE, S. A novel intrusion detection framework for wireless sensor networks. *Personal and ubiquitous computing 17*, 5 (2013), 907–919.

[24] FOO KUNE, D., AND KIM, Y. Timing attacks on pin input devices. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 678–680.

[25] GU, G., PORRAS, P. A., YEGNESWARAN, V., FONG, M. W., AND LEE, W. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Usenix Security* (2007), vol. 7, pp. 1–16.

[26] HALEVI, T., AND SAXENA, N. A closer look at keyboard acoustic emanations: Random passwords, typing styles and decoding techniques. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security* (New York, NY, USA, 2012), ASIACCS '12, ACM, pp. 89–90.

[27] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The weka data mining software: an update. *ACM SIGKDD explorations newsletter 11*, 1 (2009), 10–18.

[28] HASAN, R., SAXENA, N., HALEVIZ, T., ZAWOAD, S., AND RINEHART, D. Sensing-enabled channels for hard-to-detect command and control of mobile devices. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security* (New York, NY, USA, 2013), ASIA CCS '13, ACM, pp. 469–480.

[29] HILTY, M., PRETSCHNER, A., BASIN, D., SCHAEFER, C., AND WALTER, T. A policy language for distributed usage control. In *European Symposium on Research in Computer Security* (2007), Springer, pp. 531–546.

[30] IOANNIS, K., DIMITRIOU, T., AND FREILING, F. C. Towards intrusion detection in wireless sensor networks. In *Proc. of the 13th European Wireless Conference* (2007), pp. 1–10.

[31] JANA, S., NARAYANAN, A., AND SHMATIKOV, V. A scanner darkly: Protecting user privacy from perceptual applications. In *Security and Privacy (SP), 2013 IEEE Symposium on* (2013), IEEE, pp. 349–363.

[32] JANG, Y., SONG, C., CHUNG, S. P., WANG, T., AND LEE, W. A11y attacks: Exploiting accessibility in operating systems. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 103–115.

[33] JHA, S., TAN, K. M., AND MAXION, R. A. Markov chains, classifiers, and intrusion detection. In *csfw* (2001), vol. 1, Citeseer, p. 206.

[34] JOY PERSIAL, G., PRABHU, M., AND SHANMUGALAKSHMI, R. Side channel attack-survey. *Int J Adva Sci Res Rev 1*, 4 (2011), 54–57.

[35] KEILSON, J. *Markov chain models—rarity and exponentiality*, vol. 28. Springer Science & Business Media, 2012.

[36] KRUEGEL, C., MUTZ, D., ROBERTSON, W., AND VALEUR, F. Bayesian event classification for intrusion detection. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual* (2003), IEEE, pp. 14–23.

[37] LANE, N. D., MILUZZO, E., LU, H., PEEBLES, D., CHOUDHURY, T., AND CAMPBELL, A. T. A survey of mobile phone sensing. *IEEE Communications Magazine 48*, 9 (Sept 2010), 140–150.

[38] LANE, N. D., XU, Y., LU, H., HU, S., CHOUDHURY, T., CAMPBELL, A. T., AND ZHAO, F. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 13th international conference on Ubiquitous computing* (2011), ACM, pp. 355–364.

[39] LEI, L., WANG, Y., ZHOU, J., ZHA, D., AND ZHANG, Z. A threat to mobile cyber-physical systems: Sensor-based privacy theft attacks on android smartphones. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on* (July 2013), pp. 126–133.

[40] LINDA, O., VOLLMER, T., AND MANIC, M. Neural network based intrusion detection system for critical infrastructures. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on* (2009), IEEE, pp. 1827–1834.

[41] MACIAS, E., SUAREZ, A., AND LLORET, J. Mobile sensing systems. *Sensors 13*, 12 (2013), 17292.

[42] MAITI, A., JADLIWALA, M., HE, J., AND BILOGREVIC, I. (smart) watch your taps: side-channel keystroke inference attacks using smartwatches. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers* (2015), ACM, pp. 27–30.

[43] MENG, W., LEE, W. H., MURALI, S., AND KRISHNAN, S. Charging me and i know your secrets!: Towards juice filming attacks on smartphones. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security* (New York, NY, USA, 2015), CPSS '15, ACM, pp. 89–98.

[44] MICHALEVSKY, Y., BONEH, D., AND NAKIBLY, G. Gyrophone: Recognizing speech from gyroscope signals. In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, Aug. 2014), USENIX Association, pp. 1053–1067.

[45] MILETTE, G., AND STROUD, A. *Professional Android sensor programming*. John Wiley & Sons, 2012.

[46] MILUZZO, E., VARSHAVSKY, A., BALAKRISHNAN, S., AND CHOUDHURY, R. R. Tapprints: Your finger taps have fingerprints. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (New York, NY, USA, 2012), MobiSys '12, ACM, pp. 323–336.

[47] MOHAMED, M., SHRESTHA, B., AND SAXENA, N. Smashed: Sniffing and manipulating android sensor data for offensive purposes. *IEEE Transactions on Information Forensics and Security PP*, 99 (2016), 1–1.

[48] MOLAY, D., KOUNG, F.-H., AND TAM, K. Learning characteristics of smartphone users from accelerometer and gyroscope data.

[49] MUKHERJEE, S., AND SHARMA, N. Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology 4* (2012), 119–128.

[50] MURPHY, K. P. Naive bayes classifiers. *University of British Columbia* (2006).

[51] NARAIN, S., SANATINIA, A., AND NOUBIR, G. Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks* (New York, NY, USA, 2014), WiSec '14, ACM, pp. 201–212.

[52] NGUYEN, T. Using unrestricted mobile sensors to infer tapped and traced user inputs. In *Information Technology - New Generations (ITNG), 2015 12th International Conference on* (April 2015), pp. 151–156.

[53] OWUSU, E., HAN, J., DAS, S., PERRIG, A., AND ZHANG, J. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications* (2012), ACM, p. 9.

[54] PANDA, M., AND PATRA, M. R. Network intrusion detection using naive bayes. *International journal of computer science and network security 7*, 12 (2007), 258–263.

[55] PARK, B.-W., AND LEE, K. C. *The Effect of Users' Characteristics and Experiential Factors on the Compulsive Usage of the Smartphone*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[56] PEIRAVIAN, N., AND ZHU, X. Machine learning for android malware detection using permission and api calls. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on* (2013), IEEE, pp. 300–305.

[57] PETRACCA, G., SUN, Y., JAEGER, T., AND ATAMLI, A. Audroid: Preventing attacks on audio channels in mobile devices. In *Proceedings of the 31st Annual Computer Security Applications Conference* (New York, NY, USA, 2015), ACSAC 2015, ACM, pp. 181–190.

[58] PING, D., SUN, X., AND MAO, B. Textlogger: Inferring longer inputs on touch screen using motion sensors. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (New York, NY, USA, 2015), WiSec '15, ACM, pp. 24:1–24:12.

[59] PONGALIUR, K., ABRAHAM, Z., LIU, A. X., XIAO, L., AND KEMPEL, L. Securing sensor nodes against side channel attacks. In *High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE* (2008), IEEE, pp. 353–361.

[60] POSLAD, S. *Ubiquitous computing: smart devices, environments and interactions*. John Wiley & Sons, 2011.

[61] ROESNER, F., KOHNO, T., MOSHCHUK, A., PARNO, B., WANG, H. J., AND COWAN, C. User-driven access control: Rethinking permission granting in modern operating systems. In *2012 IEEE Symposium on Security and Privacy* (2012), IEEE, pp. 224–238.

[62] SAHS, J., AND KHAN, L. A machine learning approach to android malware detection. In *Intelligence and security informatics conference (eisic), 2012 european* (2012), IEEE, pp. 141–147.

[63] SCHLEGEL, R., ZHANG, K., ZHOU, X.-Y., INTWALA, M., KAPADIA, A., AND WANG, X. Soundcomber: A stealthy and context-aware sound trojan for smartphones. *NDSS 11* (2011), 17–33.

[64] SCHMIDT, A.-D., BYE, R., SCHMIDT, H.-G., CLAUSEN, J., KIRAZ, O., YUKSEL, K. A., CAMTEPE, S. A., AND ALBAYRAK, S. Static analysis of executables for collaborative malware detection on android. In *Communications, 2009. ICC'09. IEEE International Conference on* (2009), IEEE, pp. 1–5.

[65] SHABTAI, A., KANONOV, U., ELOVICI, Y., GLEZER, C., AND WEISS, Y. "andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems 38*, 1 (2012), 161–190.

[66] SHEN, C., PEI, S., YANG, Z., AND GUAN, X. Input extraction via motion-sensor behavior analysis on smartphones. *Computers & Security 53* (2015), 143–155.

[67] SHUKLA, D., KUMAR, R., SERWADDA, A., AND PHOHA, V. V. Beware, your hands reveal your secrets! In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2014), CCS '14, ACM, pp. 904–917.

[68] SIMON, L., AND ANDERSON, R. Pin skimmer: Inferring pins through the camera and microphone. In *Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices* (New York, NY, USA, 2013), SPSM '13, ACM, pp. 67–78.

[69] SMALLEY, S., AND CRAIG, R. Security enhanced (se) android: Bringing flexible mac to android. In *NDSS* (2013), vol. 310, pp. 20–38.

[70] SON, Y., SHIN, H., KIM, D., PARK, Y.-S., NOH, J., CHOI, K., CHOI, J., KIM, Y., ET AL. Rocking drones with intentional sound noise on gyroscopic sensors. In *USENIX Security* (2015), pp. 881–896.

[71] SPREITZER, R. Pin skimming: Exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices* (New York, NY, USA, 2014), SPSM '14, ACM, pp. 51–62.

[72] STRIKOS, A. A. A full approach for intrusion detection in wireless sensor networks. *School of Information and Communication Technology* (2007).

[73] SUBRAMANIAN, V., ULUAGAC, S., CAM, H., AND BEYAH, R. Examining the characteristics and implications of sensor side channels. In *Communications (ICC), 2013 IEEE International Conference on* (June 2013), pp. 2205–2210.

[74] SUN, M., ZHENG, M., LUI, J. C. S., AND JIANG, X. Design and implementation of an android host-based intrusion prevention system. In *Proceedings of the 30th Annual Computer Security Applications Conference* (New York, NY, USA, 2014), ACSAC '14, ACM, pp. 226–235.

[75] TIPPENHAUER, N. O., PÖPPER, C., RASMUSSEN, K. B., AND CAPKUN, S. On the requirements for successful gps spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), ACM, pp. 75–86.

[76] ULUAGAC, A., SUBRAMANIAN, V., AND BEYAH, R. Sensory channel threats to cyber physical systems: A wake-up call. In *Communications and Network Security (CNS), 2014 IEEE Conference on* (Oct 2014), pp. 301–309.

[77] ULUAGAC, S., RADHAKRISHNAN, S. V., CORBETT, C. L., BACA, A., AND BEYAH, R. A passive technique for fingerprinting wireless devices with wired-side observations. In *2013 IEEE Conference on Communications and Network Security (CNS) (IEEE CNS 2013)* (Washington, USA, Oct. 2013), pp. 471–479.

[78] WANG, X., YANG, Y., ZENG, Y., TANG, C., SHI, J., AND XU, K. A novel hybrid mobile malware detection system integrating anomaly detection with misuse detection. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services* (New York, NY, USA, 2015), MCS '15, ACM, pp. 15–22.

[79] WU, D.-J., MAO, C.-H., WEI, T.-E., LEE, H.-M., AND WU, K.-P. Droidmat: Android malware detection through manifest and api calls tracing. In *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on* (2012), IEEE, pp. 62–69.

[80] WU, W.-C., AND HUNG, S.-H. Droiddolphin: A dynamic android malware detection framework using big data and machine learning. In *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems* (New York, NY, USA, 2014), RACS '14, ACM, pp. 247–252.

[81] XU, Z., BAI, K., AND ZHU, S. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks* (New York, NY, USA, 2012), WISEC '12, ACM, pp. 113–124.

[82] XU, Z., AND ZHU, S. Semadroid: A privacy-aware sensor management framework for smartphones. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy* (New York, NY, USA, 2015), CODASPY '15, ACM, pp. 61–72.

[83] YE, N., ET AL. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop* (2000), vol. 166, West Point, NY, p. 169.

[84] YE, Y., WANG, D., LI, T., AND YE, D. Imds: Intelligent malware detection system. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (2007), ACM, pp. 1043–1047.

[85] YU, Y., WANG, J., AND ZHOU, G. The exploration in the education of professionals in applied internet of things engineering. In *Distance Learning and Education (ICDLE), 2010 4th International Conference on* (Oct 2010), pp. 74–77.

[86] YU, Z., AND TSAI, J. J. A framework of machine learning based intrusion detection for wireless sensor networks. In *Sensor Networks, Ubiquitous and Trustworthy Computing, 2008. SUTC'08. IEEE International Conference on* (2008), IEEE, pp. 272–279.

[87] ZHUANG, L., ZHOU, F., AND TYGAR, J. D. Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur. 13*, 1 (Nov. 2009), 3:1–3:26.

A Theoretical Foundation

A.1 Markov Chain-Based Detection

For the Markov Chain detection model, 6thSense observes the changes of condition of a set of sensors as a variable which changes over time. The condition of a sensor indicates whether the sensor value is changing or not from a previous value in time. For a specific time, t, 6thSense considers the combination of all the sensors' condition in the smart device as the state of our model. As 6thSense considers change in a sensor's condition as binary output (1 or 0, where 1 denotes that sensor value is changing from previous instance and 0 denotes that sensor value is not changing), the number of total states of in the detection model will be exponents of 2. For example, if we consider the total number of sensors in set S is 10, the number of states in our Markov Chain will be 2^{10} or 1024 and the states can be represented as a 10 bit binary number where each bit will represent the state of a corresponding sensor. Assume that p_{ij} denotes the probability that the system in a state j at time $t+1$ given that system is in state i at time t. If we have n number of sensors and $m = 2^n$ states in our model, the transition probability matrix of this Markov Chain can be constructed by observing the transitions from one state to another state for a certain time. Assume that 6thSense's states are $X_0, X_1, \ldots, X_T$ at a given time $t = 0, 1, \ldots, T$. We can represent the transition probability matrix [83]

with $P_{ij} = \frac{N_{ij}}{N_i}$ with N_{ij} = the number of transitions from X_t to X_{t+1}, where X_t in state i and X_{t+1} in state j; N_i = the number of transitions from X_t to X_{t+1}, where X_t in state i and X_{t+1} in any other state. The initial probability distribution of this Markov Chain can be as follows [33]:

$$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \dots & \dots & q_m \end{bmatrix} \quad (11)$$

where, q_m is the probability that the model is in state m at time 0. The probability of observing a sequence of states $X_1, X_2, \dots, X_T$ at a given time $1, \dots, T$ can be computed using the following equation:

$$P(X_1, X_2, \dots, X_T) = q_{x1} \prod_{2}^{T} P_{X_{t-1} Xt} \quad (12)$$

As described earlier in Section 5, for 6thSense, we use a modified version of the general Markov Chain model. Instead of predicting the next state, 6thSense determines the probability of a transition occurring between two states at a given time.

A.2 Naive Bayes Based Detection

Naive Bayes model is a simple probability estimation method which is based on Bayes' method. The main assumption of the Naive Bayes detection is the presence of a particular particular sensor condition in a task/activity has no influence over the presence of any other feature on that particular event.

Assume $p(x_1, x_2)$ is the general probability distribution of two events x_1, x_2. Using the Bayes rule, we can have $p(x_1, x_2) = p(x_1|x_2)p(x_2)$ where, $p(x_1|x_2) =$ Probability of event x_1 given that event x_2 will happen. Now, with c, we can rewrite this formula as $p(x_1, x_2|c) = p(x_1|x_2, c)p(x_2|c)$. If c is sufficient enough to determine the probability of event x_1, we can state that there is conditional independence between x_1 and x_2 [54]. So, we can rewrite the first part as $p(x_1|x_2, c) = p(x_1|c)$, which then modifies the formula as follows:

$$p(x_1, x_2|c) = p(x_1|c)p(x_2|c) \quad (13)$$

6thSense considers users' activity as a combination of n number of sensors. Assume X is a set which represents current conditions of n number of sensors. We consider that conditions of sensors are conditionally independent (See Section 4.2), which means a change in one sensor's working condition (i.e., on/off states) has no effect over a change in another sensor's working condition. As explained earlier, the probability of executing a task depends on the conditions of a specific set of sensors. So, in summary, although one sensors' condition does not control another sensor's condition, overall the probability of executing a specific task depends on all the sensors' conditions. As an example, if a person is walking

with his smartphone in his hand, the motion sensors (accelerometer and gyroscope) will change. However, this change will not force the light sensor or the proximity sensor to change its condition. Thus, sensors in a smartphone change their conditions independently, but execute a task together. We can have a generalized model for this context-aware detection [49] as follows:

$$p(X|c) = \prod_{i=1}^{n} p(X_i|c) \quad (14)$$

In 6thSense's context-aware activity-oriented detection model, we have a set of training data for users' activities. Assume that B represents a set which denotes m numbers of user activities. We can determine the probability of a dataset X to be classified as a user activity using the following equation:

$$P(B_i|X) = \frac{P(X|B_i)P(B_i)}{P(X)}, \quad (15)$$

where $i = 1, 2, \dots, m$. As the sum of all the conditional probabilities for X will be 1, we can have the following equation, which then will lead to Equation 17 [36]:

$$\sum_{i=1}^{m} P(B_i|X) = 1. \quad (16)$$

$$P(B_i|X) = \frac{P(X|B_i)P(B_i)}{\sum_{i=1}^{m} P(X|B_i)P(B_i)}. \quad (17)$$

This calculated conditional probability then is used to determine the benign user activity or malicious attacks in 6thSense. In this way, 6thSense computes the probability of occurring an activity over a certain period of time.

6thSense divides the sensor data into smaller time values (1 second) and calculates the probability of each instance to infer the user activity. The calculated probability of each second data is then used in the expected value to calculate the total probability. As such, the probability of the first instance is p_1 with a value of a_1, the probability of the second instance is p_2 with a value of a_2 and so on up to the value a_n. Then, the expected value can be calculated by the following formula:

$$E[N] = \frac{a_1 p_1 + a_2 p_2 + a_3 p_3 + \dots \dots + a_n p_n}{a_1 + a_2 + \dots \dots + a_n}. \quad (18)$$

As all the values of $a_1, a_2, \dots \dots, a_n$ are equally likely, this expected value becomes a simple average of the cumulative probability of each instance. In this way, 6thSense infers the user activity by setting up a configurable threshold value and checking whether the calculated value is higher than the threshold or not. If it is lower than the threshold value, 6thSense concludes that the malicious activity is occurring in the smart device.