5-2018

Applications of Context-Aware Systems in Enterprise Environments

Oyindamola D. Oluwatimi
Purdue University

APPLICATIONS OF CONTEXT-AWARE SYSTEMS IN ENTERPRISE ENVIRONMENTS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Oyindamola D. Oluwatimi

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Elisa Bertino, Chair

Department of Computer Science

Dr. Walid Aref

Department of Computer Science

Dr. Sonia Fahmy

Department of Computer Science

Dr. Ninghui Li

Department of Computer Science

Approved by:

Dr. Voicu Popescu by Dr. William J. Gorman

Head of the Department Graduate Program

This work is dedicated to all those who doubted me. It has been, and will continue to be, a pleasure to rise above your imposed limitations and my weaknesses.

ACKNOWLEDGMENTS

I would like to thank Professor Elisa Bertino who provided guidance throughout my time as a student at Purdue. Through her guidance, I have come to fully realize the sensation of achieving success "against all odds." I also thank my dissertation committee, Professor Walid Aref, Professor Sonia Fahmy, and Professor Ninghui Li, for their contributions to my academic career.

Personal thanks to Serina Woods and Rachel Scarlett for providing emotional support in difficult times. Special thanks to Adrian Thomas and Dr. Zenephia Evans for keeping it 100 and helping me navigate through all the precarious situations at a predominantly white institution as a Nigerian black man. I highly appreciate my best friend Adam Abadir for being my secret rival who unknowingly motivated me into putting significant effort into my education. Thanks to Jeff Avery, Chris Gutierrez, Abram Magner, and Vivek Patel for their valuable feedback and support. Finally, I thank the Dozoretz National Institute for Mathematics and Applied Sciences, the GEM Consortium, the Purdue Doctoral Fellowship, and the VACCINE HS-STEM Career Development for allowing me to focus on my academics by providing financial support throughout my undergraduate and graduate career.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABBREVIATIONS

BYOD	Bring-Your-Own-Device
CAS	Context-Aware System
CI	Contextual Information
COPE	Corporate Owned, Personally-Enabled
EED	Enterprise-Enabled Device

ABSTRACT

Oluwatimi, Oyindamol D. Ph.D., Purdue University, May 2018. Applications of Context-Aware Systems in Enterprise Environments. Major Professor: Elisa Bertino.

In bring-your-own-device (BYOD) and corporate-owned, personally enabled (COPE) scenarios, employees' devices store both enterprise and personal data, and have the ability to remotely access a secure enterprise network. While mobile devices enable users to access such resources in a pervasive manner, it also increases the risk of breaches for sensitive enterprise data as users may access the resources under insecure circumstances. That is, access authorizations may depend on the context in which the resources are accessed. In both scenarios, it is vital that the security of accessible enterprise content is preserved.

In this work, we explore the use of contextual information to influence access control decisions within context-aware systems to ensure the security of sensitive enterprise data. We propose several context-aware systems that rely on a system of sensors in order to automatically adapt access to resources based on the security of users' contexts. We investigate various types of mobile devices with varying embedded sensors, and leverage these technologies to extract contextual information from the environment. As a direct consequence, the technologies utilized determine the types of contextual access control policies that the context-aware systems are able to support and enforce. Specifically, the work proposes the use of devices pervaded in enterprise environments such as smartphones or WiFi access points to authenticate user positional information within indoor environments as well as user identities.

1. INTRODUCTION

Mobile devices are becoming a mandatory aspect of the daily lives of users. These devices have powerful functionality granting users various abilities through installing and executing applications which are abilities similar to desktop computing platforms. With such abilities, users are, for example, able to compose documents, set calendar reminders, and complete other daily tasks. Unlike their desktop counterparts, these devices' form factors allow mobility with respect to embedded sensors and network connectivity. With the capacity to have permanent Internet connection via cellular or WiFi infrastructures, mobile devices enable pervasive access. Users are able to access emails, access remote networks, and manage and download private, confidential, or secret data (e.g., banking data or medical data) in *any* context.

The capabilities of such mobile devices, as well as their increasing affordability and mobility, have enabled enterprises to leverage them in the workplace. This creates two main scenarios: bring-your-own device (BYOD) and corporate-owned, personally enabled (COPE) device. In the BYOD scenario, employees use their own personal mobile device also for work purposes. Conversely, in the COPE scenario, it is the enterprise that provides devices to its employees. In both scenarios, that we collectively refer to as enterprise-enabled device (EED) scenario, the same device is used for personal and business purposes. Such a dual use makes it possible for enterprises to rely on a mobile Information Technology (IT) infrastructure. Such an infrastructure allows employees to remotely access enterprise content that otherwise would not be accessible outside of the enterprise setting. Despite interesting opportunities provided by mobile devices, access to such data and resources, however, may be contingent upon the context in which they are accessed. Accessing enterprise content, whether locally or remotely, in insecure contexts increases the risk of sensitive information leakage. A user situated in a public café may succumb to a shoulder surfing attack by a random

passerby when the user is viewing sensitive medical data via his/her device. If accessing remote content via the device, sensitive information such as banking data could be captured by an adversary sniffing the café's public WiFi network. For the sake of ensuring the security of personal and enterprise content, contextual factors that exist within the physical and computing realms must be considered while evaluating access control requests.

Consider an enterprise organization in which an employee, carrying her smartphone on her person, is attending a confidential meeting. It may be required that the employee cannot access the device's microphone to prevent, whether it is unintentional or malicious, audio recording of the confidential meeting's conversation. In the same scenario, the employee is handed a draft of a patent document that is to remain in that room, and therefore, the document should not be visually recorded using the smartphone's camera. The capabilities of the employee and her smartphone in the above scenario are contingent upon, although not immediately obvious, various environmental factors that exist in the physical and computing realms such as location and mobile applications that have the potential of executing, respectively. As a consequence, policies and systems that do not incorporate contextual parameters or restrictions are not suitable for the described circumstances, as well as enterprise environments in general in which mobile devices are integrated into IT infrastructures. Below we present definitions of a context and a context-aware (CAS) system that will be used throughout this work.

Context. Various authors have attempted to construct a definition of the term context, but some definitions are not flexible or scalable enough to be applicable to a breath of scenarios old and new. In this work, we adopt the following generalized definition of context [1]: *"any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves".*

Context-Aware System. Although Dey [1] also defines context-awareness for systems, it does not reflect the *sentient* characteristic of an aware entity. As such, we adopt the following definition of context-awareness [2]: *"context-aware systems are able to adapt their operations to the current context without explicit user intervention"*.

In general, CASs operate in pervasive computing environments in which certain applications require contextual information (CI) in order to provide data, services, or resources. The objective of such systems is to maximize the usability of data and services by incorporating environmental factors such that access is automatically granted without explicit user intervention. There are many unique problems, previously and newly formulated, that CASs can be applied to [3–7], some of which we now briefly describe. For example, although it is understandable why the progression in elevator technology has become stagnant as a result of the simplicity of the task, recent system designers have envisioned that future elevators will be context-aware. Using a system of sensors, elevators will be able to automatically detect when users are heading towards or waiting to get into an elevator [8]. Movie information services such as Fandango inform users of the next available movie showings in theatres that are within the users' vicinity. In assisted living, some patients are the only resident of their household. It is vital that caretakers are immediately and remotely alerted of abnormal behavior in patients once detected [9]. While the previous example applications define context within spatial and temporal parameters, context can be quite vague and is application dependent. Context can comprise of information related to altitude, temperature, humidity, ambient light, radio, motion, audio, etc. In this dissertation, we investigate the feasibility of using various technologies to support context-aware access control, including radio-, motion-, and audio-based technologies.

The proliferation and technological advancement of wireless networking and sensor technologies – such as smartphones or WiFi access points – enable portable mobile devices to be used in CASs. In recent years, smartphones' continually advancing operating systems (OS) and hardware capabilities have allowed capabilities not present

in their desktop counterparts. For example, many smartphone manufacturers have leveraged these devices' mobile form factor by integrating embedded sensors and network connectivity peripherals including, but not limited to, Bluetooth, GPS, accelerometer, and near-field-communication. In addition, various static and dynamic CI can be extracted from mobile devices such as hardware and software configurations which also can be utilized by CASs to adapt the systems' operations automatically to the processed information. Some other systems utilize information extracted from wireless devices pervaded in enterprise environments such as WiFi access points to not only localize a user, but to also detect the presence of other users. Specifically, proximity is yet another contextual parameter that can be leveraged to support automated, CASs.

In summary, context can be divided into three different categories [10]:

- User Context: it refers to any information related to the user, including user dynamic information (user current and historical location, user current and historical activity, user current emotion, relationships or contact with colleagues or friends and so on) and user static information (user personal information, user habit, user preference, and so on).

- Physical Context: it contains environmental physical information (lighting, noise, temperature, humidity level, traffic conditions and so on) and device physical information (device battery, memory, the size and type of screen, terminal's OS, input and output method, nearby resources such as printers, and so on).

- Network Context: network capacity, connectivity, costs of computing and communication, bandwidth and so on.

The work proposed here explores the use of various types of mobile devices to extract and process primarily user contextual information in CASs and apply such information to access control and data isolation techniques in order to secure enterprise content. Access control is a security technique to regulate the sharing of resources

among entities in a computing environment. In terms of access control, context-aware systems aim to secure access to sensitive enterprise resources by adapting their access authorizations to the current context. For example, following the presented enterprise scenario, an access control policy for the employee's smartphone device would state in some manner or form "from 1PM to 2PM, device microphone is inaccessible". Such policy would prevent the device from recording the conversation during the confidential meeting. Containerization is another security technique, but it ensures the separation of enterprise content from all other non-enterprise related content on an end-user's mobile device (e.g., smartphone or laptop). With respect to containerization, CASs aim to automatically deploy, manage, and update secure containers that is dependent on CI that enterprises deem pertinent to the security of the containers and their content. For example, again following the same scenario, a policy associated with the management of containers would state in some manner that "a secure container that does not include a camera application must be deployed to the device when the employee is at the meeting room location". Thus, completely eliminating the possibility of visually documenting confidential information via the employee's device.

1.1 Dissertation Statement

It is possible to apply context-aware systems, supported by various types of mobile devices, to enterprise environments to secure enterprise content from (benign or malicious) entities whether external or internal to the enterprise organization.

Specifically, the work focuses on systems that utilize access control and containerization techniques on mobile devices in EED scenarios. We aim to answer three main questions:

1. how do we capture contextual information?
2. how do we incorporate contextual constraints into access control policies?

3. how do we enforce contextual access control policies?

We address these questions in several approaches throughout this work, which we briefly highlight in the next section. Each approach utilizes different techniques or technologies to extract various CI from the environment and applies them to access control or data isolation techniques. Further high-level details of each approach is provided below.

1.2 Context-Aware Access Control Systems

In what follows, we briefly introduce various security threats that enterprises may potential encounter in EED scenarios, and subsequently propose context-aware access control systems to address those threats.

1.2.1 Location-Based Access Control Systems for Mobile Devices

Mobile Android applications often have access to sensitive data and resources on the user's device. Misuse of this data by malicious applications may result in privacy breaches and sensitive data leakage. The problem arises from the fact that Android users do not have control over the application capabilities once the applications have been granted the requested privileges upon installation. In many cases, however, whether an application is granted a privilege depends on the specific user context.

The need for configurable device policies based on context extends from high profile employees to regular smartphone users. For example, government employers, such as in national labs [11], restrict their employees from bringing any camera-enabled device to the workplace, including smartphones, even though employees might need to have their devices with them at all times as their devices may contain data and services they might need at any time. With context-based device access control policies, employees may be allowed to use smartphones as they can disable all applications from using the camera and any device resources and privileges that employers restrict while at work, while the user's device can retain all its original privileges outside the

work area. Context-based policies are also a necessity for politicians and law enforcement agents who would need to disable camera, microphone, and location services from their devices during confidential meetings while retaining these resources back in non-confidential locations. With context-based policies, users can specify when and where their applications can access their device data and resources, which reduces the hackers' chances of stealing such sensitive data since end-users do not have control of the actions taken or exercised capabilities by possibly malicious applications especially in context-sensitive circumstances.

Previous work on security for mobile operating systems focuses on restricting applications from accessing sensitive data and resources, but mostly lacks efficient techniques for enforcing those restrictions according to fine-grained contexts that differentiate between closely located subareas [12]. Moreover, most of this work has focused on developing policy systems that do not restrict privileges per application and are only effective system-wide [13]. Also, existing policy systems do not cover all the possible ways in which applications can access user data and device resources. Finally, existing location-based positioning systems are not accurate enough to differentiate between nearby locations without extra hardware or location devices [12, 14, 15]. In most cases, such systems assume the context as given without providing or evaluating context detection methods for mobile devices [12, 16].

End-users need a context-based access control mechanism by which privileges can be dynamically granted or revoked to applications, on a per-application basis, based on the specific context of the user. We propose such an access control mechanism, which we refer to as Context-Based Access Control (CBAC). Our implementation of context differentiates between closely located sub-areas within the same location. We have modified the Android operating system so that context-based access control policies can be specified by end-users and enforced by our system.

1.2.2 Securing Remote Enterprise Content via Proximity-Based Access Control

Enterprise organizations have adopted context-aware systems that leverage proximity -based access control (PrBAC) to mitigate threats of information leakage. That is, access control decisions are not solely based on the requesting user's location, but also on the location of other users in the physical space. Consider an enterprise organization in which an employee is allowed to access a confidential financial document, but only if the access is executed within the supervisor's office. An example of a PrBAC policy would be to require the presence of the supervisor, in the supervisor's office, for the employee to be able to view the confidential document.

In a previous work [17], we introduced a secure, automated PrBAC architecture and prototype system that we referred to as the Context-Aware System to Secure Enterprise Content (CASSEC). While our system was agnostic with respect to the technological choices for detecting physical proximity, we had provided a simple implementation of the complete CASSEC architecture. We utilized Bluetooth and WiFi devices, which are widely used in enterprise environments, to address the occupancy detection problem [18] and support two practical proximity-based scenarios often encountered in enterprise settings: Separation of Duty and Absence of Other Users. The first scenario is achieved by using Bluetooth MAC addresses of nearby occupants as authentication tokens. The second scenario exploits the interference of WiFi received signal strength when an occupant crosses the line of sight. Regardless of the scenario, information about the occupancy of a particular location is periodically extracted to support continuous authentication. The proposed access control system allows end-users to automatically access enterprise content stored in a remote server, and to the best of our knowledge, our approach is the first to incorporate WiFi signal interference caused by occupants as part of a PrBAC system.

In this dissertation, we also consider the security implications of an enterprise's employees relying on CASSEC's position-based services to access remote enterprise

content via endpoint devices. In general, the sensors of a context-aware system extract contextual information from the environment and relay that information to higher-level processes of the system so to influence the system's control decisions. However, an adversary can maliciously influence such controls indirectly by manipulating the environment in which the sensors are monitoring, thereby granting privileges the adversary would otherwise not normally have. To address such context monitoring issues, we extend CASSEC by incorporating sentience-like constructs, which enable the emulation of "confidence", into our PrBAC model to grant the system the ability to make more inferable decisions based on the *degree of reliability* of extracted contextual information. In CASSEC 2.0, we evaluate our confidence constructs by implementing two new authentication mechanisms. Co-proximity authentication employs our time-based challenge-response protocol, which leverages Bluetooth Low Energy beacons as its underlying occupancy detection technology. Biometric authentication relies on the accelerometer and fingerprint sensors to measure behavioral and physiological user features to prevent unauthorized users from using an authorized user's device. We provide a feasibility study demonstrating how confidence constructs can improve the decision engine of context-aware access control systems.

1.3 Context-Aware Containerization Systems

EED scenarios enable employees to utilize their smartphone mobile device for both personal and enterprise purposes, thereby allowing sensitive enterprise content to be stored and accessed on end-users' devices *anywhere* and *anytime*. However, security is an important, and the most significant, barrier to wide adoption of such dual-use scenarios. In 2016, conducted research found that the top two security concerns of cybersecurity practitioners, such as enterprise IT admins, were data leakage (72%) and unauthorized access to enterprise resources (56%) [19]. In fact, nearly one out of five organizations (21%) experienced a security breach via EED vectors. IT admins attempt to mitigate such threats by employing Enterprise Mobility Man-

agement (EMM) systems which administer secure containers (e.g., work persona) to end-users' devices [20], but enterprises continue to suffer due to EMM systems lack of or ineffective access control and monitoring solutions.

We identify a few shortcomings of contemporary EMM systems. First, EMM systems do not consider the context in which personas are employed. EMM systems, such as Samsung KNOX [20], do not provide enterprises a means to specify or enforce contextual constraints to control access to or influence the behavior of personas. Second, modern EMM systems assume that each end-user uses her device for only one enterprise. We argue that EMM systems need to support *multi-enterprise* environments, as end-users may interface with a variety of first/third-parties with potentially conflicting contextual access control policies. To limit risks of unauthorized access, it is imperative that organizations employ secure means of *contextual authentication and authorization* to protect enterprise content after it is downloaded to end-users' devices.

To address these shortcomings, we present our position-based, **M**ulti-**E**nterp**R**ise **C**ontainerization (MERC) architecture for EED security. The MERC architecture leverages positional data to grant context-aware capabilities to container-based systems. We grant enterprises the ability of defining location- and proximity-based conditions that must be met in order for users to securely access enterprise container content. First, we provide a scalable location-based scheme that allows multiple enterprise context-aware systems to securely coexist and activate policies and personas on an end-user's device. Second, the MERC incorporates proximity-based constraints to modify a persona's behavior. We evaluate our prototype using preexisting infrastructures, and our experimental results show that MERC is an effective and practical EED security solution for context-based containerization.

1.4 Document Structure

The rest of this document discusses the above topics in further detail. Chapter 2 presents basic concepts and surveys the state-of-the-art in mobile operating systems, positioned-based systems, and access control systems that lay the foundation of this work. The first approach in Chapter 3 addresses the issue of localizing the user using client-side technology and adapting applications' access to client-side content depending on the user's location and time of access. The second approach in Chapter 4 addresses the problem of localizing a user as well as detecting the proximity of other users *without* solely relying on end-users' devices to determine positional information, and then utilizing such information in access control requests to remote enterprise content. The third approach in Chapter 5 addresses the issue of applying proximity-based constraints to the management of end-user devices via mobile containerization techniques and technologies when end-users may be employed by or consult for multiple enterprises. Chapter 6 concludes this dissertation with questions and insights directed at enterprises so that enterprises could implement or employ appropriately solutions for their particular EED scenarios.

2. BACKGROUND AND RELATED WORK

In this chapter, we present basic concepts and survey the state-of-the-art in mobile operating systems, positioned-based systems, and access control systems in order to understand the work.

2.1 Android

Android is a Linux-based, mobile phone platform designed with a multi-layered security infrastructure [21]. Loaded on top of the Linux kernel are the System Libraries, Android Runtime, and Application Framework software layers (Figure 2.1). Each application, which is assigned a unique user ID (UID), is given a dedicated part of the file system for its own data, and executes in a separate Dalvik Virtual Machine, thus creating an application sandbox. Along with Linux's discretionary access control mechanism, Android includes a fine-grained permission system that determines the set of device resources an application has access to. An application's permissions, which can be extracted from its AndroidManifest.xml file, are also associated with its UID. At the time of an application installation, users have to either grant all the requested permissions to proceed with the installation of the APK, or cancel the installation completely. As of Oreo (API 26), there are currently over 150 application developer permissions.

An *Intent* is an Android messaging facility to support inter-component communication. A component (i.e., Android activity, service, content provider, or broadcast receiver) sends an Intent message to the OS, which basically specifies the intent of starting, accessing, or requesting information from a particular component, including ones from another application.

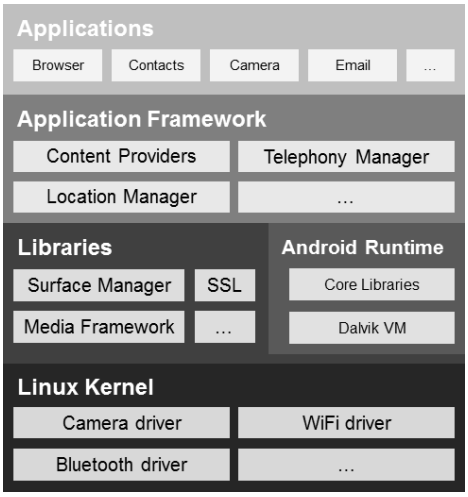


Fig. 2.1. Android software stack.

2.2 Position-based Services

There is a variety of technologies that address the localization problem, that is, to determine and retrieve a user's location. Generally, each positioning system (i.e., PBS) has at least two separate hardware components, a transmitter and a receiver to send and receive signals, respectively [22]. The receiver analyzes one of three characteristics of the received signal which are: angle-of-arrival (AoA), received-signal strength (RSS), and time of arrival (ToA).

PBSs have the ability to detect the current position of user devices, and such services are important in variety of settings including access control enforcement in EED scenarios [17]. PBSs vary with respect to many parameters, such as the position technologies on which they are based, security, privacy, affordability, resource requirements (e.g., memory or power consumption), and precision level of positional data, and therefore have their inherent advantages and disadvantages. For example, geofencing PBSs are able to place a user within a predefined area such as with the use of GPS, which is the most widely used positioning tool that uses the propagation time of signals (i.e., ToA) from satellites to compute the position of a receiver anywhere on Earth. Microlocation PBSs can locate a user with high accuracy such as with the use

of Ultra Wide-Band radios to provide an accuracy as high as 10 cm by calculating AoA and ToA [23]. Other positioning techniques based on different technologies include Infrared (IR), Radio Frequency (RF), Radio Frequency Identification (RFID) [24], magnetic field [25], ultrasound [26], Bluetooth [27], and WiFi [7, 9, 28–30].

Bluetooth Low Energy (BLE) can also be used to retrieve a user's relative location in an energy efficient manner, and it has been employed by beaconing services [23]. By utilizing widely-used BLE-based beacon protocols (e.g., Apple's iBeacon, Google's Eddystone, and AltBeacon [23]), a beacon region or the proximity of other BLE-enabled beacon devices (e.g., smartphones) can be detected. Detection is achieved by periodically broadcasting beacons that are picked up by BLE-enabled devices. Each beacon protocol has a different beacon construction, but we utilize Google's Eddystone implementation as it is open source and incorporates features that we leverage in Chapter 4 and Chapter 5. The Eddystone UID is 16 bytes long and consists of two values: Namespace (10 bytes) and Instance (6 bytes). The Namespace value is a UUID (Universally Unique Identifier) that identifies a top-level beacon region. The Instance value identifies sub-beacon regions and can be constructed using any scheme. For example, semantically, a beacon construction could represent the *auditorium within building 10* (i.e., Instance) at NekSec's campus (i.e., Namespace). Google's beacon also provides two measurements. The distance measurement is an indicator of the proximity of one device to another which is determined based on the RSS value. The ranging measurement is an intuitive, user-friendly indicator of the distance between two devices which falls into one of the following ranges: *Immediate* (very close), *Near* (at a distance of 1-3m), *Far* (greater than 3m), or *Unknown* (the distance cannot be accurately determined). We also investigate other distance-bounding techniques. In particular, we investigate techniques that measure the time elapsed, i.e., the round trip time (RTT), during the exchange of packets between the transmitter and receiver. In Chapter 5, we implemented a distance-bounding system using BLE beacons as our underlying technology, which were programmed using Android's *android.bluetooth.le* APIs [31].

2.3 Context-Based Access Control Models

2.3.1 Overview

The role-based access control (RBAC) model is mainly used in enterprise settings to facilitate administration of access control polices [32]. In such settings, users are assigned different roles whereby each role is granted predefined access privileges to enterprise resources. Various access control models and systems have been proposed that use RBAC as a foundational paradigm, and some augment the model so that privileges associated with a role can only be exercised if contextual parameters are adhered to. In this section, we provide an overview of role-based access control models that incorporate CI into the decision-making process.

The most common extension is the inclusion of spatial constraints. GEO-RBAC is a spatially-aware RBAC model that defines the concept of spatial roles which allow an authorized user to assume a role (i.e., role enabling) and exercise its associated privileges (i.e., role activation) only if the user is at or within a designated location specified by physical coordinates [33]. LoT-RBAC and STARBAC are other augmented RBAC models that incorporates spatio-temporal constraints for role enabling and role activation [34, 35]. Unlike the previously mentioned models, the authors in [36–38] did not focus simply on spatial or temporal constraints, but rather designing an access control framework that is flexible enough to allow a variety of contextual parameters. Such models however are not implemented and therefore no enforcement mechanism has been developed to support these models.

For the purpose of applying those models to real implementations, Sandhu *et al.* proposed the notion of PEI (policy, enforcement, implementation) models that define a usable structure for creating an implementation of enforcement mechanisms [39]. Gupta *et al.* proposed a context profiling framework based on the surrounding environment captured by the mobile device sensors to estimate the familiarity of a place [16]. They used such context to create context profilers that are used to configure access control policies on mobile devices.

Proximity-based Access Control (PBAC) [40] is an access control model developed specifically for Smart-Emergency Environments that takes into account the user's proximity to a resource (e.g., a computer). PBAC was implemented using ultra-wide band RFID which calculated AoA and ToA to support automated access access control. Although the system did not require user intervention, active tags (worn by users) and mounted receivers had to be deployed to determine the tags position. Prox-RBAC, which extends GEO-RBAC, is a formal authorization model based on a notion of proximity [41]. That is, access control decisions are not solely based on the requesting user's location, but also on the location of other users in the physical space. Prox-RBAC incorporates elements of the $UCON_{ABC}$ usage control model [42]. Prox-RBAC was implemented using near-field communication (NFC) allowing a NFC phone to transmit signals to a NFC reader to lock and unlock a door, and although it provides high-integrity proof of location, it requires user intervention. Prox-RBAC has been further extended to incorporate a large variety of proximity constraints in addition to the spatial ones, namely attribute-based, social, cyber, and temporal proximity constraints [43]. Many systems, including Prox-RBAC and PBAC, inherently assume that every individual within a monitored space is trusted. Systems that are solely based on location tracking devices worn or held by users can be easily circumvented through collusion. For example, assume sensitive documents are stored within a restricted office that should only be accessible by a high-level employee such as a corporate CEO. The CEO will unlock the door with his/her tracking device (e.g., NFC), but a low-level employee can easily follow immediately behind prior to the door locking. By not initiating contact between the transmitter and the receiver, the system would be tricked into believing that no unauthorized personnel is occupying the restricted office. In another insecure scenario, an employee, assuming he/she was given a tracking device, can simply remove the device (e.g., active tag) so as to not be tracked. Neither Prox-RBAC or PBAC addressed a major security problem of a user obtaining an authorized user's phone, whether by theft or voluntary provision. Consequently, individuals may be able to circumvent the access control system via

collusion, allowing one individual to impersonate another individual by exchanging tracking devices. In addition, costs for deployment and management of these systems, and others used in similar architectures, remain significant and limit the widespread adoption of these systems.

2.3.2 Context-based Application Restrictions for Android

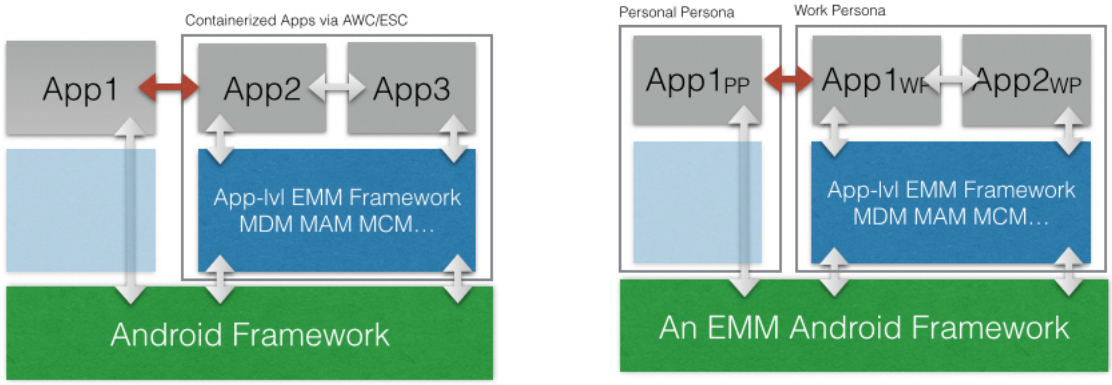
Limiting mobile applications' capabilities on the Android platform is not novel. Approaches have been proposed to support the restriction of device content access, but not in the context of EED scenarios [44]. Apex [45], AppFence [46], TISSA [47], MockDroid [48], and YAASE [49] have developed modifications to the Android OS in order to limit data leakage and restrict application permissions. Our work complements these techniques by adding more user controls and device restrictions (such as intent management) and ties these configurations to context-based policies that dynamically apply device restrictions. Our work also complements research efforts in protecting user and application data applied at the middleware and kernel layers of the Android OS, such as FlaskDroid [50], Moses [51], Saint [52], and TrustDroid [53].

2.4 Containerization

2.4.1 Overview

To support the dual use of mobile devices in EED scenarios, various mobile device containerization techniques were developed to secure accessible enterprise content as well as the privacy of employees [20]. In a broad sense, containerization primarily aims to secure a portion of a device's resources (e.g., application, storage, or network access) from other applications and systems running on the same device. When applied to EED scenarios, containerization isolates within the same device personal privacy-sensitive applications from enterprise and business applications.

In order to administer/manage secure containers to/on end-users' devices, enterprises use Enterprise Mobility Management (EMM) systems [20]. Existing EMM systems operate at either the application or platform level. The level at which these systems operate determines the types of containerization technologies they are able to leverage to isolate content. Application-level EMM systems create an application-level container supported by a non-native application-layer (EMM) framework (Figure 2.2(a)) that allow an application, or a set of trusted applications, to isolate itself and its data from other untrusted applications. Platform-level EMM systems, supported by a native EMM framework (Figure 2.2(b)), create multiple environments referred to as "personas" to isolate content so that trusted applications do not execute in a persona in which untrusted applications also reside. Independently of whether the container is implemented at the application or platform level, enterprises are able to configure policies for the container that modify the behavior of the applications (e.g., accessing data and system resources).



(a) Application-level Container.

(b) Platform-level Container.

Fig. 2.2. The interactions between two containerized applications and an untrusted application that exists outside of the secure area. The gray and red arrows represent permitted and non-permitted communication channels, respectfully.

2.4.2 Containerization on Android

There are several existing EMM-like systems that utilize multi-partition techniques to isolate private and corporate content. Contrary to our work, most fail to consider the context in which they are employed. Gupta et al [54] created a custom Android OS that supports dual-mode personas. In enterprise mode, the system could enforce policies that disable a subset of device communication peripherals, force the device to only communicate via an enterprise VPN, and ensure an encrypted external storage is utilized. TrustDroid proposed the use of domains and their isolation by monitoring and limiting data exchanged via IPC (Inter-Process Communication), files, databases, and socket connections.

IdentiDroid [6] is an application level privacy-enhancing tool based on Android that addresses the shortcomings of network anonymizers (e.g., Tor and Hotspot Shield). IdentiDroid uses configuration profiles which are analogous to personas that relocate application data when a profile is de/activated. Unlike other platform-level systems, IdentiDroid also contain application-level containerization through the utilization of several device content protection techniques. DroidARM [55] builds upon the work in IdentiDroid, but implemented on top of Android Lollipop. In this way, DroidARM is able to use native multi-user containerization to isolate applications and data as well as support EMM-based management features. Samsung KNOX 2.0 [20], the system AFW actually adopted its persona-based approach from, incorporates significantly more hardware and platform security than any other platform-level EMM system, thus providing stronger guarantees of preventing root attacks.

Cellrox [20] uses a lightweight virtualization technology called ThinVisor. ThinVisor resides in the OS and allows multiple instances of the Android OS, which Cellrox calls Virtual Mobile Instances (VMIs), using the same kernel. Cellrox's VMIs can be made portable allowing a VMI to be decoupled from the device and placed in another device without needing to reconfigure the VMI. None of the aforementioned solutions

support the activation of a container via location-based constraints or the restriction of the container's content via proximity-based constraints.

Several other solutions have been proposed that rely on the data tagging and information-flow tracking capabilities of TaintDroid [56]. In the work by Kodeswaran et al. [57], applications are classified as enterprise-related via parameters such as market source or developer signature. In addition, the data that is generated or processed by these applications are consequently tainted as enterprise. However, the proposed system does not incorporate contextual constraints. The work by Feth et al. [58] proposes a data-driven usage control architecture in which data is tainted by an enterprise-provided tag. The system supports context-aware policies by monitoring various device sensors such as location, WiFi, accelerometer, battery, Bluetooth, etc. Moses [51] isolates sensitive content from different personas by tainting data at the OS level with the name of the persona the data is associated with. Moses supports passive persona activation via GPS tracking. However, Moses, as well as the work proposed by Feth et al. [58], is not suitable for indoor environments as a result of GPS signal attenuation caused by construction materials. Furthermore, all of these solutions require significant modifications to the Android OS.

3. A LOCATION-BASED ACCESS CONTROL SYSTEM FOR MOBILE DEVICES

Mobile application developers leverage the computational and communication-based resources on mobile devices in order to incorporate new or enhanced services to their applications. However, the majority of these resources can collect sensitive data and may expose users to security and privacy risks if applications use them inappropriately and without the user's knowledge [56]. The threat arises when a device application acts maliciously and uses device resources to spy on the user or leak the user's personal data without the user's consent [59–61].

In this chapter, we propose a context-based access control (CBAC) mechanism for Android systems that allows smartphone users to set configuration policies over their applications' usage of device resources and services at different contexts. Through the CBAC mechanism, users can, for example, set restricted privileges for device applications when using the device at work, and device applications may re-gain their original privileges when the device is used at home. This change in device privileges is automatically applied as soon as the user device matches a pre-defined context of a user-defined policy. The user can also specify a default set of policies to be applied when the user is located in a non-previously defined location.

Configured policy restrictions are defined according to the accessible device resources, services, and permissions that are granted to applications at installation time. Such policies define which services are offered by the device and limit the device and user information accessibility. Policy restrictions are linked to context and are configured by the device user. We define context according to location and time. Location is determined basically through visible Wi-Fi access points and their respective signal strength values that allows us to differentiate between nearby sub-areas within the same work space, in addition to GPS and cellular triangulation coordi-

nates whenever available. We implement our CBAC policies on the Android operating system and include a tool that allows users to define physical places such as home or work using the captured Wi-Fi parameters. Users can even be more precise by differentiating between sub-areas within the same location, such as living rooms and bedrooms at home or meeting rooms and offices at work. Once the user configures the device policies that define device and application privileges according to context, the policies will be automatically applied whenever the user is within a pre-defined physical location and time interval.

Below, we first present a model overview of our access control framework in Section 3.1. Section 3.2 introduces our policy constructs and their classification followed by implementation and technical details in Section 3.3. Section 3.4 emphasizes our technique in managing CI and how we keep policy restrictions up-to-date with device location. Section 3.5 reports results of experiments to assess the accuracy of CI and the impact of policy restrictions on applications. We analyze the security of our approach in Section 3.6.

3.1 Model Overview

In this section, we present an overview of our access control framework through describing its components and the role of its entities.

Our framework consists of an access control mechanism that deals with access, collection, storage, processing, and usage of context information and device policies. To handle all the aforementioned functions, our framework design consists of four main components as shown in Figure 3.1.

The **Context Provider** (CP) collects the physical location parameters (GPS, Cell IDs, Wi-Fi parameters) through the device sensors and stores them in its own database, linking each physical location to a user-defined logical location. It also verifies and updates those parameters whenever the device is re-located.

The **Access Controller** (AC) controls the authorizations of applications and prevents unauthorized usage of device resources or services. Even though the Android OS has its own permission control system that checks if an application has privileges to request resources or services, the AC complements this system with more control methods and specific fine-grained control permissions that better reflect the application capabilities and narrow down its accessibility to resources. The AC enhances the security of the device system since the existing Android system has some permissions that, once granted to applications, may give applications more accessibility than they need, which malicious code can take advantage of. For example, the permission READ_PHONE_STATE gives privileged applications a set of information such as the phone number, the IMEI/MEID identifier, subscriber identification, phone state (busy/available), SIM serial number, *etc.*

The **Policy Manager** (PM) represents the interface used to create policies, mainly assigning application restrictions to contexts. It gives control to the user to configure which resources and services are accessible by applications at the given context provided by the CP. As an example, the user through the PM can create a policy to enable location services only when the user is at work during weekdays between 8 am and 5 pm.

The **Policy Executor** (PE) enforces device restrictions by comparing the device's context with the configured policies. Once an application requests access to a resource or service, the PE checks the user-configured restrictions set at the PM to either grant or deny access to the application request. The PE acts as a policy enforcement by sending the authorization information to the AC to handle application requests, and is also responsible to resolve policy conflicts and apply the most strict restrictions.

Through the PM, users can create CBAC policies through configuring application restrictions and linking them to contexts. When an application requests a resource or service, the AC verifies at run-time whether the application request is authorized and forwards the request to the PE. If the request is authorized, the PE then checks if there is any policy that corresponds to the application request. If such a policy exists,

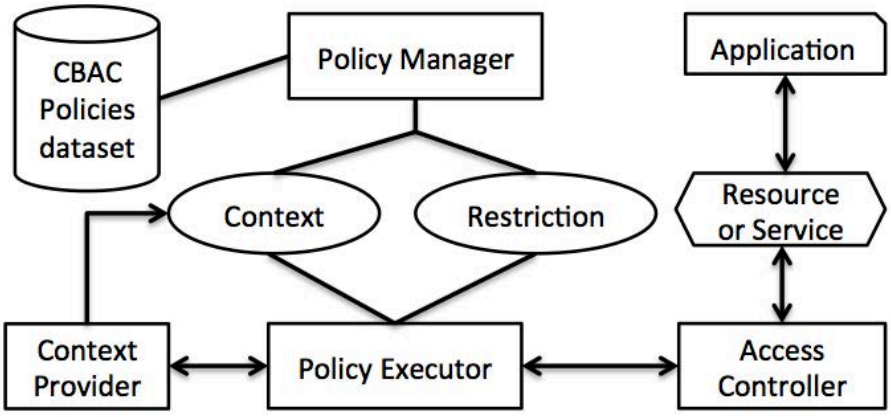


Fig. 3.1. Access control framework.

the PE requests from the CP to retrieve the context at the time of the application request. The PE then compares the retrieved context with the context defined in the policy. In case of a match, the PE enforces the corresponding policy restrictions by reporting back to the AC to apply those restrictions on the application request.

We carefully design the access control framework so that the user-configured policies are securely enforced with minimal processing steps and execution time to avoid any significant delays in responding back to the requesting application. As our design should securely handle policy execution, we maintain the context data provided by the CP to make sure it is accurate, precise and up-to-date.

3.2 Policy Core Model

In this section we describe the core policy constructs that compose our CBAC policies. We start by defining the policy constructs and then categorize our policies according to the type of restrictions and modifications that we need to apply to the Android OS.

3.2.1 Policy Constructs

We define our CBAC policies as a set of restrictions applied to the smartphone applications when the device is located within a specified context. Policy restrictions represent the constraints applied on the applications' privileges in accessing device resources, system methods, functions, user data, and services. Policy contexts represent to where and when the policy must be enforced.

In what follows, we assume three sets: (1) *SUB* the set of subjects representing the device applications, (2) *OBJ* the set of protected objects (objects, for short) representing the permissions, services, and functionalities available for the system or applications, and (3) *ACTION* the set of restriction actions that can be applied through the CBAC policies.

The set of subjects *SUB* is composed of the *PackageNames* of all applications installed on the device. In addition, a special character $*$ is added to the set to represent all installed applications. This character is useful for policies that need to be enforced on all applications, rather than creating the same policy for every application. Moreover, we assume that each object from set *OBJ* has an associated type from the set {Permission, Data, Intent, System_Peripheral}. Let o be an object from the set *OBJ*; notation *type(o)* denotes the type of o. The set of actions *ACTION* defined for our CBAC model includes the following actions:

revoke_Permission: denys permission(s) from being granted to application(s).

shadow_Data: conceals the actual user data stored on the device.

disable_Intent: intercepts and drops the specified intent message.

save_State: disables toggling the state (ON/OFF) of the specified system peripheral.

Definition 1 *(Restriction.) Let $s \in SUB$, $o \in OBJ$, and $a \in ACTION$. A policy restriction is defined as the tuple $[s, o, a]$ such that:*

$$a = \begin{cases} revoke\_Permission & \text{if } type(o) = Permission. \\ shadow\_Data & \text{if } type(o) = Data. \\ disable\_Intent & \text{if } type(o) = Intent. \\ save\_State & \text{if } type(o) = System\_Peripheral. \end{cases}$$

Access control policies are linked to a context that specifies where and when these policies should be enforced. In our model, the policy context is composed of a device location and a time interval. The device location corresponds to where a policy should be enforced. For defining a device location, we use the reference geometric model the describes how locations on Earth are represented. We adopt the spatial model compliant with Open GeoSpatial Consortium (OGC) [62] that uses the notion of GIS features. Features have a defined geometry (points, lines, or polygons) in a reference space, with points to represent a feature with a single location in the coordinate space, lines to represent a feature that has a linear interpolation of an ordered sequence of points, or polygons to represent a feature that has an ordered sequence of closed lines defining the exterior and interior boundaries of an area. Features also have types (road, town, region) and can be given an instance for each type (Champs-Elysees, Paris, lle-de-France).

Specific to our CBAC policies, we define the device location as the physical location that represents a geographically bounded feature (such as a residential and/or commercial building), with boundaries specifying the interior area in which the device is located. The physical location data and boundaries are obtained from the mobile device sensors, mainly captured from GPS, cellular network, and Wi-Fi device sensors as detailed in Section 3.4. In addition to the device physical location, users can assign logical location names to the feature or sub-area (such as living room or work office) in which the device is located. Using these logical location names, users can reuse them in multiple policies without the need to re-capture the device physical location for every policy.

On the other hand, a policy time interval represents the specific time period within which a policy should be enforced. We represent the specific date and time in the format of YYYY-MM-DD-hh:mm:ss. Additionally, we introduce the R flag to define recurring events. The value of R is drawn from the set *{O,D,W,M,Y}* defining the event frequency: $O \rightarrow$ *once*, $D \rightarrow$ *daily*, $W \rightarrow$ *weekly*, $M \rightarrow$ *monthly*, and $Y \rightarrow$ *yearly*. An event is recurred based on the value of R and the date/time set in the policy time interval. For example, to set an event that occurs every Monday from 5 pm to 10 pm, R is set to W and the time interval should be set to a sample event date-time, such as starting on $2013-04-01-17:00:00$ and ending on $2013-04-01-22:00:00$.

Definition 2 *(Context.) Let LOC be a logical location name representing a particular feature or sub-area. Let* $\{ST, ET, R\}$ *respectively be the starting time, ending time, and frequency to when a particular policy should be enforced. A policy context is defined as the tuple [LOC, {ST, ET, R}].*

Definition 3 *(Policy.) Let r be a restriction as defined in* Def. 1 *and c be a context as defined in* Def. 2*. A policy is defined as a tuple* $[r, c]$.

Below is an example of a policy that disables the Skype application from having the *Camera* permission monthly between 4.00 pm and 5.00 pm on the first of every month starting August 1, 2013 at our department meeting room *Room*110:
POLICY = [[com.skype.raider,
android.permission.CAMERA, revoke_Permission],
[*Room*110,
{2013-08-01-16:00:00, 2013-08-01-17:00:00, M}]]

3.2.2 Policy Categories and Examples

We here classify location dependent run-time policies according to the type of restrictions and modifications that we need to apply on the Android OS. Table 3.1 displays examples of policy restrictions for each policy category.

Table 3.1.
Policy categories and examples.

Policy Category	**Example**	**Policy Restriction [s,o,a]**
Resource Restriction Policies	Disable Camera for Skype	[com.skype.raider, android.permission.CAMERA, revoke_Permission]
System Peripheral State Policies	Disbale Bluetooth toggling	[*, BLUETOOTH, Save_State]
Multitasking and Intercommunication Policies	Disable loading a browser activity	[*, Intent.ACTION_VIEW, Uri.parse, disable_Intent]
User Security Policies	Disable uninstalling applications	[*, android.intent.action.DELETE, disable_Intent]

3.3 Implementation

In this section, we introduce the technical details of our implementation which includes our modifications to the Android OS and the components of the *Policy Manager* custom application that acts as an intermediary between the OS and the user's desired policy configurations.

3.3.1 Policy Manager Components

The *Policy Manager* custom application consists of the four main Android application components: Activities, Broadcast Receivers, a Content Provider, and a Service.

Activities: The user interacts with the *Policy Manager* via activities, and through these activities, a user is able to define physical locations and subsequently configure a set of policies for these locations. The main constituents of these activities include *Application Events, Permission Access, Resource Access, System Preferences*, and *Time Restriction*.

BroadcastReceiver: We extended the Android's *BroadcastReceiver* class and created two custom classes, the *StartLocationServiceReceiver* and the *BootReceiver* classes. The *StartLocationServiceReceiver* is responsible for triggering our customized *LocationService* for retrieving device location information. The *BootReceiver*'s main task is to schedule when the *StartLocationServiceReceiver* should request the location service. Once the *BootReceiver* receives the BOOT_COMPLETED *Intent* from the system, it uses the Android's *AlarmManager* service to let the receiver schedule a

pending *Intent* to be sent periodically to our *StartLocationServiceReceiver* in order to update the device location.

Service: The *LocationService* service is derived from the *IntentService* class that facilitates offloading work from the main application's thread, allowing tasks to be performed in the background on a separate thread if desired. *LocationService* determines if the device has moved to or still is in a previously registered area. Offloading the aggregation of location-based data in a separate thread reduces the performance impact of the execution of the *LocationService* on the *Policy Manager*. We use the *AlarmManager* to periodically activate the *LocationService* to ensure the device's location is always up-to-date. By default, the *LocationService* is activated once per minute, but we give the user the choice to configure how often the service is executed. The duration of the service depends on the number of snapshots of location parameters to be taken, which is currently configured to four per area.

Content Provider: The policies configured by the user are stored within the *Policy Manager* data directory. This data is private to our custom application and cannot be accessed by other applications or the system itself, as a result of Linux's kernel user ID access control mechanisms. *PolicyCP* is our custom content provider that acts as a secure intermediary between the policy database and all objects outside of the *Policy Manager*'s running process. We chose to use the SQLite database to store user-configured policies due to the support and ease of programming provided by the Android API's associated with storing and managing databases on Android devices.

We provide some built-in policies that are pre-configured and can easily be modified to achieve the required user needs. Moreover, we can also refer to existing usability techniques [63–66] that can be used to offer regular users with preset and adapted policy configurations.

3.3.2 Permission Management

In the Android system, all resources that require explicit access rights in the form of permissions are protected by the *ActivityManagerService* class via permission verification. When an application attempts to use any of these resources, the *ActivityManagerService*'s method called *checkComponentPermission* is invoked to verify if the calling application has the appropriate permission(s) to access the resource.

We apply our modifications to this particular method by simply intercepting the permission call before the system performs its standard permission verification process. Given the permission and the application name, the system subsequently calls our custom content provider's *revokeResourceAccess* to determine the next course of action. Depending on the user's policy configuration, the next course of action could either be returning the constant *PackageManager.DENIED* in the *checkComponentPermission* if the user has configured to block that permission from the requesting application, or letting the normal verification process take its course. We also give the ability to revoke any or all permissions system-wide via the *PolicyManager*'s interface.

3.3.3 Restrictions on User Data

Our implementation of data obfuscation complements many of the techniques previously used in [46] and [6], but instead under the domain of CBAC policy restrictions. We obfuscate user data from applications attempting to access it if the policy restriction applies to those applications. We modify the Android APIs that access the user data saved on the device.

Relational database systems are the common data management systems used to create, store, and manage user data. Accessing these data usually require calling the *ContentResolver's query()* method, and thus we modify it for our purposes. Instead of returning the expected *Cursor* object needed to point to the required data, a *NullCursor* object is substituted. A *NullCursor* object represents an empty dataset,

such as an empty list of pictures as if pictures were not present or never stored on the device.

3.3.4 Managing System Peripheral State

We also give users the option to configure a policy to restrict access to peripherals (e.g., Bluetooth) when entering a particular location. Specifically, users can set up their devices to prevent applications from modifying a peripheral's current state (enabled/disabled). While it is possible to modify a peripheral's current state by using permission management, we modify the specific methods that enable/disables these peripherals in order to prevent applications from crashing that do not have code for handling exceptions resulting from revocation of permissions. As an example, for Bluetooth we modified the *BluetoothAdapter* class and for Wi-Fi we modified the *WifiManager* class so to assure that these modifications do not result in application crashes and to prevent applications from modifying peripherals current state. Whenever an application tries to modify the state of a system peripheral, our content provider *PolicyCP* checks the validity of the request and would refuse the request if the request tries to override a user-configured restriction.

3.3.5 Intent Management

Intent messages are one of the common forms for inter- and intra-communication between application components, sent via three methods: *startActivity(), sendBroadcast()*, and *startService()*. Preventing an application from sending intents is simply a matter of intercepting the intents when the aforementioned methods are called by applications. *Intent* interception provides the user the ability to prevent an application component from starting another activity, broadcasting any possible sensitive information, or executing a possibly suspicious background service. For example, without the need of declaring the Android permission "RECORD_AUDIO", an application can indirectly access the device's microphone recorder application by re-

Table 3.2.
Examples of policy restrictions that can be controlled via intercepting Intents.

Restriction Category	**Description**
Application Install/Uninstall	Prevent an application from sending an intent to install or uninstall an application.
Application Multitasking	Prevent running multiple user-application simultaneously.
Services	Prevent applications from starting background services.
Broadcasts	Prevent applications from broadcasting Intents.
Application Launching	Prevent certain applications from running on the device.
Lock/Unlock Device	Preventing requesting pin code to unlock the device.

questing the *Activity* class to send a record audio intent. Therefore, we modified the *Activity* class which hosts *startActivity()* and *startActivityForResult()*, and the *ContextWrapper* class which contains *sendBroadcast()* and *startService()*. We modified these methods to intercept the *Intents* and control the actions performed based on those *Intent* objects.

We classify these *Intents* based on the contents and description of the intent objects. The user is given, via the *Policy Manager* interface, the ability to prevent a specific set of *Intents* from being sent. Table 3.2 lists few examples of these *Intent*-related restrictions.

Launching applications is achieved by intercepting those intents and preventing applications from being started either by users or by other applications. The first method is when users launch applications through the default Android *Launcher* application, which is the home screen of the device. The second method for starting another application is calling its activity within an already opened application. We extract the action and category from the *Intent* object, and verify if it is "android.intent.action.MAIN" and "android.intent.category.LAUNCHER", respectively. If those specific contents are present and if the simultaneous running of applications is restricted, we discard the intent preventing the framework to handle it.

Finally, through the *Intent* management, users can control when to request a pin code when unlocking the device. In our implementation, we modify the *Keyguard-ViewMediator* class in order to intercept the locking operation of the device, and thus controlling when a PIN is required.

To summarize, users will have all the options to specify applications restrictions associated with context data through the policy managers. In the policy manager, the permission management is used to configure application restrictions related to device resources (e.g. Camera). Restrictions on user data are used to shadow user data (usually saved in relational databases) and to return fake data to the application (e.g. Contacts). Managing system peripheral state is used to control applications actions in toggling the state of certain resources (e.g. enabling/disabling Bluetooth). Finally, intent management is used to control the communication between applications and filter user and application actions on the system. Intent management is also used to configure restrictions on applications accessing resources, as in some cases, these applications are developed to access system resources indirectly through using an intent message rather than requesting a permission.

3.4 Context Management

The main source of location-related information for our access control system is the Wi-Fi Access Points (AP) and their corresponding signal strengths. Location information acquired from GPS and cellular towers is also aggregated to our context definition but may not be sufficient for indoor localization especially that they may become weak or unavailable inside buildings or areas within building structures [67, 68]. However, location information retrieved from Wi-Fi parameters could be more precise to differentiate between closely located sub-areas within the same GPS location [69, 70].

A spatial region is represented by combining GPS coordinates, cellular triangulation location data, and Wi-Fi APs and signal strengths. In Android, the GPS

coordinates and cellular triangulation are obtained in a similar fashion by invoking the Android *LocationManager* service. Once the *LocationManager* is invoked, we request location updates by calling the *requestLocationUpdates* method that returns a *Location* object which contains latitude, longitude, timestamp, and other information.

Wi-Fi is handled differently than the previous two location methods. We obtain the Wi-Fi parameters by invoking the *WifiManager* service to retrieve the Wi-Fi access points scans. We register our *BroadcastReceiver mWifi_receiver* with an IntentFiler action to receive the broadcasted Wi-Fi scanned intent, and then request for and subsequently process the actual scanned access points data.

In our CBAC policy system, we provide users with a utility to either capture the physical location of the device or to manually enter the device location coordinates. In the following sections, we show our design and implementation of the location capturing phase and detection phase and how the device's context is matched with a pre-defined policy context.

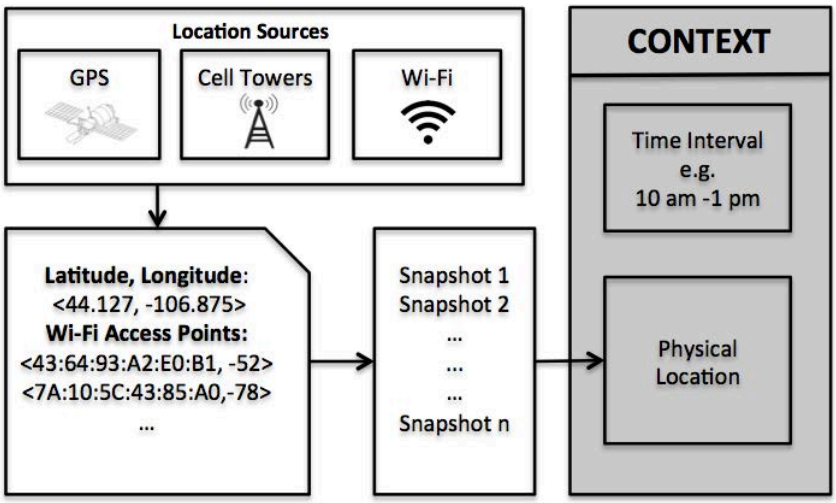


Fig. 3.2. Location capturing phase.

3.4.1 Location Capturing Phase

Figure 3.2 describes how location data is captured for each context defined by the user. Through the location scan interface, the user is able to capture several snapshots of location data in different sub-areas. For each sub-area, location data is accumulated from each snapshot; the GPS coordinates and the cellular triangulation, when applicable, import the latitude and longitude from the captured snapshots and only select those with the highest position accuracy. With respect to Wi-Fi, we noticed that the Wi-Fi access points signal strengths fluctuate even if the device is stationary or motionless. Therefore, our application scans the signal strengths of each access point for several seconds gathering the RSSI values at each particular sub-area. We conduct the scans with no other users in the vicinity as the presence of other users would affect the RSSI values [9,71]. Finally, the accumulated data, which mainly consists of Wi-Fi access points with signal strength ranges in addition to GPS and cellular triangulation data as supporting location information, will represent the device's physical location.

Any location that is not defined by the user or does not have location information saved on the device will be considered "Unregistered". Therefore, we designate a default policy restrictions for the user to configure whenever the device is located in an unregistered location. In addition, we allow users to register locations that have not been previously visited. This is achieved through either manually entering the publicly known longitude and latitude of the desired location, or by acquiring the fine-grained Wi-Fi parameters from other devices who have saved those parameters. This becomes very practical when the user is switching between two devices and needs to import previously saved policy contexts to the new device.

Our implementation does not store all the GPS or triangulated cellular coordinates acquired, rather a subset of those coordinates that bound into a convex hull and their associated precision. The points in the interior of the convex hull are discarded. We also only store the RSSI range for each distinct Wi-Fi access point scanned. This

range is the minimum and maximum RSSI values aggregated from all the sub-areas for each access point. A sub-area is therefore represented as a range of Wi-Fi signal strength values at the least, and if with high position accuracy, also a representation of a convex hull of GPS or triangulated cellular coordinates.

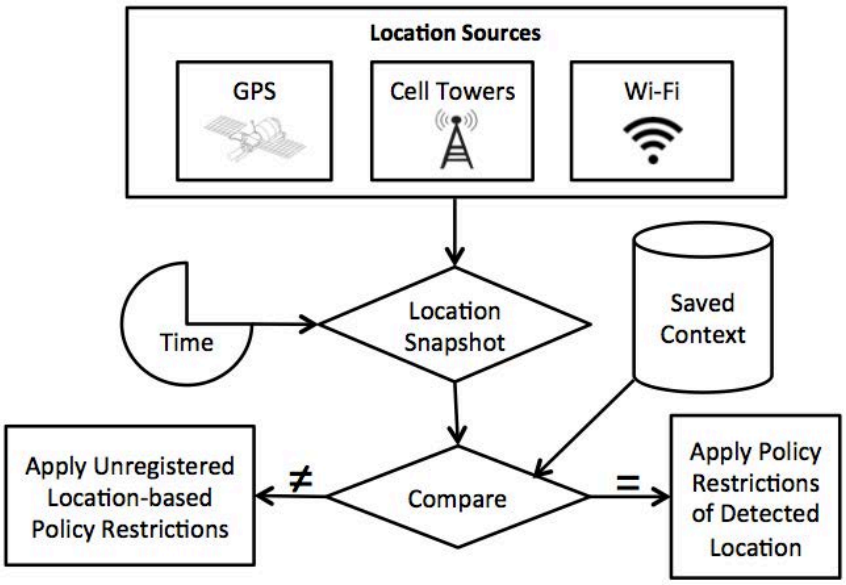


Fig. 3.3. Location detection phase.

3.4.2 Location Detection Phase

Figure 3.3 describes how device context is detected and matched with pre-defined context. Periodically, the location background service is re-instantiated to accumulate location-context data to determine the device's current whereabouts. Like when registering and scanning a sub-area in the location capturing phase, we scan the device's location-related data. The list of user-registered areas that have a subset of the scanned neighboring access points are extracted from the database first. Matching distinct access points is computationally less expensive than determining if coordinate position falls within the boundaries of a convex hull. Then, using the current signal strengths of the access points, we reduce the list to only a set of "best-match" list of

physical locations whose access points fall within the current captured signal strength values. If the current scanned GPS or cell network coordinates fall within the convex hull of the associated sub-area, then it is highly likely that the sub-area has been located.

In the unlikely situation when the "best-match" list of physical locations contain more than one location, the user is given the list to confirm his/her location. Even though this event is unlikely to happen, it may still occur because Wi-Fi access point's signal strengths are volatile; their signal strengths fluctuate at a given location. As a consequence, an access point's signal may be, at some point, too weak for the Android's device sensor regardless of whether the device is in motion or stationary. In the location capturing phase, we aggregate the retrieved data that records a range of RSSI values from different sub-areas within the same location, for example, instead of just a single snapshot. In the detection phase, however, we take a snapshot of the location data, which may have only a subset of the previously aggregated data. We compare the previously stored values with the snapshot data. Specifically, with respect to Wi-Fi, we determine if the captured RSSI value of a particular access point is within the stored range. We perform this operation for each access point captured in the snapshot and count the number of tests passed, which is the basis in determining the physical location of the device.

3.5 Experimental Results

In this section, we report experimental results about the CBAC mechanism and evaluate its impact on the device system and applications. Our modifications to the Android source code were tested on the Android Nexus 4 cellular device and Android Nexus 7 tablet running the Android 4.2.2 OS (API level v. 17). We ran the top 250 applications from the Google Play market for testing and evaluating our modifications. Each experiment has been carried out with the help of the Android Debug Bridge (ADB) utility by using the command "adb logcat". We inserted logging

commands in various parts of the operating systems where modifications were made to observe, for example, application access events.

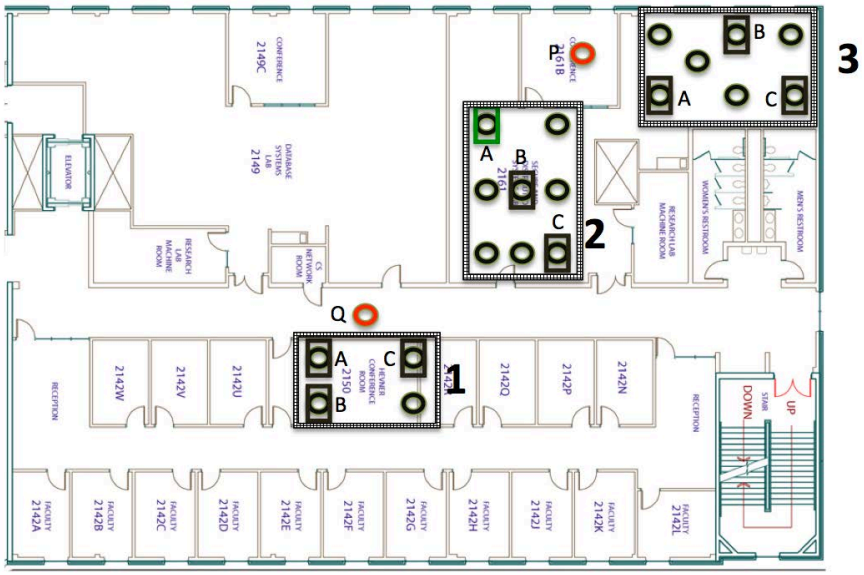


Fig. 3.4. Tested areas in one of our campus buildings.

Experiment 1: Location Detection Accuracy. The goal of this experiment is to evaluate the accuracy of the location detection algorithm used in our CBAC mechanism. We measure the number of success and failure detections per sub-area. Figure 3.4 displays the schematics of one building where we performed some of our experiments. The large, grid-pattern rectangles point out main locations or areas, identified by numbers. Areas outside the rectangles are considered "unregistered." The black circles indicate the specific sub-areas examined during the location capturing phase. All other colors indicate other sub-areas examined during the detection phase.

Figure 3.4 shows three tested rooms located on the same floor. However, our experimentation included several buildings and areas. In each room, we chose at least four spots to participate in the location capturing phase to accumulate location-related data, in order to construct a robust set of location parameters per room to be stored in the database. In each location, we analyzed three sub-areas, indicated

by 'A', 'B', or 'C' and measure the detection rate in each of these subareas. For that particular floor which contains over 15 Wi-Fi access points, we captured the top 5 Wi-Fi access points per snapshot with the highest signal strength for each tested sub-area.

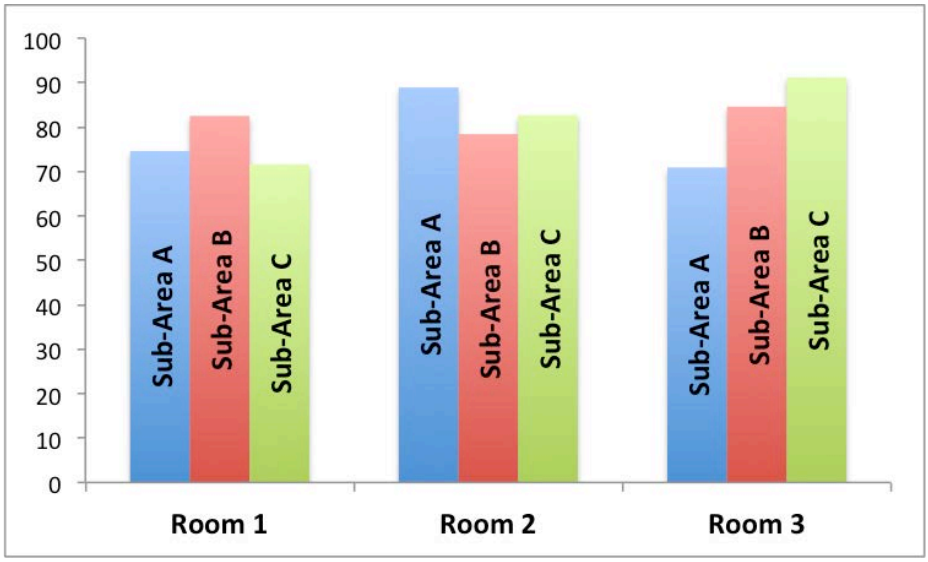


Fig. 3.5. Detection accuracy rate of closely located areas.

Figure 3.5 displays the detection accuracy rate in the 3 sub-areas of rooms 1, 2 and 3. At each of the sub-areas of each room, we performed 50 location tests and counted the number of successful detections. Our experimental results show that the successful detection rates were up to 91%, and in the worst case scenario we had up to 29% of incorrect detections. This experimental result was complemented by testing several "unregistered" areas around the registered rooms. We detected 16% of false positives, that is, unregistered areas that appeared to be user-defined. Within the registered areas, the values of the signal strengths of matching Wi-Fi access points fell within the range of signal strengths first acquired during the location capturing phase. However, in the unregistered areas, especially the further away the device was when the snapshots at the location capturing phase were taken, the values fell outside the stored range because of building structures hindering the Wi-Fi signal strengths.

Experiment 2: Impact of Permission Restrictions. The purpose of this experiment is to observe the impact of permission-related policy restrictions on applications. Specifically, we are interested in whether or not an application crashes as a result of being denied a permission that was initially granted at installation time. Therefore, we performed a stress test on each application and observed the impact on the application upon revoking its permissions when requesting a service or resource. We performed our experiment on 245 Android applications and used the ADB logging utility to view the permission being revoked when the *checkComponentPermission()* method is called.

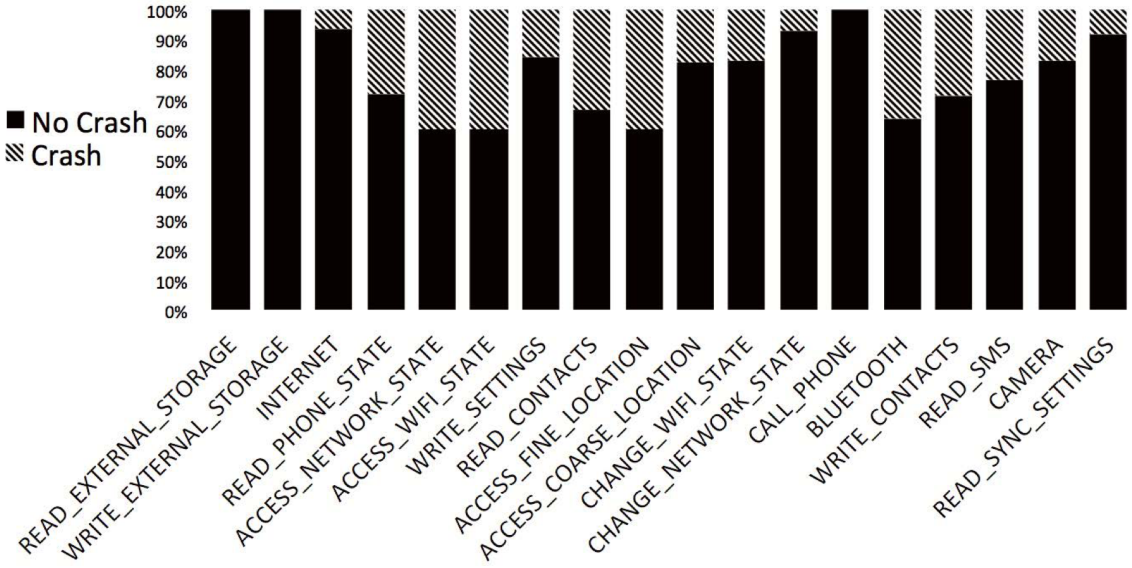


Fig. 3.6. Impact of permission revoking on applications.

Figure 3.6 shows the percentage of application crashes upon performing the stress test on each permission. We counted an application as crashing even if it crashed during the execution of one minor functionality. The main cause of these crashes is due to the developers' mishandling the denial of previously granted permissions. Since application crashes are due to developers' mishandling the denial of previously granted permissions to their applications, application crashes can be prevented if

Table 3.3.
Time overhead for modified Android methods.

Android Method	**Overhead (ms)**
checkComponentPermission(..)	12.220
Intent-startActivity(..)	12.708
Intent-startService(..)	5.402
Intent-sendBroadcast(..)	5.208
User Data-ContentResolver(..)	12.300
Device Peripherals-setEnable(..)	8.351

error-handling is added whenever an application attempts to access a resources or request a service. In fact, throughout testing several application versions, we realized that the number of application crashes has been decreasing over time. This is because developers are now aware that not having the permission error-mishandling script is causing several application crashes. A script is thus being added in their application updates especially with the evolvement of many permission restriction techniques.

Experiment 3: Performance Overhead. The purpose of this experiment is to evaluate the timing overhead introduced by our modifications to the Android OS. We calculate the amount of time it takes for our modified methods to fully execute once called by applications. We also compare the execution times of these methods before our modifications to estimate the overhead introduced by our modifications. Specifically, we measure the overhead time caused by intercepting application permissions, user data accesses (e.g. *Contacts*), *Intent* messages, and access to system peripherals (e.g. Bluetooth).

Table 3.3 reports in milliseconds the time imposed on these methods. As the results show, the overall delay introduced by enforcing our CBAC policies is not perceivable by the end-user.

Experiment 4: System Memory Overhead. The purpose of this experiment is to measure the amount of memory overhead placed on the system after our modifications. Mainly, we aim to observe the changes in memory usage caused by our

application restrictions and by the *LocationService* method that continuously run in the background for context updates.

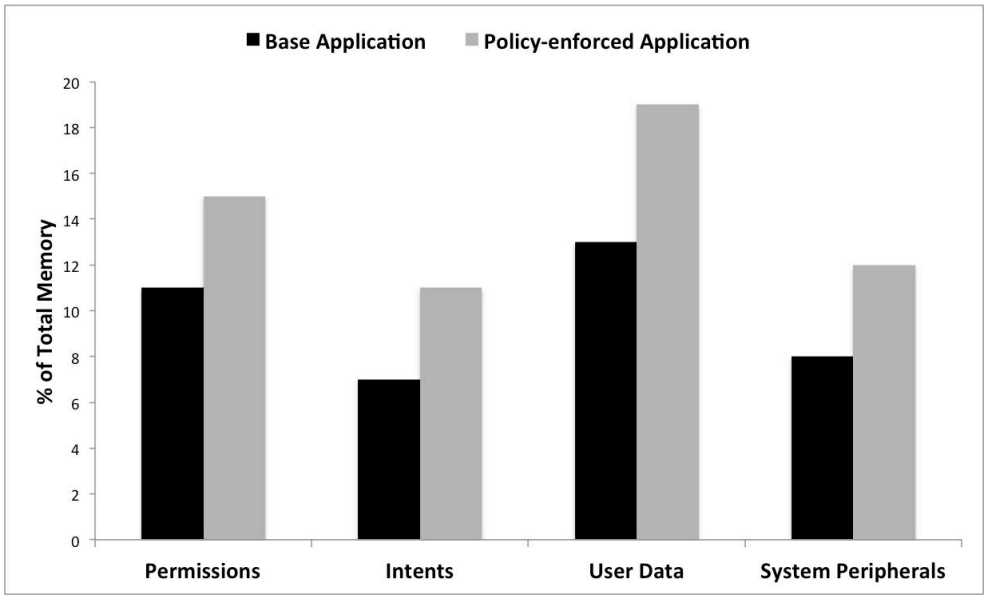


Fig. 3.7. Memory overhead with and without our CBAC policy restrictions.

Figure 3.7 shows that the memory usage when enforcing our CBAC policies closely matches the memory usage when these policies are not enforced. Even though our experiments test the different restriction categories separately as shown in the figure, we believe that the observed memory overheard is due to the *LocationService* that is instantiated periodically to keep the device's context up-to-date.

Experiment 5: Battery Consumption. The purpose of this experiment is to observe the Android device's battery consumption change when CBAC policies are enforced compared to when they are not. For this purpose, we monitored the device's battery percentage when running both the unmodified OS and our customized system, separately. In both cases, we forced the device's screen to never turn off with Wi-Fi and GPS enabled, a representation of a somehow worst case scenario as when the user is continuously using the device for that duration. Since the period for which the *LocationService* method t hat is responsible for checking the device's location

can be customized by the user, we tested the battery consumption for different time periods for the purpose of getting a fair evaluation.

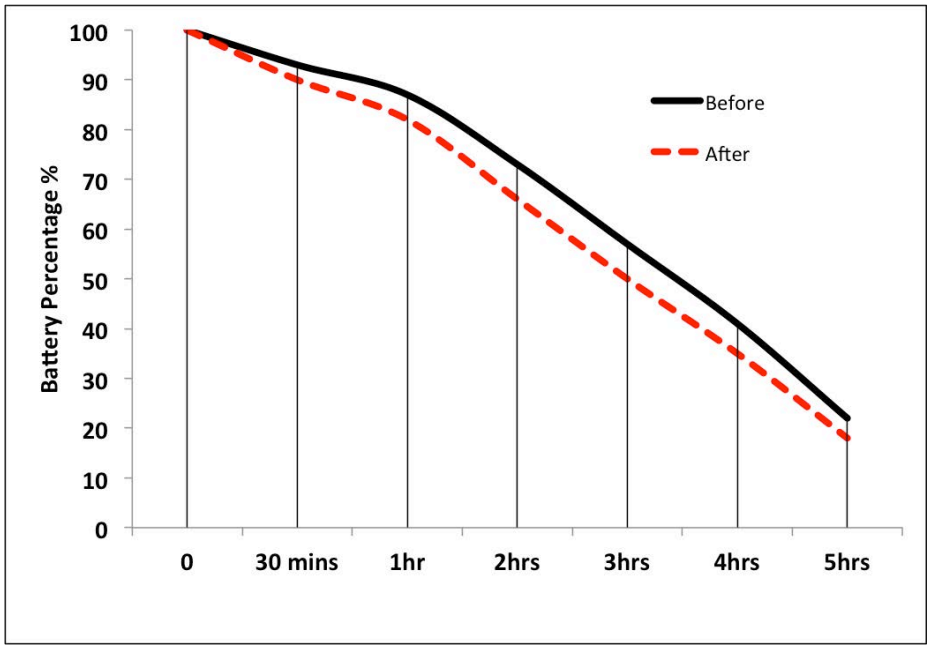


Fig. 3.8. Device battery consumption when checking for context updates every 30 seconds.

We started our experiment by setting the device to check for any device location updates every 30 seconds. Figure 3.8 shows that the battery percentage displayed on the device when enforcing our CBAC policies drops $\sim$ 5% less per hour compared to when the policies are not enforced. We achieved similar results when context was updated every 45 seconds. However, when we set the timer to check for device context every 60 seconds or above, the battery usage percentage of when the CBAC policies are enforced closely matches the battery consumption when the policies are not enforced.

3.6 Security Analysis

In this section, we present a security analysis of our implementation of the CBAC system to analyze possible threats from a malicious user or applications that can bypass our policy restrictions. The aim of our security analysis is to identify possible threats by malicious users or applications that can bypass CBAC policies and to mitigate these threats.

Colluding Applications. In Android, each device application is assigned a unique UserID (UID) that the system uses to refer to an application. However, if two applications are created and signed by the same developer, the system will give both applications the same UID, which gives these applications the ability to share the same processes if needed [72]. The stock Android OS applies its security policies not based on the application label or its package name, but rather on the process UID. In our modifications of the OS, we obtain the name of the package (application) which is performing an action by calling the PackageManager's *getPackagesForUid(int uid)*. This way, our restrictions are not based on UID but are transformed in order to refer to the package name. As an example of such threat, consider two applications, Application A and Application B, that share the same UID. Suppose that the user blocks access to GPS capabilities from application A. However, although successfully blocked, application A may still be able to acquire information about the user's or device's location because Application B was not denied access to GPS. In our system, we prevent such threat by blocking *all* package names associated with a UID using the *getPackageForUid()* method.

Circumventing Application Multitasking Restriction. A malicious developer may attempt to bypass the restriction that disallows multiple applications from running simultaneously by creating a custom launcher-like app. Android is modular, and thus the default home screen can be replaced. Our system is not vulnerable to such an attack because we check all possible intents as our system is not limited to intents related to the stock *Launcher*. Thus any intent with "an-

droid.intent.action.MAIN" and "android.intent.category.LAUNCHER" will be intercepted and processed to disable multitasking of any launcher the user decides to use.

Protection of Policies. As users can configure policy restrictions based on time and location, these restrictions are either applied system-wide or per application. If these policy restrictions can be altered by applications, then any malicious application can perform specific attacks based on policy configurations. To protect policies, we thus do not allow write privileges to be granted on policies so to prevent policies from being modified.

Malicious applications that are aware of our CBAC policies may try to drop a policy or modify the device's detected context so that the wrong policy is applied. However, in our implementation we retrieve context information directly from the system protected APIs that cannot be altered by applications. Moreover, context information is managed by our *Content Provider* that gathers such information regardless of which applications are running on the device or services requested by applications. This independency from the *Content Provider* gives robustness in gathering context data that is forwarded to the *Policy Manager* as discussed in Section 3.1.

Sensitive Information Disclosure. Some applications may maliciously leak private user information once they detect that a previously granted privileged application is revoked. As an example, a malicious application that was granted access to *Camera* and *GPS* at installation time may upload the device GPS location to the application server once it detects that the *CAMERA* permission is revoked, leaking the user-private location. For this reason, our implementation provides the user, at the time of policy configurations, with a list of privileges that are still granted to the configured applications, acting as a warning for the user to be aware of the sensitive information still accessible by the configured applications in a particular context.

Continuous Running Application Processes. When an application requests access to a resource, the Android OS checks if the application has the appropriate permission(s) just at the time of the request. If the user-configured policy grants such permission to the requesting application in a given context, some processes associated

with certain resources may continuously run even if the device is later located in a different context for which the user has denied access to such resource. The reason is that permission granting is not checked continuously while the process is running, rather is only checked when the request is issued. Malicious applications may take advantage of this, for example by continuously recording audio in one context while transitioning to another context.

Audio recording using the *Microphone* resource is one example of a continuously running process that will not terminate until the recording is stopped. Take for example a user who attends private meetings in a same meeting room that he configured a policy to disable the *Microphone* resource. A malicious application can begin recording outside of the meeting room area without alerting the user, and continue recording when the user enters the restricted meeting room. The Android OS does not continuously verify whether an application has audio recording permission during recording. It verifies each time a request is made, and thus when approved the application can continue using the peripheral for that specific session. Our implementation prevents this type of attack. Once a registered area is associated with a restriction on video or record audio access, the location service forces the applications with the associated permissions in their *AndroidManifest.xml* to close.

3.7 Conclusion

We proposed a modified version of the Android OS, which we refer to as CBAC, supporting context-based access control policies. These policies restrict applications from accessing specific data and/or resources based on the user context. The restrictions specified in a policy are automatically applied as soon as the user device matches the pre-defined context associated with the policy. Our experimental results show the effectiveness of these policies on the Android system and applications, and the accuracy in locating the device within a user-defined context using mainly RSSI values from WiFi access points.

CBAC, however, has limitations. CBAC is a user-centric approach to device management in order to protect sensitive content on the device. This user-centric approach to configuring policies for end-users' devices may not be suitable for enterprises that require strong security guarantees. A critical security issue in enterprise environments is the inability of enterprises to (fully) trust employees to perform necessary tasks to secure enterprise content. Assuming an employee is benign with no malicious intent, the employee may incorrectly configure his/her device policies. Enterprises must also consider insider threats within the organization. That is, employees with malicious intent may purposefully leak sensitive enterprise content stored on their devices. Enterprise content may not only exist on mobile devices, which is an assumption implicitly made in this chapter. Content may also exist in remote servers which can be accessed via mobile devices. In addition, the proposed mechanism does not consider that access to content may be contingent upon the presence of other users in the area in which the user requesting access is located. The next chapter, Chapter 4, addresses such concerns and limitations by proposing a context-aware access control system that incorporates proximity-based constraints when access to enterprise content is remotely requested.

4. SECURING REMOTE ENTERPRISE CONTENT VIA PROXIMITY-BASED ACCESS CONTROL

The previous chapter addresses the issue of localizing the user and adapting applications' access to client-side content depending on the user's location and time of access. However, the proposed system had several limitations which include localizing the user with untrusted client-side technology, lack of consideration of other users within proximity when sensitive content is accessed, and delegating the user to appropriately configure access control policies.

To support PrBAC scenarios such as Separation of Duty (SoD) and Absence of Other Users (AOU), we proposed the first iteration of our Context-Aware System to Secure Enterprise Content (CASSEC). CASSEC took a wireless, infrastructure-based approach to achieve the localization of occupants within a monitored space which enables geo-spatial RBAC [15, 33]. A wireless, infrastructure-based approach makes the system more resilient to malicious attacks; we assumed, for example, the least amount of trust in users since users may attempt to circumvent the access control process by not manually reporting their location or providing false location data. In addition, the architectural model allowed a fluid context-sensitive authorization process, thereby enabling zero interaction authorization (i.e., it did not require user intervention). We first showed how to enforce SoD by using Bluetooth MAC addresses of Client devices of nearby occupants as proof-of-location, which enabled the system to determine *who* was in a given space. We then showed how to enforce AOU by exploiting the degradation of WiFi received signal strength as a result of human-induced interference when people are near access points, which enabled the system to determine *how many* people were in a given space. With such information obtained passively by a Proximity Module (PM), the Authorization Server (AS) component was able to enforce PrBAC policies whenever an authenticated Client requested from

the Enterprise Content Server (ECS) component access to resources depending on the presence, or lack thereof, of users. Figure 4.1 displays CASSEC's architectural components.

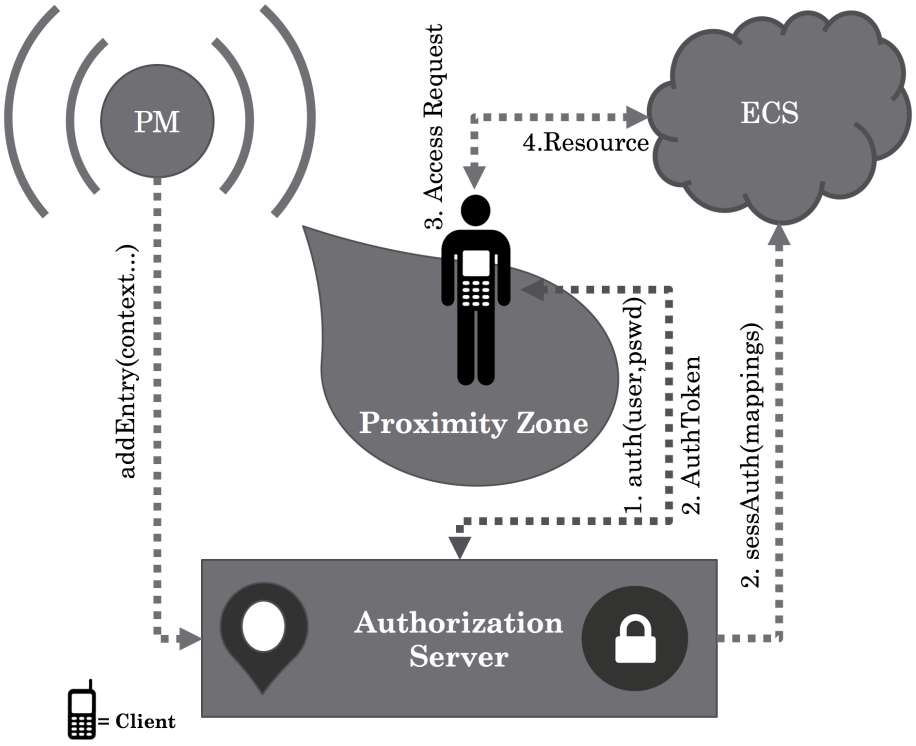


Fig. 4.1. CASSEC's proximity-based access control architecture. Arrows indicate secure wireless network communication.

Our first iteration of CASSEC, however, has several drawbacks. First, it does not take into account the phenomena of radio signals permeating through walls. Multiple proximity modules residing in adjacent proximity zones would simultaneously detect the same Bluetooth-enabled Client, when in fact, the Client only existed in one of said proximity zones. As a result, such a benign occurrence is automatically inferred as malicious activity. Given that Bluetooth's omni-directional transmission range is 10m (~33 ft), the number of false attack detections may increase in standard enterprise settings, such as small offices or conference rooms. A non-negligible false detection rate is a major drawback that hinders the practicality and ease-of-adoption

of the solution. Second, the system was susceptible to observable Bluetooth manipulation (see Section 4.7), such as an unauthorized individual obtaining an authorized user's phone, whether by theft or voluntary provision. If such an attack occurs, the unauthorized individual can gain access to restricted resources s/he would normally otherwise not have access to.

To address such context monitoring issues, we further investigate techniques that leverage existing contextual information from both the physical and computing realms. Contextual information extracted from the environment can help a context-aware system in inferring the situation of entities within that environment. However, being able to infer the correct or more probable conclusion w.r.t. the situation of an entity highly depends on the reliability of the extracted contextual information. Reliability could be measured, for example, by the level of accuracy, precision, or security in using a technique or technology (e.g., occupancy detection or biometric authentication) to extract or process contextual information. With respect to security, a context-aware system may also need to adapt its access control decisions to the *degree of reliability* of such information. Given the dynamic nature of EED scenarios and idiosyncratic phenomenon observed in radio-based occupancy detection technologies, it is essential that context-aware systems emulate a sentient characteristic when making inferred decisions: confidence. Access control policies should incorporate confidence constructs when specifying contextual restrictions.

In this chapter, we thus propose a major extension to CASSEC, which we refer to as CASSEC 2.0, by adding confidence constructs to the location and role constructs in PrBAC policies. In addition, we conduct a feasibility study to show that the approach is viable within an enterprise environment, which can be achieved via preexisting technologies and solutions integrated within the enterprise's mobile IT infrastructure. Through the location construct, a policy can specify that resource access authorization is granted only if the context-aware system can determine to a specified probability that a user is in a room. We employ Bluetooth Low Energy (BLE) capabilities of PMs and Clients to perform continuous co-proximity authentication, and

use BLE beacons transmitted during this authentication phase to provide a certain degree of confidence that the Client is in a particular proximity zone, even when multiple PMs in adjacent rooms detect the same Client. Through the role construct, a policy can specify that access to resources is only granted if the system can determine with high confidence whether the current user of a Client device is the true owner of the device. We leverage accelerometer and fingerprint sensors within smartphones to achieve behavioral and physiological biometric authentication. Behavioral biometric authentication is achieved by passively analyzing the gait patterns of the Client's current user via the smartphone's accelerometer. Although human gait is behavioral and resistant to significant change over time, various factors can slightly influence the extracted gait features at runtime [73]. Consequently, if the Client cannot passively identify the current user through runtime gait measurements with high levels of assurance, the Client will take an active approach and request the user to authenticate him/herself via the fingerprint sensor (i.e., physiological biometric authentication) when the user next requests access to resources.

The CASSEC 2.0 system has thus the following contributions:

1. *Confidence Constructs:* We incorporate confidence specifiers into context-based access control policies. More specifically, we incorporate such specifiers into PrBAC's role and location constructs, thereby enabling the CASSEC 2.0 system to factor in the degree of reliability of contextual information during authentication and authorization processes.

2. *Feasibility Study:* We conduct a feasibility study to show that the approach of CASSEC 2.0 is viable in a practical enterprise setting. We leverage solutions within an enterprise's preexisting IT infrastructure to evaluate confidence constructs, and apply such constructs to biometric and co-proximity authentication.

3. *Co-Proximity Authentication:* We provide a timed challenge-response protocol using BLE beacons as our underlying co-proximity authentication technol-

ogy. The protocol prevents an adversary, who has modified his device's unique user ID, from impersonating another user. However, our study shows that using distance-bounding techniques over BLE beacons is a feasible defense only against a sophisticated attacker able to execute relay attacks under a certain adversarial model.

4. *Biometric Authentication:* We leverage behavioral and physiological biometric authentication to evaluate confidence specifiers. Our study shows that our approach is feasible as we are able to verify that the current user of the Client device is the true owner with high confidence when the phone is placed on the hip and within the pocket, respectively.

The chapter is organized as follows. Section 4.1 introduces proximity-based scenarios and specific examples that motivate this work. We then briefly discuss background information on biometric authentication techniques in Section 4.2. We provide in Section 4.3 a PrBAC policy specification for CASSEC 2.0. Section 4.4 establishes our system's assumption. Section 4.5 introduces the architecture and underlying components of our approach. Section 4.6 discusses implementation details followed by a report of data collected from our use case study. We analyze the security of our approach in Section 4.7. Section 4.8 concludes the chapter.

4.1 Motivating Scenarios

In what follows, we present scenarios motivating the need for context-aware systems in which access to sensitive resources must be controlled based on proximity parameters.

Consider a military organization with monitored government facilities such as restricted military bases or buildings. Military personnel are assigned roles that reflect ranking and privileges. The roles *General* and *Private* are assigned to the highest- and lowest-ranking personnel in the army, respectively. In terms of accessing restricted facilities or resources, the former is granted many privileges, while the latter has very

few. Consider also the role *Civilian*, which indicates an individual operating outside of the military organization, and who is granted no privileges. Suppose that three military personnel, two *Generals* and one *Private*, are granted access to documents classified up to the level of *top secret* and *restricted*, respectively, according to a multi-level security model.

Separation of Duty Scenario. *A document classified as top secret is highly sensitive, and requires that* ***at least*** *two personnel with the role General be present in order for it to be accessed. The document is accessed via desktop terminal and is stored within a designated, but restricted office in which only Generals are allowed to enter.*

This scenario reflects the security principle SoD. That is, two or more people are responsible for cooperatively completing a task. In addition, the circumstances requires that said document must be accessed at a specific location.

Absence of Other Users Scenario. *A document classified as restricted, but with the additional caveat "****for your eyes only****", requires that a specific Private can access it via smartphone mobile, however, only if no other individuals are present at the time of access.*

Such an absence-based restriction not only includes military personnel of various rankings, but also individuals that assume the role of Civilian. Civilians are often temporarily recruited to work on military projects, but are highly monitored and usually given only the set of privileges needed to complete the project and nothing more. We note that, unlike the SoD scenario, in this AOU scenario the document can be accessed via the Private's smartphone device in any location including locations that Civilians may have access to. Therefore, less infrastructure is required as it is not necessary to know the identity of every person in the Private's vicinity.

4.2 Background

Biometric information characterizes measurable human biological features [74]. Most biometric features are unique per person and can be found in every individual. In the context of security, *biometric authentication* refers to techniques that rely on such features to uniquely identify and validate the identity of an individual. Human biometrics can be classified into two types: physiological and behavioral. Physiological biometric authentication is based on static physical attributes such as fingerprints, iris, retina, or facial features, whereas behavioral biometric authentication relies on identifiable characteristics of a user's behavior that typically do not change over time such as keystroke dynamics, signature, or gait.

At a high level, biometric authentication has two phases: enrollment and authentication. Before authentication can occur, an individual must first be enrolled into the system by extracting and storing his/her biometric data within a template. Later in the authentication phase when the identity of the individual must be verified, the biometric data collected at runtime is compared to the previously constructed template. From this comparison, a similarity matching score is produced, and whether an individual is accepted/rejected (i.e., non-/identified) depends on a threshold set for the system. In this system, we employ both physiological and behavioral biometric authentication for user verification using two techniques: fingerprint and gait recognition. Modern mobile devices already have integrated solutions to enroll and authenticate users via fingerprint scanning technology [31]. However, such devices lack gait recognition solutions. We therefore only describe user verification via gait recognition below.

User Verification via Gait Recognition

Lee and Grimson defined gait as "an idiosyncratic feature of a person that is determined by, among other things, an individual's weight, limb length, footwear, and posture combined with characteristic motion. Hence, gait can be used as a biometric

measure to recognize known persons and classify unknown subjects" [73]. Empirical evidence supports this definition as researchers have conducted experiments which analyzed over 700 users' gait patterns and found gait patterns to be unique [75]. As a result, it is possible to verify whether the user of a mobile device is the true owner of that device.

Gait recognition for the purpose of user verification is not novel [74], nor is it the focus of this chapter. The main approaches to measuring and analyzing gait biometric are machine vision, floor sensor, and wearable sensor. Deploying additional hardware incurs additional costs, as is the case in the first two approaches. Fortunately, state-of-the-art cellular devices are embedded with a set of sensors, including accelerometers, which have now become a standard for modern smartphones. Consequently, we only employ a wearable sensor approach. We leverage a recent work proposed by Ren *et al.* [76] for several reasons: (1) it utilizes readily available accelerometers embedded within smartphones to detect possible user spoofing in mobile healthcare systems; (2) it takes into account the fact that computational resources are limited on mobile devices; and (3) it is robust to variations in users' walking speed. See Section 4.5 for more details.

4.3 Policy Specification

Several research efforts have focused on the design of access control policy languages [32, 33, 41, 42, 77, 78]. In this section we introduce a simple, yet expressive policy specification (Table 4.1) that leverages existing policy languages. We adopt the syntactical structure of XACML, which is an XML-based language for access control, and apply it in defining proximity-based RBAC policies for CASSEC 2.0. The terms in quotes ' ' represent static tokens. The terms in italics indicate functions.

As it is standard in RBAC policies, a **role** is a job function that represents a set of privileges to perform actions on objects. An **object** is a data construct that is acted upon by a subject that has assumed a role. An **action** is an appropriate

Table 4.1.
PrBAC policy language

<Policies> ::= 'Begin' <policy-list> 'End'
<policy-list> ::= <policy> <policy-list> | <policy>
<policy> ::= <role-predicate> <object> <action> (<context>)
<role-predicate> ::= <role> (<confidence>) | <ranking>'('<role>')' (<confidence>)
<confidence> ::= <digit>
<digit> ::= ['0'-'9']
<ranking> ::= *equal* | *inferior* | *superior*
<action> ::= *read* | *write* | *delete* ...
<context> ::= <obligation> <location-constraint> | <obligation> <location-constraint> <proximity-constraints>
<obligation> ::= prior | while
<location-constraint> ::= <topology> <location> (<confidence>)
<topology> ::= in | out | adjacent ...
<proximity-constraints> ::= <proximity-constraint> <proximity-constraints> | <proximity-constraint>
<proximity-constraint> ::= <cardinality> <digit> <role-predicate> <location-constraint>
<cardinality> ::= at_least | at_most

```
<Policies>
    <policy-list>
        <policy>
        <role> General</role>
        <object>TopSecretDocument</object>
        <action>Read</action>
        <context>
            <obligation>while</obligation>
            <location-constraint>
                <topology> in </topology>
                <location> GeneralsRoom</location>
                <confidence>100</confidence>
            </location-constraint>
            <proximity-constraints>
                <proximity-constraint>
                    <cardinality>at_least</cardinality>
                    <digit>2</digit>
                    <role> General </role>
                    <location> GeneralsRoom </location>
                </proximity-constraint>
            </proximity-constraints>
        </context>
        </policy>
    </policy-list>
```

```
<Policies>
    <policy-list>
        <policy>
        <role> Private</role>
        <object>RestrictedDocument</object>
        <action>Read</action>
        <context>
            <obligation>while</obligation>
            <location-constraint>
                <topology> in </topology>
                <location> Room105</location>
            </location-constraint>
            <proximity-constraints>
                <proximity-constraint>
                    <cardinality>at_most</cardinality>
                    <digit>0</digit>
                    <role>empty </role>
                    <location> room105</location>
                </proximity-constraint>
            </proximity-constraints>
        </policy>
        </context>
    </policy-list>
</Policies>
```

Fig. 4.2. Two example proximity-based access control policies.

operation that can be applied to an object. We assume that users of our system may be mobile, and therefore, we incorporate usage controls regarding continuity of access [42]. An *obligation* specifies that certain constraints must be satisfied *prior* to or *while* accessing an object. A *topology* indicates a relation between the role and the **location** within the spatial domain. Often in enterprise environments, access to restricted resources is contingent on not only the presence (or absence) of other people, but the relation towards the individual requesting access. A **role-predicate** specifies a specific role or relational function that takes the role of the requesting user and outputs a ranking relative to that role (i.e., superior(roleOfRequestingUser)). Last, an entity designated to enforce a policy may need prerequisites to be fulfilled, at least to a certain in extent. A **confidence** indicates the numerical threshold at which a requirement must be fulfilled, otherwise anything below that threshold is considered a policy violation. For example, specifying a role (General) with a confidence constraint (80%) semantically states that the system must be "80%" sure that the current user is the General. Figure 4.2 provides two examples of access control policies to specify

the restrictions in SoD scenario and AOU scenario. The policy on the left refers to the SoD scenario: at least two Generals must be present in order to access the TopSecretDocument. The policy on the right refers to the AOU scenario: the Private can access the ResctrictedDocument only if no one else is around.

4.4 Threats and Assumptions

We make the following assumptions about the proposed system and the adversary. Each user, including the adversary, has full access to his/her device. Each device has been preauthorized by the IT admin for BYOD use. Preauthorization consists of verifying that (1) the device supports hardware-backed cryptographic key generation and storage and (2) the device's sensors, including Bluetooth, accelerometer, and fingerprint sensors, are functioning correctly. Consequently, we assume IT admins can be trusted. Each device must generate asymmetric cryptographic keys via Androids Hardware-Backed Keystore [31], in which the public key for that device is uploaded to a server for later use while the unexportable private key is stored securely in hardware. We trust the Android access control system, which includes the Android middleware and Linux Kernel, to correctly enforce all security policies. Physical security or video monitoring is employed to prevent the adversary from compromising proximity modules and entering the environment with foreign objects such as a non-secured phone. We only consider a passive adversary, and not active adversary. That is, the adversary has control of the communication channel, but is not able to inject new packets or compromise transmitted packets. The adversary is only able to relay packets transmitted between parties. In other words, the attacker posses standard Dolev-Yao capabilities [79]. We also consider insider threats to the organization. In particular, malicious employees may attempt to circumvent our context-aware access system through collusion. Last, we assume each proximity module has access to the public keys of Client devices, which can be retrieved on demand or during the installation of the proximity module.

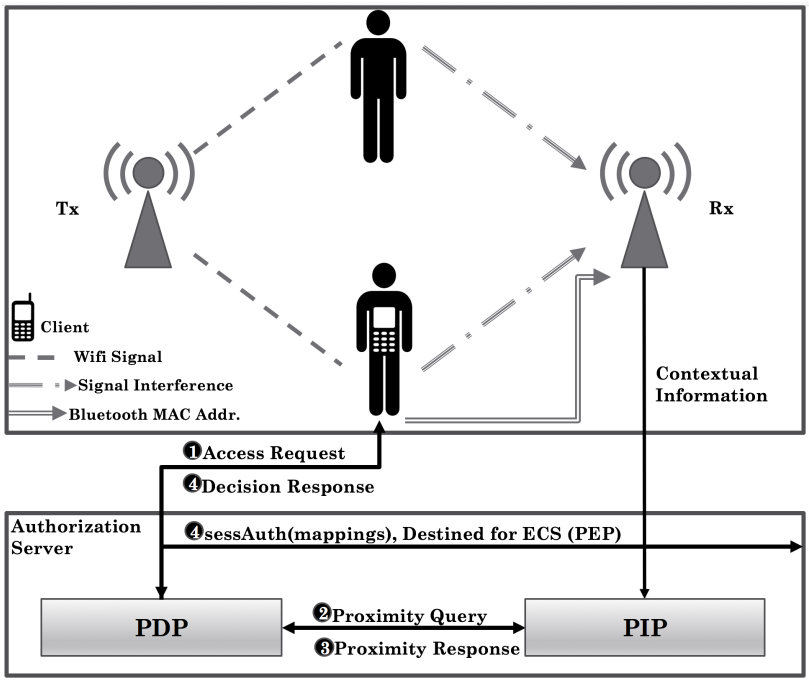


Fig. 4.3. CASSEC 2.0's access control framework.

4.5 System Design

In this section, we describe our CASSEC 2.0 platform that securely supports the SoD scenario and the AOU scenario described in Section 4.1. We adhere to design goals from the previous work, which include providing a secure, automated, and generalized architecture with responsibilities of each system component clearly defined. In CASSEC 2.0's architecture, we assume the least amount of trusted parties as possible. Our context-aware system proactively monitors and collects information about the environment in lieu of manual intervention by entities within that environment. Specifically, we do not rely on users, possibly malicious, to manually report their location. Therefore, we choose an infrastructure-based approach that uses wireless hardware to localize occupants within a monitored space. In the rest of the section, we define our interpretation of the term *proximity* and then provide an overview of the architectural components of CASSEC 2.0 and how they relate to our access control framework.

4.5.1 Proximity Zone

We rely on geographical proximity, which indicates that two entities are located within a certain distance in the physical space [43]. That is, in our work, *proximity* of a user is defined by a region of space monitored by a proximity module. The user must be within the region of space in order to gain access. We refer to this region of monitored space as a *proximity zone*. The level of precision in determining the location of a user and the proximity of other users is application dependent [9, 18].

4.5.2 Components

Access Control

Here we describe the architectural components tasked with enforcing our PrBAC policies.

Enterprise Content Server (ECS): The ECS, which acts as the Policy Enforcement Point (PEP), delivers enterprise resources to users who request access. By designing this component as a server, a heterogeneous network of end-users' devices can be serviced. Therefore, access to resources can be requested from desktop terminals or mobile devices.

Authorization Server (AS): The AS hosts the access control decision-making engine of the authorization framework. After a user has been authenticated by the AS via login credentials, it returns an authentication token to the Client device. The token, which is submitted to the ECS by the Client, is used to associate an authenticated user with authorized roles. The AS itself is composed of two sub-components: Policy Decision Point (PDP) and Policy Information Point (PIP). We discuss in more detail the construction of the authentication token and AS's sub-components later in Section 4.5.3.

Contextual Information

In order to extract contextual information from the environment, we take both an active and passive approach. We use the terms *active* and *passive* to indicate whether or not users are required to physically interact with the entity collecting contextual information. The components involved in contextual information acquisition are as follows:

Client: A Client is a device used to request access to a resource by a user. If the request is granted, a user can view the data on the device (e.g., desktop terminal or mobile smartphone). Unlike their desktop counterparts, smartphone devices allow mobility with respect to embedded sensors and network connectivity. Consequently, in our prototype system, we take an active approach to user verification via biometric authentication by utilizing a smartphone as the Client device. That is, the Client is also designated to verify that the current user of the device is the true owner of the device. We note that solutions have been developed that take a passive approach to the collection of biometric features, which may be more secure. If we were to take a passive approach to biometric authentication, the example policies in Figure 4.2 would also include confidence thresholds under the *role* specifier since the AS is designated to evaluate if policies are adhered to. We discuss this further in Section 4.7.

Proximity Module (PM): The role of the PM is to collect and analyze contextual information in order to detect the proximity of users. This detection process occurs periodically, and proximity-related information is sent to the AS. Although a PM is the set of physical devices that determine proximity, we consider them as independent of the PIP as the PIP is the entity that directly communicates with the PDP. Users do not physically interact with the PM in our prototype system, and therefore it is considered passive.

Our architectural components are shown in Figure 4.1. We do not discuss cryptographic schemes to protect network communication between the entities in our system

model. We assume that an underlying secure network infrastructure is in place, as usual in enterprise environments. Although the figure only shows one PM and consequently only one proximity zone, in practice an enterprise building will have multiple PMs, possibly one for each room.

4.5.3 Access Control Framework

The **PDP** is the specific entity that is delegated to make access decisions. It maintains a database of PrBAC policies. Given these policies, the PDP first verifies if someone is a user of the system. The PDP then retrieves the latest information regarding the user's location and the presence of other users from the PIP. Such information allows the PDP to determine the set of authorized geo-spatial roles if proximity constraints are satisfied. Next, the PDP constructs and returns to the Client an authentication token. The token, at minimum, contains a generated temporary ID. It may also contain an expiration date. As such, the token is utilized as a session identifier. Last, the PDP maintains a database mapping of session IDs to the set of active authorized geo-spatial roles for each user. This mapping is *also* sent to the PEP each time a role is authorized.

The **PEP**'s role, implemented as part of the ECS, is to enforce proximity restrictions for enterprise content. During a request, a Client submits an authentication token to the ECS. The PEP extracts the temporary session ID from the token. The PDP continually updates the PEP of mappings of session IDs to a set of active authorized geo-spatial roles. First, the mapping makes it possible to enforce access restrictions according to the roles associated with that ID. Second, it also enables it to service multiple Client devices simultaneously. Third, this design anonymizes users as the PEP does not have any information that identifies users such as locations and credentials.

The **PIP**'s role is to store and maintain contextual information about an enterprise's proximity zones. Each PM, after co-proximity authentication of Clients, is

required to transmit four pieces of information to the PIP: a proximity zone identifier, the number of people detected, a list of captured UIDs[1] and corresponding RSS values of BLE beacons, and a timestamp. The PIP then records the collected data into its context database. Instead of the PIP polling the PM for information, we minimize communication by requiring that the PM updates the AS only when characteristics of the proximity zone changes. In addition, this clear designation of duties also minimizes overhead in both the PM and AS. Considering the dynamic nature of the environment, the PIP must update the PDP as frequently as the occurrences of updates to the context database. Such updates allow the PDP to continuously check for any instance of proximity-based violations by users. At the time of violation, the PDP invalidates the relevant session ID mappings by associating existing session IDs with newly recomputed *authorized* geo-spatial roles, if any, according to PrBAC policies. The PDP then remotely informs the PEP of invalid mappings while providing new authorized ones. The PDP can also alert the enterprise's administrators to take appropriate action. Such a design makes the system completely automated by only requiring users to be authenticated once by the AS.

4.5.4 Co-Proximity Authentication

Radio signals permeate through walls, and therefore it is possible that two PMs located in two adjacent rooms may detect the same Client device, even though in reality the Client is located in one of the rooms. However, such signals exhibit attenuation as they pass through walls. We leverage this phenomena to determine the likelihood that a Client is in a given room. In particular, we analyze the RSS values from BLE beacons to initiate the co-proximity authentication process, which determines that a *legitimate* Client is within a specific proximity zone.

Overview. The protocol to authenticate the user's co-proximity to a PM consists of two phases: the initialization phase and the location authentication phase. First,

[1]We assume that each user of the system has an identifier unique to that user.

the initialization phase establishes a temporary session key (SK) securely shared and only accessible between a PM and a Client. Next, the SK is later used in the location authentication phase, in which a timed challenge-response protocol is executed. The crux of authenticating the user's co-proximity is analyzing the content of the beacon as well as the measured round trip time. We explain both phases in detail below.

Initialization Phase. The initialization phase is activated once the user enters Δ_2, that is, the concentric region as indicated by BLE's *Near* ranging measurement (i.e., between 1-3m from the PM as displayed in Figure 4.4). Placing a PM at the center of an average sized conference room (e.g., ~6m x 6m) allows the PM to detect and monitor the movements of any Client device that enters the room. In addition, positioning in such a way may minimize the overlapping of concentric regions of two adjacent PMs' proximity zones. Once the Client enters Δ_2, the PM generates a temporary SK and encrypts it with the Client's public key. As stated in Section 4.4, the public key can be retrieved from the authorization server on demand or during the installation of the PM. The SK is a one-time pad which consists of a string of bits generated using a cryptographically secure pseudo-random number generator. The encrypted SK (Step 1 in Figure 4.5) is then sent to the Client via the AS on a secure out-of-band channel, which is then decrypted at the Client using the Client's hardware-bound private key. The Client finalizes the initialization phase by responding with an acknowledgement of message receipt, which is relayed back to the PM. We note that there are a number of methods to securely exchange temporary session keys. For example, the PM could purely rely on BLE beacons to transmit the encrypted SK, thereby minimizing communication with the server. However, we did not choose this mode of transmission because of limited data capacity in BLE beacon's advertising data structures [23].

Authentication Phase. The PM will continue to monitor and track the Client's movements. The authentication phase is activated once the user enters Δ_1, that is, the concentric region as indicated by BLE's *Immediate* ranging measurement (i.e., less than 1m from the PM). At this point, the PM initiates a timed challenge-response

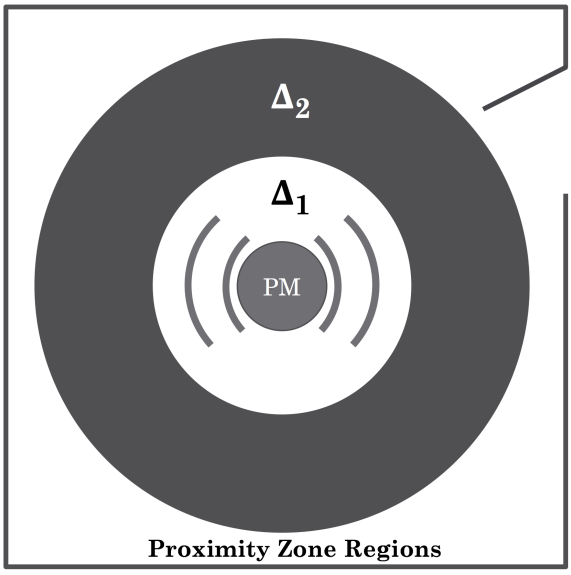


Fig. 4.4. A Proximity Module's proximity zone regions.

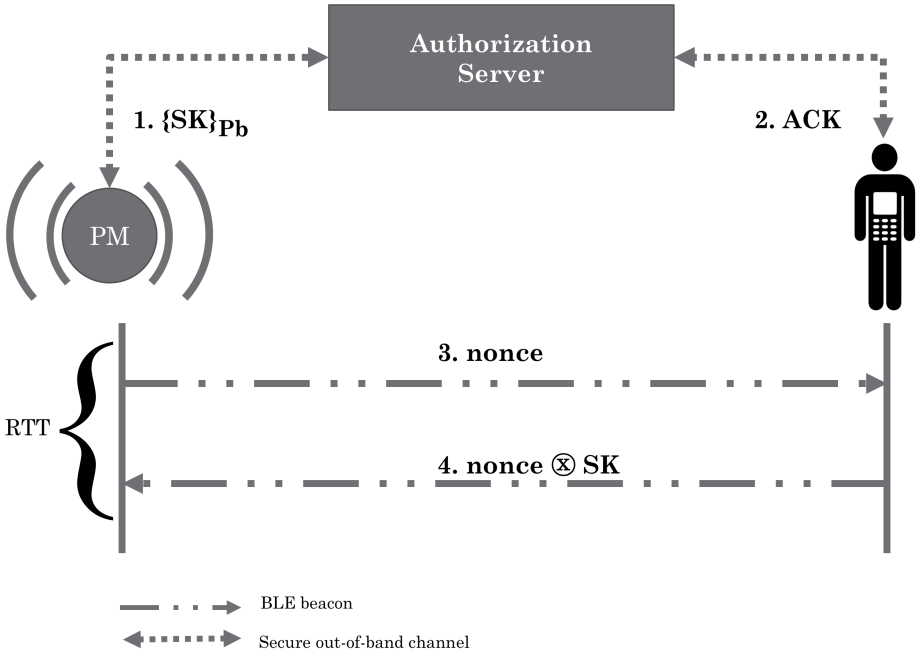


Fig. 4.5. CASSEC 2.0's co-proximity authentication protocol.

protocol with the target Client. The PM generates a fresh nonce (string of random bits), embeds the nonce into a BLE beacon, and transmits the beacon. Upon suc-

cessful transmission, the PM records the time of transmission and precomputes the expected response. Upon reception, the Client calculates an XOR value, using the nonce and the SK as the two inputs. XOR operations are simple and require minimal CPU cycles to compute as opposed to other widely-used cryptographic schemes with non-negligible encryption/decryption times [80]. Leveraging XOR operations thus allows the Client to minimize the time to calculate a response to the challenge, and subsequently package and transmit the response within a BLE beacon. Upon reception of the Client's response beacon, the PM calculates the RTT value and verifies that the precomputed value matches the received value. If the values match and the RTT is less than or equal to a specified threshold (RTT_{TH}), the PM informs the AS that the specific Client's location has been authenticated with 100% confidence, otherwise the PM and Client must repeat both the initialization and authentication phases. We discuss how we determined RTT_{TH} in Section 4.7. Both phases must be repeated since information about the temporary session key that is generated in the initialization phase is leaked in the authentication phase. An attacker can simply perform an XOR of the nonce, which was transmitted in cleartext, and the Client's response beacon to calculate the session key.

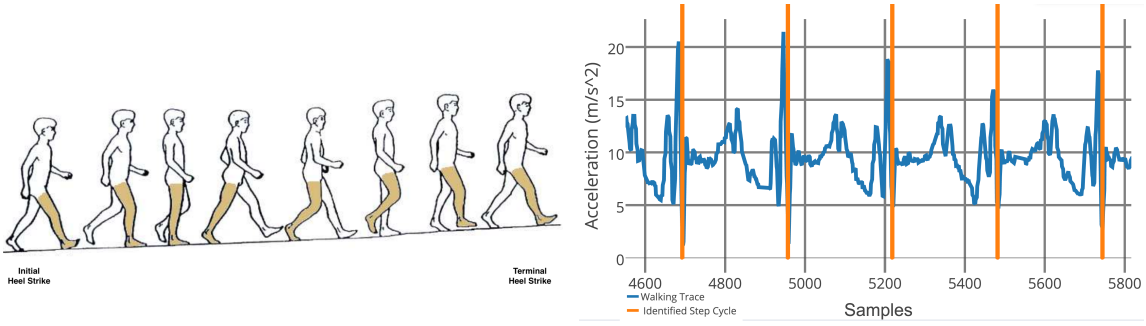
To address circumstances resulting in proximity zones partially overlapping, we take a binary approach. In the case that multiple PMs detect and authenticate a Client via BLE beacons simultaneously given that BLE beacons can travel several meters, for simplicity, we classify a Client to be in one of the corresponding rooms with 100% confidence only if information sent by a PM meet two conditions: (1) the RSS value (measured from the beacon) is the strongest of all RSS values detected by other PMs; (2) the number of people detected and captured UIDs match. Otherwise, there is 0% confidence in the Client's location. The left policy in Figure 4.2 provides an example of PrBAC policy that specifies that the entity enforcing the policy must determine that the General is in fact located in the *GeneralsRoom* with 100% confidence to grant access to the *TopSecretDocument*.

4.5.5 Biometric Authentication

User verification via biometric authentication is isolated to the only active component in our prototype system, that is, the Client. We specifically develop an Android application that leverages the smartphone's capabilities to scan fingerprints and measure acceleration in order to achieve physiological and behavioral biometric authentication, respectively. User verification is abstractly a two phase process (see Section 4.2): the enrollment and authentication phases. With respect to security, it is vital that enterprise administrators proctor the enrollment phase in-person to confirm that biometric measurements taken by a Client device match the true owner of the device. Fingerprint scanning and the collection of walking traces are achieved and easily integrated into our application using Android's Fingerprint Authentication and Sensor Manager APIs[2]. To ensure the privacy of users, the fingerprint and gait templates constructed during the enrollment phase never leave the device.

We implemented the behavioral component of the user verification framework in a similar fashion as proposed by Ren *et al.* [76]. The framework consists of three components, which can be abstracted to the enrollment and authentication phases previously mentioned: Step Cycle Identification, Step Cycle Interpolation, and Similarity Comparison. The components are built on the fact that human gait should be cyclic in nature, and hence should exhibit high correlation. Here, a step cycle is the period defined by the two consecutive heel strikes on the same leg (see Figure 4.6(a)). The Step Cycle Identification component identifies step cycles in a walking trace, and then uses the extracted features to construct and store a biometric template. Although smartphone accelerometers provide signals in three dimensions, the framework extracts only the signals from the vertical direction to identify impacts caused by heel strikes. Figure 4.6(b) displays a walking trace with identified cyclical heel strike impacts. Users usually walk at varying speeds, which would negatively impact the verification process if the template and the runtime measurements are of

[2]We do not elaborate on implementation details as Android provides detailed instructions and samples to utilize Android Fingerprint Authentication and acceleration measuring [31].



(a) (b)

Fig. 4.6. 4.6(a) is an illustration of a complete gait cycle from the initial heel strike to the terminal heel strike (from [81]). 4.6(b) displays preliminary measurements of accelerometer signals of a walking trace in the vertical direction we collected using a Nexus 6P smartphone. Orange lines indicate step cycles identified by heel strike impacts.

traces with different speeds. Addressing this potential problem, the Step Cycle Interpolation phase enables robust user verification by normalizing identified step cycles of different lengths into fixed lengths. Figure 4.7 displays the interpolated accelerometer signals, recorded using a Nexus 6P, of slow (slower than 0.7 m/s), normal (about 0.7 - 1.1 m/s), and fast (about 1.1 - 1.4 m/s) walking traces to a fixed length of 400 samples. The figure demonstrates that step cycles are highly correlated regardless of walking speed. Last, user authentication is performed in the Similarity Comparison phase, which utilizes a weighted Pearson correlation coefficient (PCC) based method.

We apply defense-in-depth within the authentication phase. We first use Pearson correlation coefficients when computing the similarity between the gait template and the walking trace runtime measurements. Users are only verified if similarity scores are above a predefined threshold (see Section 4.6.4). If similarity scores fall below the threshold, the user is then required to perform authentication via fingerprint scanner when the user attempts to access the phone. We are unable to set a threshold for fingerprint authentication as we rely on the Client device's integrated fingerprint solution. If the user neither can be verified via behavioral nor physiological biometric

authentication, the Client ensures that sensitive enterprise content is inaccessible by locking the device[3]. In addition, the Client can alert enterprise administrators for possible user spoofing.

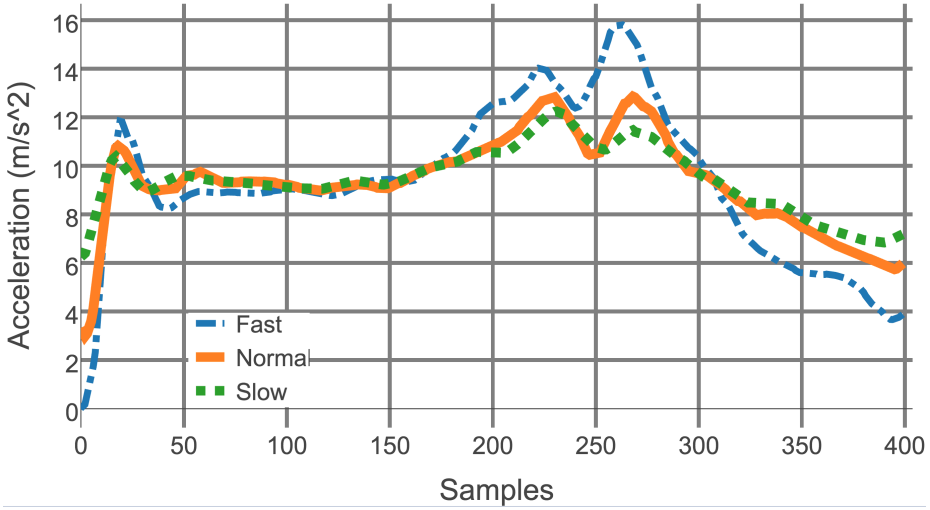


Fig. 4.7. Step cycle interpolation applied to walking traces collected using our Nexus 6P smartphone at three different speeds: slow, normal, and fast.

4.6 Prototype Implementation

4.6.1 The ECS

The ECS was implemented in PHP and hosted on a remote commercial server. The resources that it could serve to Clients were simple text files. We implemented user interfaces (UI) in order for Clients to request access to specific files. The ECS provides a function that can be remotely invoked via URL: *sessAuth(mappings)*. The function is invoked by the AS to update the ECS regarding the active geo-spatial roles for Clients in the event that location updates reflect proximity violations.

[3]We build an application on the Client using Android's Device Administration API, which includes the device lock ability [31].

4.6.2 The AS

The AS was also implemented in PHP and hosted on the same server as the ECS. We implemented the UI in order for Clients to pass in authentication credentials via a login page. The AS provides two functions that can be remotely invoked via URLs: *auth(user,psswd)* and *addEntry(pzoneID,numOfPpl, UIDs+RSSs, time)*. The first function is invoked by a Client via the UI and the second is invoked by the PM to update proximity information within the context database.

4.6.3 The PM

As in any basic positioning system, a PM incorporates a transmitter and a receiver. We define a transmitter as a wireless-enabled device that is a source of contextual information regarding the occupants within a proximity zone. A receiver is a wireless-enabled device that acts as a sink for such contextual information.

We utilize BLE-enabled smartphones and WiFi access points (APs) as transmitters. In regards to smartphones, we embed three values into BLE beacons to support co-proximity authentication. Generally, these devices periodically broadcast their 48-bit Bluetooth MAC addresses with a less than 10 meter range indoors when Bluetooth is enabled. However, since Android 6.0, the MAC address found in a BLE beacon is replaced with a random value at various intervals to protect user privacy [31]. User privacy is not a concern within the enterprise scenarios that CASSEC targets. Disabling this feature would require modifying the Android OS, which reduces the deployability of our solution. Therefore, we cannot rely on this hardware address to identify users. Instead, in CASSEC 2.0, we embed a 48-bit UID into BLE's local name data structure using Android's *BluetoothAdapter.getDefaultAdapter().setName(UID)*. The BLE beacon data protocol is limited with respect to the amount of custom data we are able to embed within a beacon. As a result, the nonce, as well as the one-time pad SK generated by the PM, is restricted to 12-bytes. With the remaining space, we embed a 16-byte service UUID which enables Clients and Proximity Modules to com-

municate under a beacon service. We require that users of the system permanently enable their smartphones' Bluetooth. Such a requirement can be easily enforced by Enterprise Mobility Management services [20]. WiFi APs transmit data over signals that can be measured. However, such signals are significantly influenced by the environment. We rely on the interference of signals as a result of human activity to determine the number of occupants in a proximity zone.

The PM was implemented as two physical devices: a Pixel C tablet running Android Oreo (API 26 v8.0) and a laptop using Python running Linux. For brevity, we refer to these devices as simply the PM. The PM was charged with periodically scanning signals produced by BLE and WiFi devices. Beacon scan settings were set to *SCAN_MODE_LOW_LATENCY* from the *ScanSettings* API, while WiFi signals were scanned every 10 seconds. The PM extracts the UIDs of beacons from nearby occupants' smartphones. The UIDs are used as proof-of-location once co-proximity authentication has been established, which determines *who* is in a given space. The PM also measures the received signal strength from a designated WiFi AP. The receiver processes the measured WiFi RSS value and determines *how many* occupants are in a given space. Last, the receiver publishes the UIDs, beacon RSS values, and the number of occupants to the authorization server only when previously collected contextual information changes.

We note that the various components of the CASSEC 2.0's system architecture can be integrated into the same physical component when implemented. For example, a smartphone mobile device can act both as *Client* and transmitter because the same device used to request access to a resource is the same device that periodically broadcasts its Bluetooth data structures. Similarly, a desktop terminal can act both as *Client* and receiver because it can also be used to scan and process WiFi and Bluetooth contextual information.

4.6.4 Use Case

In this section, we evaluate features of the CASSEC 2.0 prototype system in order to provide clear insights into addressing the issues raised in Section 4.1. We measure the performance of the system's biometric and co-proximity authentication components to prove the feasibility of securing enterprise content under a proximity-based access control model.

Deployment

We deployed our hardware and tested our prototype system in a two bedroom apartment whose layout is shown in Figure 4.8. We now briefly describe the hardware utilized in our platform.

The Wireless-N (802.11n) WiFi AP transmitter was a Motorola SURFBoard SBG6580, indicated in blue, that supports two frequency bands which are 2.5GHz and 5.0GHz. We chose the higher-frequency band to take advantage of additional channels that are less prone to interference than 2.4GHz. The receiver was a Dell Latitude E6430, indicated in green, equipped with a BCM4313 802.11bgn wireless network adapter and a Dell Wireless 380 Bluetooth 4.0. The transmitter and the receiver were placed 3 meters apart and were elevated 1 meter above the floor. The Bluetooth-enabled transmitters used in our study were Samsung S3 GT-i9300 and Nexus 6P. The Nexus 6P, which has a fingerprint scanner and an accelerometer that supports a 200Hz sampling rate, was used for biometric collection and analysis.

Use Case Evaluations

Evaluation 1: Selecting Frequency Channel. Given a wireless link between a transmitter and a receiver, an individual crossing the line of sight between the two communicating wireless sensors affects the RSS measured by the receiver. However, the change in RSS depends on the frequency channel [9]. Our goal is to determine

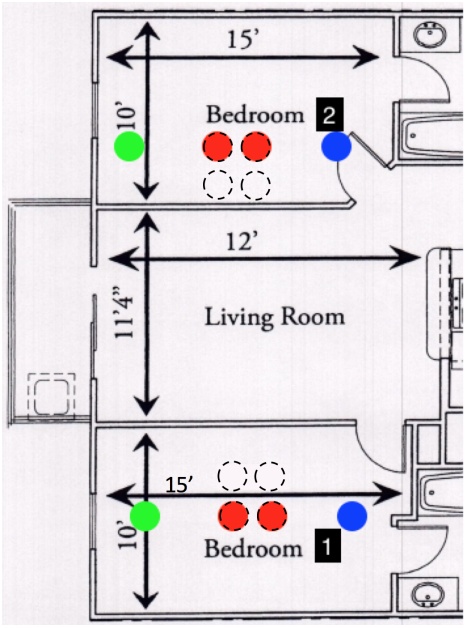


Fig. 4.8. The blueprint of a two-bedroom apartment in which the prototype system had been deployed. The blue markers and green markers indicate the positions of WiFi access points and laptops, respectively. The dotted lines indicate the two possible positions for each human, and transitions simply require moving two steps without changing body orientation. The red dots represent the current positions of the humans standing still while facing the laptop.

which channel is the best for detecting human activity based on our particular WiFi-enabled devices. We test 2 non-overlapping 40MHz channels: Channel A (5180MHz) and Channel B (5220MHz). The experimental setup is as follows. Throughout the complete test, we continuously measure the RSS value sampling twice per second. Every 30 seconds we change the number of individuals obstructing the LOS by 1 starting from zero to two, and then in a decreasing fashion. The occupants were situated equidistant from each receiver. A Python script was written to automatically begin the test. The tests were conducted in Bedroom 1.

The results in Figure 4.9 demonstrate that there is no significant difference in measurement variation in human-induced interference in RSS signals between Channel A and Channel B. At first, Channel A appears to be more consistent as the level

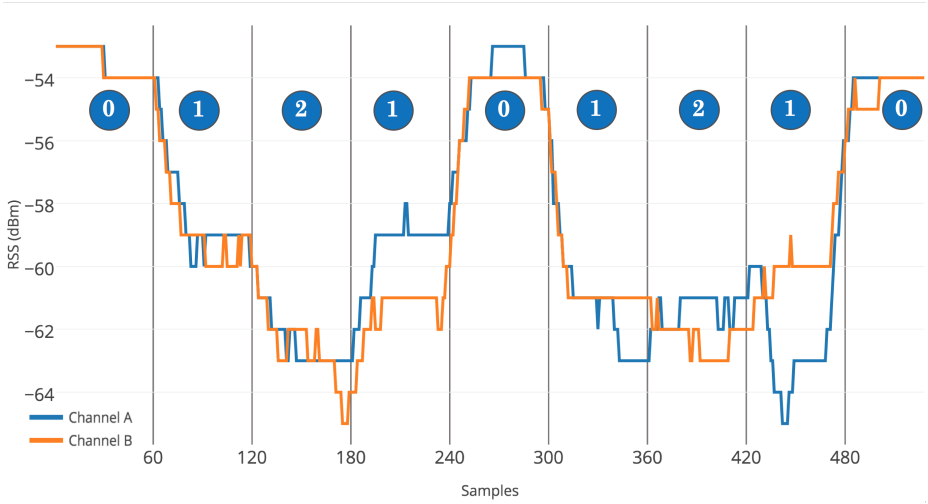


Fig. 4.9. RSS measurements of wireless links on different frequency bands when human bodies obstruct the line of sight (LOS). The blue circles indicate the number of humans in the LOS within each 60-sample period (i.e., every 30 seconds).

of signal interference in samples 60 - 120 aligns with values in samples 180 - 240 when the number of individuals increases from zero to one and two to one, respectively. This is not observed in Channel B during that period. However, the values for Channel A appear to indicate the presence of a number of individuals different from the number of individuals actually present from samples 330 onward. This fluctuation is not observed in Channel B. Although Figure 4.9 shows the results of only one complete test, we performed this test 3 times and observed similar changes in values. Given these observations, we select Channel B as a means for testing in the rest of the study.

We also make some general observations about human-induced RSS changes. We observed distinct variances in signal strength almost every 30 seconds (multiple of 60 units in Figure 4.9). First, by initiating the test with no individuals obstructing the LOS, we were able to establish a baseline for the signal strength between the transmitter and receiver. The RSS value remained always constant within that time period up until to two seconds after the 30 second mark. That is, using our existing

hardware, we were able to determine that once we increase the number of individuals by one, the individuals must remain in the LOS for *at least* one second for the receiver to observe some interference from human activity. Such phenomena was also observed at the beginning or end of each period. Second, regardless of the selected channel, when the LOS is obstructed by an individual the RSS on average decreases. In addition, distinct dBm drop ranges exist depending on the number of individuals. Therefore we can infer the presence or absence of humans based on RSS' ranges. For example, in Channel A, we consistently observed a drop range of 6-8 dBm between 30-60, 90-120, 150-180, and 210-240 seconds. We note that our observations are likely to change using different WiFi-enabled hardware.

Evaluation 2: WiFi Detection Accuracy. The goal here is to test the WiFi localization component of our PM. Specifically, we implemented a simple algorithm to detect the number of people within the LOS based on our observations of human-induced RSS changes from Evaluation 1. The setup to this test is similar to the setup for Evaluation 1, except that we perform the test in *both* Bedrooms 1 and 2. We conduct the test on Channel B.

Table 4.2 displays the results. The system was able to detect with strong accuracy (89%) the number of occupants obstructing the line of sight in Bedroom 1. At certain points, sporadic fluctuations occurred that caused the system to return an incorrect number. On the other hand, the system was only able to detect occupancy with 43% accuracy in Bedroom 2. After further analysis (by performing Evaluation 1 in Bedroom 2), we observed the human-induced interference was slightly different in RSS levels. Although the physical layouts of Bedroom 1 and 2 are identical, there may be other (unseen) environmental factors that also influenced the RSS levels to slightly differ between the two rooms. For example, such factors may include overlapping wireless networks (possibly using the same channel) from neighboring apartments, appliances and electronics emitting radio frequency interference, and simply walls and floors blocking wireless signals in different ways depending on the location of access points [71].

Table 4.2.
WiFi detection accuracy.

Location	**Detection**
Bedroom1	89%
Bedroom2	43%

Evaluation 3: Gait Recognition Detection Latency. The goal here is to determine the required length of a walking trace to identify the true owner of a Client device using gait recognition. CASSEC was developed with certain enterprises in mind that desire high assurances that sensitive enterprise content on end-users' devices is well protected. We therefore set the pre-defined threshold to 0.8. A single user participated in this study using a Nexus 6P smartphone device to record and analyze accelerometer values. In order to execute the test, the user performed the enrollment and authentication phases. We first collected from the user six 60-second walking traces at normal speed: the first and subsequent five traces to be used for gait template construction and runtime measurements in the enrollment and authentication phases, respectively. Then, in the enrollment phase, $n, n = 5, 6, ...60$, gait templates were constructed for the user which were derived from extracting n seconds from the first trace. Next, in the authentication phase, we extracted n seconds from each of the subsequent traces to analyze and compare biometric templates with runtime measurements that are of corresponding lengths. In its entirety, we repeated this test twice; differentiating the two by placement of the smartphone device: on the hip and within the pocket.

Figure 4.10 displays the results of the test. We first note that all similarity scores produced were at least greater or equal to the predefined threshold. On average, the system was able to detect that the current user of the Client device is the true owner with approximately 91% and 89% confidence (i.e., similarity score) when the phone is placed on the hip and within the pocket, respectively. The system only required five

seconds of each walking trace to make such an assertion. We also observe that longer traces eventually produce higher levels of confidence in identifying the true owner because more gait features were extracted, and therefore more identifying features can be determined during the authentication phase.

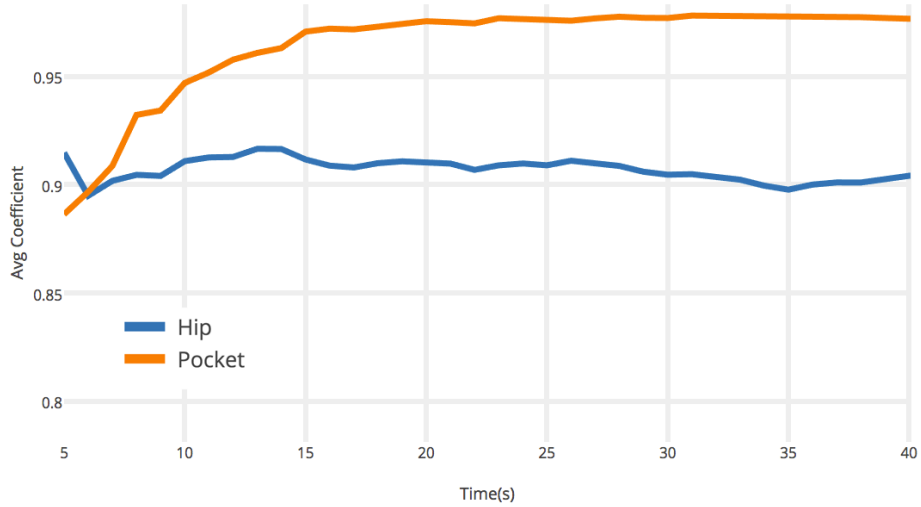


Fig. 4.10. Average similarity score by varying the duration of user profile trace and runtime measurement trace.

Extracting more gait features over a longer period of time produces higher levels of confidence while in the pocket as compared to on the hip. Upon further analysis of individual traces from both the hip and pocket, it appears that hip traces have increased oscillations that are not quite periodic. Particularly, we observed that there are more variations in-between the heel strikes as compared to pocket traces. First, the Step Cycle Identification component may falsely identify when the user's leg comes in contact with the ground if oscillations closely resembles heel strikes. Second, the gait recognition program assumes a cyclic nature, and thus if no repetition occurs within these sporadic oscillations, correct heel strikes, which occur outside of the oscillations, may not be properly analyzed as well. It is evident that the hip is continuously gyrating, and therefore, has a periodic motion. However, we believe that the increased (and erroneous) variations are the result of the method in which

we attached the device to the hip. While the device is securely fastened and flushed with the hip clip in order to minimize erroneous movement of the device, it is difficult to replicate such a secure grip with the hip clip itself as it is attached to the wearer's clothing. However, while placed in the pocket, the device is resistant to minor shuffling because it is pressed against the user's clothing and leg. Nevertheless, the results of this test demonstrate that feasibility to detect the true owner of the Client device with high confidence when placed within the pocket or attached to the hip, even considering the inherent erroneous data that is acquired while the device is attached to the hip.

As stated in Section 4.2, the development of gait recognition techniques for user verification is outside the scope of this work. We emphasize that this work is a feasibility study that demonstrates the application of biometric techniques such as gait authentication to securing enterprise content under a proximity-based access control model solely using one mobile device. We refer readers to the work by Ren *et al.* [76] for an extensive user evaluation of the gait authentication technique we have leveraged.

Evaluation 4: Robustness Against Different Walking Speeds. The goal here is to test the robustness of the system against various walking speeds. We applied the same methodology as Evaluation 3 with an exception. We also compare the biometric template constructed from the normal walking trace to five runtime measurements collected from the user for both slow and fast walking speeds.

Figure 4.11 displays the results of the test. Similarity score calculations derived from the normal and fast walking traces were at least greater or equal to the predefined threshold of 0.8. However, in a few instances, our system was unable to authenticate the current user as the true owner of the device. 07% of the similarity scores calculated, which we consider negligible, fell within the range of [0.7,0.8). Nevertheless, we can observe in any walking trace, including the slow walking trace, that there is a positive correlation between the trace length and the similarity scores produced.

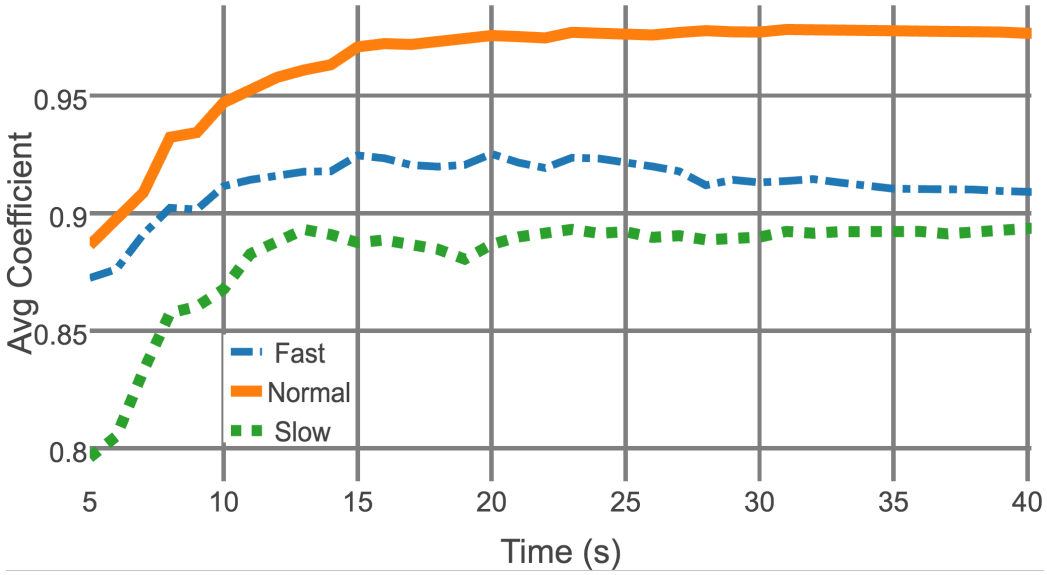


Fig. 4.11. Average similarity score calculated by comparing the normal walking biometric template with both the slow and fast runtime measurement walking traces.

That is, the system can reliably determine the user with increasing confidence over a longer period of time.

Evaluation 5: Capturing BLE Beacon RTT Values. One type of distance-bounding technique uses the elapsed time between two devices for distance estimation. Our goal for this test is to apply such a technique to BLE and determine if indeed that the round trip time of beacons is a function of distance. We exchanged beacons between two BLE-capable devices (Pixel C tablet and Nexus 6P smartphone) and recorded 100 RTT values at various distances. The devices were laid down across a wooden desk with the front screen facing upwards. Figure 4.12 shows the distribution of RTT values measured between the two devices. We note that displayed values reflect distance estimation as implemented in our co-proximity authentication. We first observed that most of the RTT values, at each distance, are centered around the median (the black line within the inner quartile range). For example, at distances of 1ft, 4ft, and 6ft, the RTT values are centered around approximately 81ms ($\pm$ 1ms),

while at a distance of 2ft, RTT values are centered around 77ms. We also observed that the IQR, the box that spans the first and third quartiles, are centered in between 72ms and 86ms. Consequently, no significant statistical variations of RTT values exist when the PM and the Client device executed the timed challenge-response protocol at distances between 1-6ft. Moreover, we produced similar results when we applied the same experimental process, but instead separated the devices with a 1ft wall. We discuss the security implications in Section 4.7.

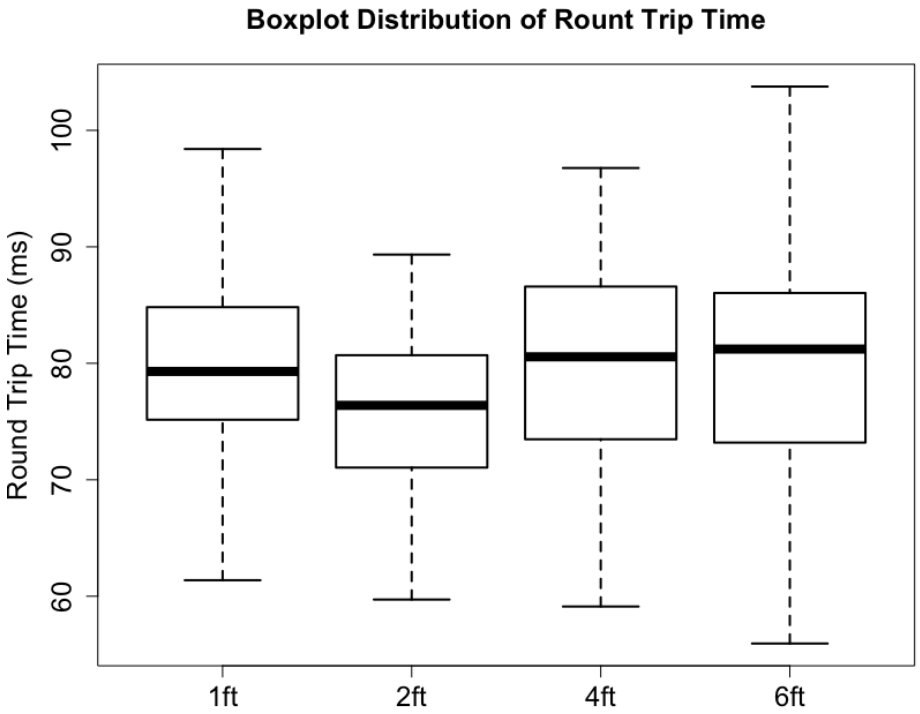


Fig. 4.12. Distribution of round trip time of 100 Bluetooth Low Energy beacons each at various distances, exchanged between a Proximity Module and a Client.

4.7 Security Analysis

In this section, we present a security analysis of our CASSEC 2.0 platform to analyze attacks aiming at circumventing its PrBAC restrictions. Below, we provide

various attack vectors that could be used and, subsequently, a means to mitigate the threat or minimize the attack vector surface.

4.7.1 Bluetooth Manipulation

In our previous work, when a PM publishes a MAC address to the PIP, it attests that a specific individual is at a specific proximity zone. A malicious user may attempt to root his/her device and modify the MAC address in order to impersonate another user of the system. In CASSEC 2.0, however, such malicious modification of MAC addresses would do no harm for two reasons: (1) we rely on UIDs that are dynamically embedded into data structures within BLE beacons; and (2) Android Oreo (API 26 v8.0) automatically randomizes the MAC addresses of beacons. Moreover, an attacker that roots his device to dynamically alter beacon UIDs (through a modified and unauthorized custom OS) to impersonate a legitimate user would fail the challenge-response protocol for several reasons including the attacker's inability to access the legitimate user's private key, which is bound to the user's Client device hardware (i.e., not exportable). In addition, Samsung has demonstrated via Samsung KNOX 2.0, a custom Android OS intended for enterprise environments [20], hardware and software security features that leave the device inoperable once it detects a root attack, which is a sufficient mechanism to defend against malicious modification of the OS.

One attack that malicious users may attempt is masking their smartphones' Bluetooth peripheral services by either disabling the Bluetooth or simply leaving the device in another room. Although we require that Bluetooth be permanently enabled on users' devices, we do not incorporate an enforcement mechanism within the phone to meet such requirement. However, our system is able to detect if the violation of such requirement occurs. The WiFi localization technique is able to determine the number of occupants in the room. If the number of occupants and the number of UIDs, which are published to the PIP, for a given room do not match, the PDP will infer such malicious behaviour and subsequently revoke access to resources. In ad-

dition, appropriate actions can be taken by the system administrator. We also note that Android provides Device Administrator APIs for BYOD scenarios, which allow enterprises to take control of sensitive resources and modify system configurations on their employees' devices. Through such APIs, an enterprise can then permanently enable Bluetooth services.

Another attack vector involves an unauthorized individual obtaining an authorized user's phone, whether by theft or voluntary provision. If such an attack occurs, then the unauthorized individual can gain access to restricted resources. In reality, this sort of attack exploits social engineering and/or insider threats that are usually already covered as part of an enterprise's global security efforts. Nevertheless, in our extended system, we mitigate this previously unaddressed attack vector by incorporating mechanisms that are able to determine biometric signatures for every user in the system. In particular, we employ behavioral (i.e., dynamic gait analysis) and physiological (i.e., fingerprint analysis) biometric authentication, which ensures that unauthorized users will not be able to bypass security by using someone else's device.

One of the objectives of this work is to address context monitoring issues including adversarial context manipulation via passive attacks (e.g., malicious relay of BLE beacons). However, we emphasize that if we relax assumptions stated in Section 4.4 and elevate the adversary's capabilities to active attacks, we envision two active attack vectors the adversary could employ that would *not* circumvent the security of the system: packet injection and Denial-of-Service (DoS). An astute active attacker would determine that an advantage cannot be gained by injecting packets at either Step 3 or Step 4 of the co-proximity authentication protocol (Figure 4.5). Intercepting the BLE beacon that encapsulates *nonce* at Step 3 and transmitting a new malicious beacon that encapsulates a *nonce'*, which would be now received by the Client, is unnecessary. The original *nonce* is transmitted in cleartext, thereby allowing the adversary to simply record the observed value, which may be potentially used in Step 4. However, the attacker again would not need to inject packets in Step 4 since the attacker has acquired the information needed (i.e., *nonce*) to extract and calculate the

temporary session key (SK) from the BLE beacon sent by the Client. Knowledge of *nonce* and SK also does not violate the security of the system (see Section 4.5.4). In summary, injecting attacker-generated BLE beacons would serve no purpose towards the goal of fooling the CASSEC 2.0 system into establishing co-proximity between the PM and the Client.

Given that intercepting and subsequently injecting malicious BLE beacons between the PM and the Client would serve no purpose towards circumventing co-proximity authentication, an active attacker may instead rely on DoS attacks. A malicious user may attempt a DoS attack by acquiring a high-powered Bluetooth-enabled device [82–84]. Specifically, the user first adjusts the special device to mimic his original (or another user's) smartphone's UID, and then boosts the signal strength. As a consequence, receivers in different rooms within a certain radius may incorrectly publish the proof-of-location. Therefore, the PDP will believe that multiple violations are occurring. First, the system is inherently resistant to such attack. Because of signal attenuation, proximity modules, which have lower transmission capabilities than of the adversary's high-powered device, may not be able to transmit the challenge beacon to the malicious device, which may be potentially far from the proximity module. However, if reception of the challenge does occur, several methods could be employed to counteract this attack. For example, our study shows that the majority of beacon RTT values fell between 72ms and 86ms. The PM could invalidate the challenge, thereby invalidating the corresponding response, after 86ms has elapsed. We emphasize that we are describing this DoS attack under the assumption that the attacker is able to somehow relay the temporary session key (once it has been decrypted) from his Client to the special device, otherwise the objective of the DoS attack would be to simply waste computing resources by repeatedly initiating the co-proximity authentication process.

The results in Section 4.6.4 demonstrated that no significant statistical variations of RTT values exist when the PM and the Client device executed co-proximity authentication at distances between 1-6ft. Consequently, time-based distance esti-

mation techniques that rely on BLE beacons as its underlying technology are not reliable methods for differentiating between adjacent proximity zones *within* an enterprise environment. However, such techniques may be resistant to an adversary's attempts to execute relay attacks when the Client is far away, that is, *outside* the enterprise environment. Let us assume the adversary's attack takes the form of a ghost-and-leech attack vector [85] in which the adversary employs two relay devices ($\mathcal{A}_{\mathcal{PM}}$, $\mathcal{A}_{\mathcal{C}}$) that are each within 6ft of the PM and the Client, respectively, and the two malicious devices communicate over a high-speed connection. Let us also assume that $\mathcal{A}_{\mathcal{PM}}$ and $\mathcal{A}_{\mathcal{C}}$ have similar hardware and software to that of the Client and PM, respectively. The total RTT (RTT_T) is the sum of the RTT values between the PM and $\mathcal{A}_{\mathcal{PM}}$ (RTT_{PM}), $\mathcal{A}_{\mathcal{PM}}$ and $\mathcal{A}_{\mathcal{C}}$ (RTT_{PMC}), which consists of RTT values between the network communication nodes that support the high speed connection, and $\mathcal{A}_{\mathcal{C}}$ and the Client (RTT_C). The communication relationship between said entities is visually depicted as:

$$PM \text{ — } \mathcal{A}_{\mathcal{PM}} \cdots \mathcal{A}_{\mathcal{C}} \text{ — } Client$$

$$RTT_T = RTT_{PM} + RTT_{PMC} + RTT_C$$

It is difficult to approximate RTT_{PM} and RTT_{PMC} because their values are significantly influenced by and dependent on many factors (e.g., communication nodes' connection medium, network traffic load, propagation delay, etc.). However, since the beacon transmission between $\mathcal{A}_{\mathcal{C}}$ and Client simulates the transmission between the PM and the Client as consequence of employing similar hardware and software, we are able to approximate RTT_C to 81ms based on our study. In addition, the triangle inequality theorem ensures us that $RTT_{PM} + RTT_{PMC} > RTT_C$ since the path from the PM to the Client is not a direct route. Thus, the RTT threshold (RTT_{TH}) should be set to 81ms for each legitimate proximity module to prevent relay attacks when the Client is outside the enterprise environment.

4.7.2 WiFi Manipulation

We leverage the WiFi signal interference caused by human activity to determine the number of occupants in a given room. A malicious user could attempt a DoS by disrupting WiFi signals. That is, an attacker could acquire a special device that would, for example, completely nullify WiFi signals [82–84]. Another means to circumvent the system would be to obstruct the LOS with something other than a human body such as a chair. Therefore, in either case, when the receiver processes the signal interference, it may publish an incorrect number of users within that room. However, the authorization server will detect violations because inconsistencies will exist within the PIP.

Regardless of whether Bluetooth or WiFi manipulation is employed, the scenarios that we address make it more difficult to circumvent CASSEC 2.0. That is, in both the SoD Scenario and the AOU scenario, multiple users with mutual interests must collude and agree in order to attempt bypassing the system.

4.7.3 True Continuous Authentication

Our passive biometric authentication scheme only provides continuous authentication while the Client smartphone device is within the user's pocket. It is possible that an authorized user, whom the Client had previously authenticated, simply removes the device from the pocket, and subsequently gives the device to an unauthorized user. Consequently, the device is unlocked and its content is accessible by the unauthorized user. Therefore, other biometric authentication must be used. While there are both active and passive biometric authentication solutions, passive solutions should be used to maximize usability as they would not require users to actively authenticate themselves. To protect against such an attack, other passive biometric techniques to continually authenticate while the user is holding the phone should be used. Some biometric features that could be analyzed and passively authenticated include timing of keystrokes, touchscreen behavior, face, retina, or iris [74,86,87]. In fact, passive fa-

cial recognition technology has been recently (Nov, 2017) integrated into the Apple's new flagship mobile device: iPhone X [88].

4.8 Conclusions

In this chapter, we propose a proximity-based context-aware access control mechanism that also incorporates constraints concerning the confidence about user and location information. Such constraints allow the system to make decisions based on the *degree of reliability* of extracted contextual information. We have integrated such mechanisms into CASSEC 2.0 and have conducted a feasibility study to show our approach is viable in practice. We have evaluated our confidence constructs and collected some data by implementing behavioral and physiological biometric authentication and extending the occupancy detection mechanism with a robust co-proximity authentication protocol that is resistant against relay attacks. However, occupants were required to stand in the line of sight for at least one second to detect human activity and subsequently the number of occupants within a room. CASSEC 2.0 also implicitly assumes that remote enterprise content is only accessible online and is never stored on the end-user's device. In order to continue productivity, the end-user may need to download enterprise content to the device. Therefore, an enforcement mechanism must exist on the end-user's device to secure the content. We address these concerns in the next chapter. We investigate other localization techniques with higher accuracy and more robust detection. In addition, we investigate client-side technology to isolate enterprise content from all other device content and enforce proximity-based access control policies on enterprise content.

5. A CONTEXT-BASED CONTAINERIZATION SYSTEM

Platform-level EMM systems enable enterprises to deploy and manage containers, which contain sensitive enterprise content, to end-users' devices. However, we argue that given the dynamic nature of EED scenarios as users are assumed to be mobile, EMM systems must also consider the context in which containers (i.e., personas) are employed. In this chapter, we present the design of our Context-Based, Multi-Enterprise Containerization (MERC) context-aware system (CAS) that utilizes our prototype PBS to influence the behavior of containers. The MERC architecture limits employees' accesses to work personas and enterprise content within them dynamically and passively through the enforcement of context-based constraints. The novelty of our system is twofold. First, our system is the first to apply both location-*and* proximity-based (hereinafter, context/ual) constraints to containers. Second, we utilize sound to determine a device's logical location. We observe that many proposed PBSs are radio-based, and objects of various size, shape, and material in the environment can obstruct the propagation path of radio signals, thus diminishing the accuracy. For example, the work by Bocca et al. [9] demonstrates the effect of human interference on the propagation path of radio signals. Instead, we leverage the unique characteristics of sound that make it suitable for supporting PBSs. First, unlike radio signals, sound is inherently localized as it cannot penetrate walls or propagate over long distances. Second, it does not require a line-of-sight, as with GNSS systems whose signal degrades if a device is within a building. Third, the audio frequency of sound can be shifted so that it is inaudible to the human ear (i.e., ultrasound). To detect the proximity of other users, we also leverage Bluetooth Low Energy (BLE) capabilities of mobile devices as standardized BLE protocols provide proximity ranging measurements. We implemented our custom Android operating system (OS), MERCOS, on Pixel C and Nexus 6P, which OEMs can readily incorporate into their

proprietary EMMs (e.g., Samsung KNOX [20]). Enterprises can also readily deploy our custom PBS using devices with microphones, speakers, and Bluetooth pervaded in existing mobile IT and building infrastructures [89]. Last, we extend Android Device Administration policies with proximity-based constraints to influence persona behavior.

The contributions of this chapter can be summarized as followed:

1. *Position-Based Containerization:* We propose a context-based containerization policy enforcement scheme to control EED devices based upon positional constraints. Different from existing containerization architectures, our approach is the first to introduce proximity-based constraints to Android containers.

2. *Position-Based Service:* We investigate the feasibility of a novel application of ultrasound to determine a user's location. We also evaluate proximity-based detection via BLE. Our results produced high accuracy, with 100% location detection accuracy and a maximum false negative rate of 4% in proximity detection accuracy while introducing a minor impact on battery life, thus demonstrating the feasibility and effectiveness of our solution.

3. *Secure Beacon Protocol:* We propose an acoustic-based protocol that addresses several significant challenges in supporting a multi-enterprise position-based architecture, including interoperability, privacy, and security.

This chapter is organized as follows. Section 5.1 introduces motivational position-based scenarios. We provide additional background information on PBSs in Section 5.2. Sections 5.3 and 5.4 introduce our approach followed by the implementation and technical details in Section 5.5. We next report our experimental results in Section 5.6. We analyze the security of our approach in Section 5.7. Section 5.8 concludes the chapter.

5.1 Motivating Scenarios

The MERC specifically targets enterprises that are currently using EMM systems to manage employees' devices, but require that such systems be context-aware. In terms of access control, such systems aim to secure access to sensitive content on employees' devices by adapting access authorizations to the current context *without* explicit user intervention. Below, we describe two scenarios motivating the need to incorporate context-aware capabilities within EMM systems.

Consider an enterprise setting in which two, but independent enterprise organizations exist. Each enterprise allows enterprise containers, containing sensitive content, to exist on employees' devices. Each employee, regardless of the enterprise, is assigned a role that reflects the privileges granted to that employee within the respective organization. *NekSec*, the first enterprise, is a network security consulting agency whose objective is to identify security vulnerabilities within a client's computer network infrastructure. *Banker*, the second enterprise, is a financial institution that provides online banking facilities to its customers. Two relevant roles within *NekSec* and *Banker* are (network) *Supervisor* and *Consultant*. With respect to accessing enterprise content, the role Supervisor grants an employee many privileges. The privileges assigned to the Consultant role vary depending on whether the consultant is an external or internal entity to the organization. However, we focus on the former in our paper, which is further discussed below.

Location-based Containerization Scenario. *An enterprise container belonging to NekSec is deployed to Alice's, a NekSec Consultant, smartphone. The container's content is highly sensitive as it contains confidential information regarding NekSec's clients. The enterprise requires that the container must only be accessible on NekSec's campus.*

The scenario reflects a real world circumstance in which an employee only has access to resources while the employee is on premise. For example, an employee would normally only be able to access his/her enterprise user account via *stationary* ter-

minals; the terminals do not leave the work premise. However, implications of EED must be considered. The dual use of mobile devices allows employees to remotely access resources that otherwise would not be accessible outside of the enterprise setting. Therefore, such circumstances require that containers (e.g., user accounts or personas) must only be accessible at specific locations.

Proximity-based Containerization Scenario. *Banker's network Supervisor has hired Alice to investigate the possible existence of insider threats and the leakage of confidential financial information through the institution's network-enabled computing devices. Similarly, to conduct her investigation, a Banker container with confidential network security data has been deployed to Alice's smartphone, but can only be accessed within a designated office on Banker's campus. In addition, only employees with the role of Supervisor or higher are authorized to be within Alice's immediate proximity while in the office.*

This scenario also reflects real world circumstances; in fact, 23% of enterprises make EED available to contractors [19]. Consultants are often hired to temporarily provide their expertise on an on-going project, but are only granted a limited set of privileges required to execute their duties. In addition, investigating the existence of nefarious activities executed by employees is of a sensitive nature; Banker should be extremely cautious not to alert low level employees that are concluded to have malicious intent. Consequently, such circumstances require that containers must only be accessible depending on proximity-based information.

5.2 Background

Besides BLE (Chapter 2), acoustic communication is also a possible technology for both geofencing and microlocation PBSs [90, 91]. By applying coding schemes, data can be transmitted and received through the air using acoustic hardware in mobile devices. Various coding schemes (e.g., on-off keying) have been proposed that encode data into sound by modulating sound wave properties, such as frequency, amplitude,

or phase which affect the bit rate, bit error rate, and range of transmission [90, 91]. As our work targets EED scenarios, we are particularly interested in encoding mechanisms that allow reliable acoustic communication using audio signals not perceivable to humans. That is, we embed location-based information in ultrasonic signals that operate at frequencies above 18 kHz, which is understood to be frequencies at which adult humans are unable to detect. Developing or determining the optimal encoding scheme that allows efficient transmission of ultrasonic signals is outside the scope of the paper. We utilize a third-party sound-based, data communication SDK [92] to embed and extract location information within ultrasonic signals. With such a SDK, we are able to transmit data at the speed of sound, which is approximately 340 m/s [26] at standard temperature and pressure.

5.3 Design Goals, Challenges, and Assumptions

The design of a multi-enterprise CAS that utilizes a PBS to influence the behavior of containers introduces several challenges:

Interoperability: We consider the dynamic nature of users in EED scenarios involving individuals using their devices for multiple enterprises. The CAS must therefore address the *occupancy detection problem* [18], that is, *who* is and/or *how many* people are in a given space. As such, we aim to make our position-based architecture and secure beacon protocol interoperable so to allow a device's secure container to be influenced even when the device owner moves from one enterprise environment to another.

Privacy: Another design goal is to ensure the confidentiality of enterprise building infrastructures. Some PBSs (e.g., iBeacons) transmit cleartext positional data, thereby divulging information particular to a given enterprise. Our system protects such information as enterprises may desire confidentiality.

Security: One design goal is to minimize the trust placed in users of the system. Specifically, we do not rely on users, possibly malicious, to manually report their

location or proximity to others. Instead, the system takes a proactive approach by automatically monitoring entities within the environment. In that way, we make access to personas secure and as fluid as possible.

Ease of Integration: Another design goal is to maximize ease of integration into enterprises' IT infrastructures. First, we must consider the method in which we incorporate context-based constraints into existing OEMs' containerization solutions. Second, we intentionally employ sensor technology already integrated into IT infrastructures, thereby removing deployment costs.

Flexibility: As each enterprise environment is unique, we also aim to make the specification of locations as flexible as possible.

Performance: Given that users are assumed to be mobile, continuity of access must be considered; CASs should be readily updated when context changes. For instance, a persona should be quickly activated/deactivated once an employee enters/leaves the workplace. As such, we aim to minimize and simplify the steps in the communication process between the architectural components while not impacting system performance.

5.3.1 Assumptions

We make the following assumptions about the proposed system and the adversary attempting to view content (i.e., information regarding the current investigation) on Alice's device. Each employee/contractor, including the adversary, has full access to his/her device. Each device has been preauthorized by the IT admin for EED use. Preauthorization consists of (1) deploying a work persona to the device in which the IT admin controls and (2) verifying that the device's acoustic and Bluetooth sensors are functioning correctly. Consequently, we assume IT admins can be trusted. We trust the Android access control system, which includes the Android middleware and Linux Kernel, to correctly enforce all security policies. Our ultrasonic beacon protocol requires the exchange of cryptographic keying material between MERC's

architectural components to protect communication. We assume that such material is secured. Physical security or video monitoring is employed to prevent the adversary from compromising positioning modules and entering the environment with foreign objects such as a non-secured phone/camera so as to configure a remote monitoring device within Alice's designated office room.

5.4 MERC System Architecture

In this section, we describe the MERC architecture. We first describe the architectural components for the sake of defining terms. Next, we provide an overview of the system.

5.4.1 Client-Server Architecture

Client: A Client is a device that is operated by a user to access enterprise content, and in our work, content includes persona, applications, and data. We focus on smartphones, but the same techniques are also applicable to desktops.

Enterprise Policy Server (EPS): The EPS component hosts policies and disseminates them to employees' devices when required. By designing this component as a server, a heterogeneous network of end-users' devices can be serviced. Therefore, access to resources can be requested from desktop terminals or mobile devices.

Positioning Module (PM): The role of the PM is to detect the positions (i.e., location and proximity) of Clients by periodically collecting and analyzing contextual information.

We note that the various components of MERC's architecture may not be integrated into the same physical component. In our proof-of-concept implementation, for example, the PM only provides location verification support by transmitting periodic ultrasonic beacons, which Clients consume. In addition, a Client consumes/emits BLE beacons to determine the proximity of other Clients. We depict the prototype's architecture in Figure 5.1.

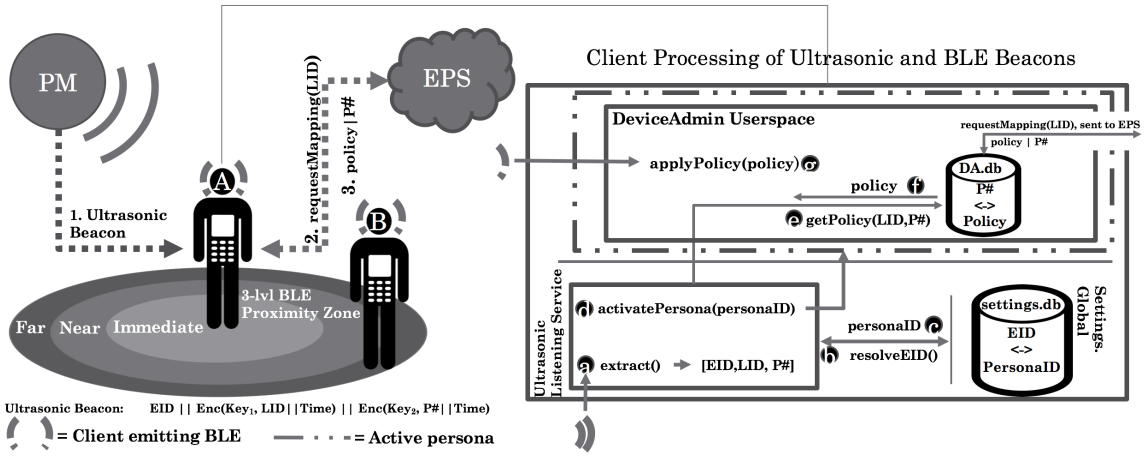


Fig. 5.1. Processing of beacons within MERC's architecture.

5.4.2 Overview

The operations of the MERC are centered around enterprises' mobile IT management. Each enterprise has enterprise members that deploy personas. We define an *enterprise member* as the entity which deploys, controls, and manages a persona. An enterprise member could be either the IT admin (of a business/corporation) or simply an end-user; we, however, focus only on the duties of the former because enterprises desire to have full control over employees' personas.

Client Registration. Each enterprise has an Enterprise ID (EID)[1] that is communicated to Clients. This EID serves two purposes: (1) as an unique identifier for an enterprise; (2) as information regarding the top-level locations that belong to the particular enterprise.

Persona Deployment. We leverage Android facilities to deploy personas on our platform. The persona deployment is achieved through the development of a custom device admin application built using Android's Device Administration APIs. Use of such APIs increases the MERCOS's ease of adoption as current IT admins

[1]The creation and management of unique EIDs are neither delegated to MERCOS or enterprises. AFW, through Google Enterprise solutions, handles such operations [93].

would have already built a device admin to control their employees' devices running AFW, which MERCOS is built upon. Once the device admin is downloaded to an employee's device, the device admin creates a new persona dedicated to work-related activities. At this point, a clean userspace containing the same list of applications as the default persona, including the device admin, will be instantiated. However, unlike the rest, the device admin is the only application that is removed completely from the default persona as its duties only lie in the created persona. Once the persona-creation process is completed, the device admin must register a legitimate EID, acting as a persona ID, via our custom interface.

Activating Personas & Deploying Policies. End-users authenticate their logical location by passively consuming, via Clients, an ultrasonic beacon that is partially encrypted with two different keys. The Ultrasonic Beacon (Figure 5.1) contains several pieces of information, including an EID, a Location ID (LID), timestamps, and a policy number. The ultrasonic beacons are periodically sent every 10 seconds by a PM. The EID and LID describe a Client's general (e.g., NekSec's campus) and specific (building 10, room 100) location, respectively.

We apply defense-in-depth by employing three layers of defense to access content on the Client. First, the Client extracts the EID from the beacon, and compares the identifier with previously registered EIDs. If there is a matching EID, the Client activates the associated persona, otherwise, the default persona is activated. We use the term "activate" to simply indicate that a persona is brought to the foreground, and thus, all other personas are not visible since they execute in the background. Unlike the first defense layer, the second and third layers are handled by the device admin. Android delegates the responsibility of screen-locking a persona to the device admin [31]. However, the device admin allows the screen lock to receive user input *only* if the policy number is decipherable. Such defense-in-depth ensures legitimate access authorizations, regardless of an attacker attempting a replay or denial-of-service attack (see Section 5.7).

At this point, the Client forwards the beacon's LID to the device admin, which is within the Client's active persona. As the device admin is built by the enterprise, it is aware of the EPS' remote address, and thus forwards the LID to the EPS for processing (requestMapping in Figure 5.1). The content of the LID dictates the content of the EPS' response message. Each LID is tied to a unique policy, and this association is configured by the enterprise member. The EPS responds with the policy number and the policy itself.

To minimize communication between the Client and the EPS, the device admin stores the policy number sent by the EPS. Whenever the device admin extracts a policy number from the ultrasonic beacon, the device admin compares it with previously stored values. If a value previously exists, the matching policy is adhered to, otherwise, the device admin forwards the LID to the EPS to retrieve the appropriate policy.

Encryption Keys. As previously stated, ultrasonic beacons are partially encrypted using two keys (Figure 5.1): Key_1 is a periodically updated symmetric key generated by and stored on the EPS. It is used to encrypt a LID and a timestamp. Such encryption ensures the confidentiality of an enterprise's building infrastructure as per our design goal. Key_2 is the private key component of an asymmetric pair generated by the EPS. The private key is used to encrypt a policy number and a timestamp. The public component, which is stored within an X.509 certificate downloaded to a Client device, is used for decryption. Such encryption minimizes the communication between the Client and EPS while maintaining high security since the policy number is used to apply an associated policy previously downloaded to the Client, and therefore obviating the need to contact the EPS as frequently as ultrasonic beacons are broadcasted.

Detecting Proximity. The policies a system can enforce are dependent on the underlying technology. Similarly to ultrasonic beacons, each Client scans for BLE beacons every 10 seconds. Particularly, we utilize BLE to detect if other users are within *Near* reach (c.f. to Section 5.2). That is, the proximity zone (Figure 5.1)

encompasses the area within 3 meters around a Client. We selected *Near* for the maximum range as we believe that if, for example, any employee with an inferior role to that of the Supervisor is within 3 meters of Alice, then it is a clear indication that the employee is able to visually observe content on Alice's device he/she is unauthorized to view. We emphasize that this hard-coded metric of proximity is implementation specific as we rely on BLE's four-step ranging measurements. We assume that the device admin maintains a database that maps Bluetooth MAC addresses to user IDs, which can be easily retrieved from an enterprise's preexisting RBAC system that is incorporated into the EPS. We chose the Client (rather than the EPS) to track user movements so that Clients can immediately react to proximity violations and minimize communication with the EPS.

5.4.3 Proximity-Based Device Admin Policies

Much work has been done in the design of policy languages [17, 33, 41, 77]. However, we extend CASSEC's PrBAC policy specification [17] and integrate it into Android's existing Device Administration policy specification (MercBAC), thereby enabling MERC to enforce restrictions based on Separation of Duty and Absence of Other Users. The device admin is solely responsible for enforcing policies on the persona such as password configuration or device force lock, and similar to Android's permission model, the device admin must request the privilege to exercise such capabilities through a security metadata file stored within the application's binary. We do not modify this file. Instead, the device admin reads a MercBAC policy, written by the IT admin, which is stored within the application's data directory. The device admin must ensure the current context violates the policy's contextual constraints prior to exercising force lock (*DevicePolicyManager.lockNow()*), for example, thereby applying proximity-based access control to personas. In this way, we increase the ease-of-adoption and resiliency to change in MERCOS as a subset of these policy

features are pre-built into the stock Android OS. Figure 5.2 provides an example MercBAC device admin policy that reflects the scenario presented in Section 5.1.

```
<device-admin
xmlns:android="http://schemas.android.com/apk/res/android">
   <uses-policies>
        <force-lock >
         <context>
            <obligation>ongoing</obligation>
            <proximity-constraints>
               <proximity-constraint>
                  <cardinality>at_least</cardinality>
                  <digit>1</digit>
                  <role>inferior(Supervisor)</role>
               </proximity-constraint>
               ...
</device-admin>
```

Fig. 5.2. Example proximity-based MercBAC policy.

5.4.4 The EID and LID

We take advantage of preexisting data structures for EIDs. An IT admin is required to perform a registration process with Google which entails claiming and verifying an enterprise domain name (e.g., www.example.com) to use Android For Work (AFW). In addition, an Eddystone UID Namespace is generated by selecting the first 10 bytes of a SHA-1 hash of the domain name. The MERC uses the Eddystone UID as the EID to describe top-level locations.

The intentional naming system proposed by Adjie-Winoto et al. [94] is an attribute-value naming system with nested attribute-value pairs. Such a naming architecture provides significant flexibility in defining location-based information of broad resolutions. As such, we adopt this attribute-value system to construct a LID. The advantage is that the particular construction of a LID is defined by each enterprise IT admin instead of uniformly. Here, we use a simple construction of the following form: [building = B [floor = F [room = R]]].

5.5 Prototype Implementation

5.5.1 Client

We made minor modifications to the Android OS to support MERCOS. We first created and exposed a new system API called *switchPersona(string eid)*. This API switches the persona, and is called whenever a new EID is detected from an ultrasonic beacon. For simplicity, we allow device admins to register an EID within Android's "settings.db", which is a database managed by the Settings application, via the *Settings.Global* interface. In this way, the database can be globally accessed no matter which persona is currently active. Normally, third-party applications can only read from Settings.Global, and not write. To ensure that the device admins are the only entities with the write privilege, we call Android's *DevicePolicyManager.getActiveAdmins()* function. It returns a package name list of all device admin applications, but only *one* should exist as the enterprise controls the list of applications that exist within the work persona. Prior to updating the EID database, we verify that the package name of the entity attempting to update the EIDs is authorized to do so. Read access to EIDs is not a privacy concern since they are constructed based on enterprises' public web domain [31].

To send/receive ultrasonic beacons, we integrate a third-party sound-based, data communication SDK [92] into our custom device admin which operates at frequencies above 18 KHz (i.e., frequency range inaudible to the human ear). The reader may wonder why we integrate such a feature at the application level rather than the system level. We believe that this is sufficient as modern smartphones already silently process audio in the background via application-level programs. For example, the Google Now application allows the Android OS to respond to voice-commands [95]. So in reality, the ability to read ultrasonic beacons would be integrated into the Google Now application so that it can be a system-wide functionality. The device admin also periodically scans for BLE beacons every 10 seconds using the *android.bluetooth.le*

APIs [31]. Nearby users are identified by maintaining a SQL database which contains a mapping from BLE MAC addresses to user IDs.

5.5.2 EPS

The EPS was implemented in PHP and hosted on a remote commercial server. It disseminates policy files to Clients. The EPS provides a function that can be remotely invoked via URL: *getPolicy(NetMsg)* (Figure 5.1). The function is invoked by Clients whenever a new policy number is detected in ultrasonic beacons.

5.6 Experimental Results

5.6.1 Deployment

In this section, we report experimental results. First, we deployed our MERC prototype in one of our campus buildings. Figure 5.3 displays the schematics of our tested area. The green (benign) and red (malicious) circles and arrows indicate the placements and facing directions of PMs, respectively. A large, grid-patterned rectangle points out a sub-area of an enterprise environment that contains only one PM. The gray-filled circles indicate the current positions of Clients. The gray, circular outlines indicate the possible positions of Clients during testing. Second, our modifications to the Android source code were tested on the Android Pixel C tablet device running Android Nougat (API 22 v7.0). Last, each PM was a Dell A215 Multimedia Speaker, and each speaker was connected to a device capable of playing MP3s. All experiments were conducted in areas in which the ambient noise were minimal.

5.6.2 Experiments

Experiment 1: Enterprise Setting Suitability. An enterprise may place PMs in arbitrary locations such as an office or a large sitting area (e.g., auditorium). Therefore, it is necessary to understand how messages embedded in ultrasounds will

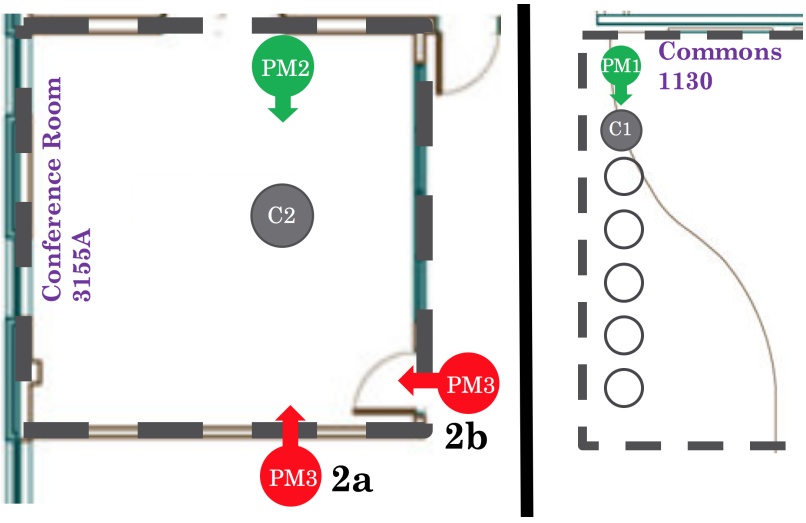


Fig. 5.3. Testing area which contains our positioning module (PM#) and Clients (C#). Arrows indicate the directions PMs are facing.

propagate. Specifically, the goal of the experiment is to determine if the Client is able to capture location information at varying distances away from a PM. The Client (*C1*) was placed at six different positions away from the PM (*PM1*), and at each position, *PM1* transmitted 10 ultrasonic beacons at its maximum possible amplitude.

Figure 5.4 shows results of this experiment. *C1* was able to detect beacons with strong accuracy, at least 90%, up to 30m away from *PM1*. However, at a distance of 36m, *C1* was only able to detect beacons with 60% accuracy. The lower detection rate at 36m was expected as it is a natural phenomena that everyone observes on a daily basis. That is, there is a direct correlation between the distance between a source of sound and a listener and the likelihood of the sound being heard. Therefore, a speaker that can emit sounds at larger volumes would be able to transmit beacons to Clients at farther distances. Nevertheless, this experiment has demonstrated that ultrasonic beacons can be detected with 100% accuracy in most enterprise settings since such settings (i.e., offices and meetings rooms) are significantly smaller than 24m on the longest sides [89].

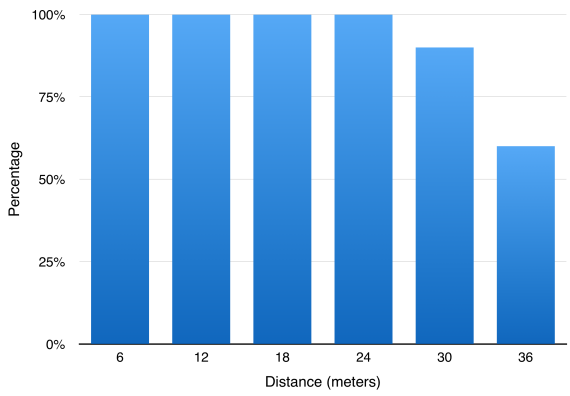


Fig. 5.4. Capturing location information at varying distances.

Experiment 2a and 2b: Collisions. Sound waves can transmit arbitrarily far, and sounds from varying sources can mesh together. If multiple PMs are placed in relatively close proximity, the ultrasounds may also blend together. We must determine if placing PMs in isolated areas that are in proximity, but demarcated by walls or closed doors will cause any interference with Clients. We explain in the security analysis below that an adversary is unable to transmit *valid* ultrasonic beacons. However, we temporarily relax our video monitoring assumption (Section 5.1), thereby enabling an attacker to transmit beacons on the same ultrasonic frequency to cause collisions via malicious PMs (red PMs in Figure 5.3) from adjacent rooms/areas. We perform two experiments to determine the extent in which an adversary can perform a Denial-of-Service attack with the constraint that the adversary is using the same hardware deployment acquired by hijacking legitimate PMs. In Experiment 2a, *C2* and *PM2* are placed within a closed-door room situated roughly 3 and 6 meters away from *PM3*, respectively. *PM3* is pressed against and facing a 1 ft (∼0.3 m) thick wall. In Experiment 2b, *C2* and *PM2* remain in the same positions, but *PM3* is now pressed against and facing the room's door. A notable difference between the two experiments is that, although still demarcated by some obstruction, *PM3* may have a likelier chance to permeate through the room as cracks exist around the door that sound can travel through.

Table 5.1.
Location detection rates

Experiment	**PM2**	**PM3**
2a	100%	0%
2b	100%	6%
2b′	77%	57%

In each experiment (Table 5.1), each PM begins transmission, at maximum volume, of 30 ultrasonic beacons at a specified time using a time-based activation program. In Experiment 2a, *C2* did not detect beacon collisions as it identified 100% of *PM2*'s beacons and 0% of *PM3*'s beacons. Such results demonstrate that sound is indeed inherently localized as the attacker could not successfully penetrate the obstructing wall(s). In Experiment 2b, *C2* detected beacon collisions as it identified 100% and 6% of *PM2*'s and *PM3*'s beacons, respectively. We performed Experiment 2b once more (i.e., 2b′), but we instead increase the adversary's attack power by leaving *PM3* at full volume while reducing *PM2*'s volume by half. As a result, *C2* identified 77% and 57% of *PM2*'s and *PM3*'s beacons, respectively. Such results demonstrate that under a certain adversarial model, an attacker can cause collisions. Given the unprecedentedly fine-grained nature of EED scenarios that we envision for MERC (e.g., different LID per room), processing beacons from adjacent rooms/areas would cause the Client to continuously switch containers or apply the wrong policies. Such erratic behavior is a major issue w.r.t. security, and it would also potentially ruin the user experience. To address this issue, we implemented a temporal localization analysis mechanism to determine the correct candidate to enforce in a set of beacons recently heard by a Client. We first determine which beacon is consumed more frequently, but using only this criteria is insufficient as an attacker could simply increase the rate of transmission of malicious beacons. Therefore, beacons must also

Table 5.2.
Proximity detection rates of two stationary BLE devices

Distance	**Rm1**	**Rm2**	**Rm3**
2m	FNR: 0%	FNR: 0%	FNR: 4%
4m	FPR: 2%	FPR: 0%	FPR: 0%

be consumed at a valid transmission rate, otherwise the attack is detectable. We discuss this further in Section 5.7.

Experiment 3: Proximity Detection. We test the proximity detection method which relies on BLE beacons. The device admin is configured to enforce the MercBAC policy in Figure 5.2. In particular, we test if Alice's Client can accurately determine the distance to the unauthorized user. Two BLE-enabled devices were used to conduct the experiment: a Nexus 6P smartphone (*C1*) and a Pixel C tablet (*C2*) acting as Alice's and the unauthorized user's Clients, respectively. We repeated this experiment twice; differentiating the two by placement of the stationary Clients: a distance of 2m ($<$ *Near*) and 4m ($>=$ *Near*) between each other which indicates attack and non-attack instances, respectively. We use the following metric to evaluate the effectiveness of the proximity detection. False negative rate (FNR) is defined as the percentage of attack instances in which *C1* mistakenly evaluates as non-attack instances. False positive rate (FPR) is defined as the percentage of non-attack instances in which *C1* mistakenly evaluates *C2* as attack instances. We performed the experiment three times, each on the 1st, 2nd, and third (Figure 5.3) floor of an isolated room. *C2* emitted 100 BLE beacons with each five seconds apart.

Table 5.2 presents the false positive rates and false negative rates of proximity detection under varying distances. *C1* precisely evaluated *C2* as *Near* when *C2* was placed 2m away, with a 4% FNR in the worst case. When Clients were situated 4m apart, a FPR of 2% was observed in the worst case. Such occurrences can be attributed to possible interferences caused by the environment since BLE is

a radio-based technology, and such technology is susceptible to signal attenuation. The experiment has demonstrated that if unauthorized employees enter Alice's vicinity, with high accuracy, the device admin would be able to force lock the persona. However, an enterprise may consider a 4% FNR non-negligible as personas would be inaccessible in such instances. We leave the investigation of alternative proximity technologies for future work.

Experiment 4: Battery Consumption. Mobile devices are resource constrained, and continually probing sensors can tax the device. The Clients are continually listening for ultrasonic and BLE beacons. The goal of this experiment is to observe the consumption of the device's battery. We monitored the device's battery percentage when running both the unmodified OS and our customized system, separately. We performed this experiment three times on each system. Towards this goal, we set *WindowManager* class's FLAG_KEEP_SCREEN_ON which is an Android mechanism to force the screen to never turn off. It is vital that this flag is set as it ensures that the listening service is not temporarily halted or shutdown by the stock Android resource management system. We logged the battery consumption every hour.

Figure 5.5 shows results of this experiment. As observed from the graph, the average performance impact of MERCOS is minimal as compared to the non-modified OS. The maximum difference observed each hour was 2%. An explanation for this result is that Android already silently processes audio in the background when the unmodified OS is used (e.g., Google Now application's voice-activation services). The processing of ultrasonic beacon in MERCOS takes precedent over the voice-activation services. Thus, the only additional processing that is performed in our custom OS is the scanning of BLE beacons. Therefore, integrating features of our CAS into resource-constrained devices is practical.

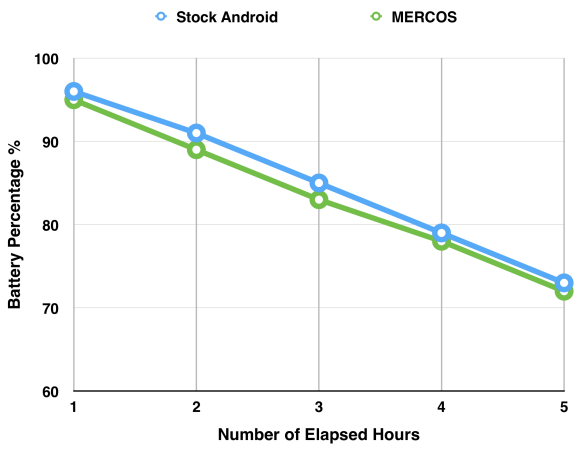


Fig. 5.5. Average battery consumption of a Client.

5.7 Security Analysis

We discuss possible attacks to our system, and means to to prevent or mitigate them.

5.7.1 Attacking Ultrasonic Beacons

Replay Attack. An attacker may attempt a simple replay attack. The goal of the attack is to confuse the MERC system to activate an incorrect persona and policy on a Client. The attack is executed by recording a previously transmitted ultrasonic message and re-transmitting it at a different time or location. We protect the system from replay attacks by embedding temporal information within the beacon. The components in the system must have loosely synchronized clocks. We extract the timestamps and compare them to the current time, and then determine if the time difference exceeds a specified threshold. In the midst of Experiment 1, at a distance of 24m, prior to signal degradation at 30m, we averaged the elapsed time in milliseconds to transmit, receive, and process 30 ultrasonic beacons. The longest sides of most offices or meeting rooms are significantly smaller than 24m, which we believe reflects the maximum distance an ultrasonic beacon must travel. On average, the complete

process took approximately 1.33 seconds. Although the longest elapsed time recorded was 2.5 seconds, all other recorded times fell well below two seconds. As a result, we set our threshold to two seconds. Therefore, the system would be able to detect malicious activity under the threat of a simple attacker.

In this chapter, we do not consider a sophisticated attacker that is able to execute a wormhole attack [96]. A wormhole attack is similar to a replay attack except that the adversary tunnels the beacon through a "wormhole". Defending against such an attack is quite challenging as the wormhole allows the re-transmission at a different location with minimal delay, possibly within milliseconds. A sophisticated method to address both attacks would be to employ ultrasonic distance-bounding techniques [26, 97]. Such techniques were not investigated for several reasons. First, recently proposed techniques require special hardware, and modern smartphones are not currently capable of handling such a task. Second, given the scenarios that we envision for MERC, such a feature would incur significant overhead for each architectural component. For these reasons, MERC would not satisfy the main goals of the paper as ease of integration and interoperability would be significantly reduced.

Denial-of-Service. In Experiment 2, we addressed the issue of collisions as result of *benign* position modules in adjacent rooms. However, an attacker can execute a denial-of-service (DOS) attack by physically tampering with or altering the beacons that are transmitted from PMs. For example, an attacker may tamper with a benign PM to transmit audio shifted to a frequency that will negate legitimate ultrasonic beacons. If this occurs, the Clients may consume data that is corrupted and indecipherable; persona activation and policy deployment will not function properly. It is difficult to defend against such a DOS attack, but it can be detected. The results in Experiment 1 and 2 demonstrate high accuracy with respect to detecting beacons. Consequently, repetitious consumption of indecipherable beacons can be inferred as malicious activity, especially if enterprises appropriately place benign PMs as to minimize collisions between those devices. Another practical method that can be immediately employed is Android's Geofences APIs [31]. A geofence is a circular

area defined by a latitude, longitude, and radius, which can be specified by the device admin. The device admin thus becomes more context aware as it is alerted whenever a user enters/exits the geofence. Experiencing GPS signal attenuation is not a cause for concern since the device admin would instead place a geofence to entirely encompass, for example, NekSec's campus. An expected EID is thus established once a Alice enters the campus, otherwise, a DOS attack can be inferred. Last, physical security could also be utilized by delegating the responsibility of monitoring for malicious location devices to sentries placed throughout the campus. Nevertheless, the security of personas would still be ensured if enterprises apply defense-in-depth as described in Section 5.4.2.

5.7.2 Attacking BLE Beacons

Rooting. A possible attack to our system involves a user rooting his/her Client, and maliciously modifying the Bluetooth MAC address. In this way, the Client can impersonate another user of the system. To mitigate such an attack, an enterprise must employ hardware and software mechanisms (e.g., Android's dm-verity) that enhance device security. For example, Samsung KNOX 2.0 is a custom Android OS which has a low-level security feature that leaves the device inoperable once it detects a root attack [20], which is a sufficient mechanism to defend against malicious modification of the MAC address (or any root-based attack targeting MERC).

Masking BLE. The simplest attack malicious users could execute is to mask their MAC address by either disabling the Bluetooth peripheral on the Client or simply leaving it in another room. By doing so Clients will be unable to correctly enforce proximity constraints. There are several measures that can be taken against such attack. First, an enterprise simply has to enforce mandatory enabling of Bluetooth through the device admin application, which completely controls the settings and configurations of the Clients. Second, accidental or malicious misplacing of the Client can be addressed by supplementing the system with a facility to detect the number

of individuals in a room. For example, the system by Oluwatimi et al. [17] takes an infrastructure-based approach (i.e., independent of the Clients) utilizing signal attenuation of WiFi radios caused by human interference to achieve occupant localization. Therefore, if the number of individuals in a room and the number of MAC addresses do not match, the system will infer malicious behavior, and subsequently revoke access to resources.

Unauthorized Device User. Another attack vector involves an unauthorized individual obtaining an authorized user's phone, whether by theft or voluntary provision. Such an attack would allow an individual to gain unauthorized access to persona content. To mitigate such a threat, biometric techniques can be employed. For example, Draffin et al. [98] demonstrate that it is possible to passively detect that a mobile device is being used by a non-authorized user by modeling user keyboard interactions. Wang et al. [99] explore biometric signatures using WiFi-based techniques to determine the identity of an individual. Using such techniques, it is therefore possible to associate an individual with a device.

5.8 Conclusion

In this chapter, we investigate the feasibility of introducing context-based constraints to containers under a multi-enterprise context. Contextual information is supplied by our prototype PBS that relies on ultrasonic[2] and Bluetooth Low Energy beacons to address occupancy detection. With such information, proximity-based constraints can be effectively and efficiently enforced on Android's personas, which, to the best of our knowledge, has never been investigated. We also demonstrate how to allow multiple context-aware systems from different enterprises to serve a fleet of devices while maintaining privacy, security, scalability, and interoperability. Serving devices in such a manner is accomplished via our secure ultrasonic beacon protocol.

[2]Careful construction and transmission of ultrasonic signals must be taken to prevent adverse effects on the human body [100].

6. CONCLUSION

In this dissertation, we investigated various aspects of context-aware systems that is applicable to mobile systems technology within enterprise environments. We specifically asked how do we capture contextual information, incorporate contextual constraints into access control policies, and enforce contextual access control policies? To those ends, we first proposed a modified version of the Android operating system called CBAC (Chapter 3) that enables end-users to configure context-based policies for mobile applications. The context-based policies were used in the decision process by our custom OS to determine whether a mobile application, possibly malicious, has access to content (i.e., data and resources) that exists on the device depending on the context in which the content was requested. In this system, contextual constraints consisted of spatio-temporal parameters (i.e., location and time) which included the user being able to specify logical locations. Location-based information was captured via the end-user's device using the WiFi peripheral. We specifically triangulated the user's position by analyzing the surrounding WiFi access points and their RSSI levels.

While the security of enterprise content may depend on the presence of others, we also proposed a proximity-based mobile architecture. The proposed architecture CASSEC 2.0 (Chapter 4) supports context-based access control decisions based on the location of the user requesting access and the proximity of other users in a monitored area, so that the appropriate privileges to access remote enterprise content is automatically granted. In addition, we extend the PrBAC model to incorporate confidence constructs that would allow the system's access control decisions to be influenced by the degree of reliability of extracted contextual information. We evaluate our confidence constructs by implementing two new authentication mechanisms. Co-proximity authentication employs our time-based challenge-response protocol, which leverages Bluetooth Low Energy beacons as its underlying occupancy detection tech-

nology. Biometric authentication relies on the accelerometer and fingerprint sensors to measure behavioral and physiological user features to prevent unauthorized users from using an authorized user's device.

	Administrator	Content Location	Context-based Policies	Technology
CBAC	End-user	End-user Device	Location Time	WiFi
CASSEC	Enterprise	Remote Server	Location Proximity User	WiFi BLE Accelerometer
MERC	Enterprise	End-user Device	Location Proximity	Sound BLE

Fig. 6.1. High-level characteristics of each context-aware system.

We also proposed the MERC (Chapter 5) system which addresses the issue of applying proximity-based constraints to the management of end-user devices via mobile containerization techniques and technologies when end-users may be employed by or consult for multiple enterprises. As enterprise content may be downloaded to end-users' devices, MERC, our custom Android OS, ensures that the content is isolated from non-enterprise related content via platform-level containerization (i.e., personas). MERC then applies location-based and proximity-based constraints to the secure containers and their content. Contextual information is captured via a device's microphone and Bluetooth peripheral. Our novel approach uses ultrasonic sound that is inaudible to the human ear to determine a user's location. We then use Bluetooth Low Energy technology to determine proximity of other users. We demonstrate how to integrate context-based constraints into Android Device Administration policies in order for an enterprise to restrict employees' access to containers' content based on the proximity of other employees. Distinguishing characteristics of each of the previously mentioned systems are highlighted in Figure 6.1.

In the future, we will investigate how to enhance system security by incorporating more contextual information. Experiment 2 in Section 5.6 demonstrates that adjusting the amplitude of ultrasonic beacons affects the transmission distance. We can potentially exploit phenomenon such as this to support micro-position access control policies. For example, in a large sitting area containing tens of employees, a single high-level employee may instead use his smartphone device as a positioning module to support micro-position transmission of ultrasonic beacons. In this way, the high-level employee may adjust the device's volume to minimal levels to impose temporary restrictions on only low-level employees' devices that are within a few feet. One limitation of the proposed proximity-based systems presented in this work is the inability to deduce potential shoulder-surfing attacks. Shoulder-surfing attacks exemplify proximity-based situations in which an authorized user accessing sensitive content via his mobile device is unaware of an unauthorized user in the immediate vicinity, thereby leaving the authorized user's sensitive content vulnerable to potential information leakage. The most interesting aspect of this constructed scenario is that current proximity-based access control systems, including ones presented in this dissertation, would apply access control constraints regardless of whether the attack is possible. The unauthorized user may be situated *in front* of the authorized user, outside of the device's front screen peripheral. We intend to also develop the expressiveness of proximity-based access control policies and rely on micro-positioning technologies to resolve such contentious access control decisions.

Amongst the proposed systems, techniques, technologies, and context-based policies, which is the best strategy to employ? There is no *one-size-fit-all* approach; a specific solution may be only applicable to a specific problem or scenario. An enterprise that desires to integrate a context-aware access control system into its mobile IT infrastructure must first answer several questions, which include which content must be protected, who or what must the content be protected from (e.g., constructing the adversarial model), and what it means for the content to be secured (e.g., sustaining desirable properties under intelligent adversaries)? Once such questions

are answered, the enterprise must determine the set of contextual information within the physical or computing realm that can be leveraged to provide more dynamic and robust access control mechanisms, while minimizing the cost of deploying the technologies required to implement such security facilities. This dissertation examines various security requirements and real-world context-based problem scenarios and subsequently proposes several example context-aware access control solutions to address those scenarios.

REFERENCES

REFERENCES

[1] A. K. Dey, "Providing architectural support for building context-aware applications," PhD dissertation, Georgia Institute of Technology, 2000.

[2] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[3] R. Technologies, "Use cases," http://www.redwall.us/resources/Downloads/Redwall-Mobile-Use-Cases.pdf.

[4] I. M. Santos, "Use of students personal mobile devices in the classroom: Overview of key challenges," in *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2013*, T. Bastiaens and G. Marks, Eds. Association for the Advancement of Computing in Education (AACE), October 2013, pp. 1585–1590.

[5] Y. Song, "Bring your own device (BYOD) for seamless science inquiry in a primary school," *Computers & Education*, vol. 74, pp. 50–60, 2014.

[6] B. Shebaro, O. Oluwatimi, D. Midi, and E. Bertino, "Identidroid: Android can finally wear its anonymous suit," *Transactions on Data Privacy 7*, 2014.

[7] B. Shebaro, O. Oluwatimi, and E. Bertino, "Context-based access control systems for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 2, pp. 150–163, 2015.

[8] "Project proposal," http://hfid.olin.edu/sa2013/s_engr3220-unibros/proposal.php.

[9] M. Bocca, O. Kaltiokallio, and N. Patwari, "Radio tomographic imaging for ambient assisted living," in *Evaluating AAL Systems Through Competitive Benchmarking*. Springer, 2012, pp. 108–130.

[10] W. Liu, X. Li, and D. Huang, "A survey on context awareness," in *2011 International Conference on Computer Science and Service System (CSSS)*. IEEE, 2011, pp. 144–147.

[11] L. L. N. Laboratory, "Controlled items that are prohibited on LLNL property," https://www.llnl.gov/about/controlleditems.html.

[12] M. Conti, V. T. N. Nguyen, and B. Crispo, "Crepe: Context-related policy enforcement for Android," in *Proceedings of the 13th International Conference on Information Security*, ser. ISC'10. Springer-Verlag, 2011, pp. 331–345.

[13] A. Kushwaha and V. Kushwaha, "Location based services using Android mobile operating system," *International Journal of Advances in Engineering and Technology*, vol. 1, no. 1, pp. 14–20, 2011.

[14] S. Kumar, M. A. Qadeer, and A. Gupta, "Location based services using Android," in *Proceedings of the 3rd IEEE International Conference on Internet Multimedia Services Architecture and applications*, ser. IMSAA'09, 2009, pp. 335–339.

[15] M. S. Kirkpatrick and E. Bertino, "Enforcing spatial constraints for mobile RBAC systems," in *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*. ACM, 2010, pp. 99–108.

[16] A. Gupta, M. Miettinen, N. Asokan, and M. Nagy, "Intuitive security policy configuration in mobile devices using context profiling," in *IEEE International Conference on Social Computing*, ser. SOCIALCOM-PASSAT '12. IEEE Computer Society, 2012, pp. 471–480.

[17] O. Oluwatimi, D. Midi, and E. Bertino, "A context-aware system to secure enterprise content," in *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*. ACM, 2016, pp. 63–72.

[18] S. K. Ghai, L. V. Thanayankizil, D. P. Seetharam, and D. Chakraborty, "Occupancy detection in commercial buildings using opportunistic context sources," in *2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2012, pp. 463–466.

[19] H. Schulze, "BYOD & mobile security 2016 spotlight report," http://crowdresearchpartners.com/wp-content/uploads/2016/03/BYOD-and-Mobile-Security-Report-2016.pdf, March 2016.

[20] O. Oluwatimi, D. Midi, and E. Bertino, "Overview of mobile containerization approaches and open research directions," *IEEE Security & Privacy*, vol. 15, no. 1, pp. 22–31, 2017.

[21] N. Elenkov, *Android Security Internals: An In-depth Guide to Android's Security Architecture*. No Starch Press, 2014.

[22] M. Vossiek, L. Wiebking, P. Gulden, J. Wieghardt, C. Hoffmann, and P. Heide, "Wireless local positioning," *Microwave Magazine, IEEE*, vol. 4, no. 4, pp. 77–86, 2003.

[23] F. Zafari, I. Papapanagiotou, and K. Christidis, "Microlocation for internet-of-things-equipped smart buildings," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 96–112, 2016.

[24] A. Larchikov, S. Panasenko, A. V. Pimenov, and P. Timofeev, "Combining RFID-based physical access control systems with digital signature systems to increase their security," in *Software, Telecommunications and Computer Networks (SoftCOM), 2014 22nd International Conference*. IEEE, 2014, pp. 100–103.

[25] M. Moreno, J. L. Hernandez, and A. F. Skarmeta, "A new location-aware authorization mechanism for indoor environments," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference*. IEEE, 2014, pp. 791–796.

[26] K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun, "Proximity-based access control for implantable medical devices," in *Proceedings of the 16th ACM Conference on Computer and Communications security*. ACM, 2009, pp. 410–419.

[27] R. Bruno and F. Delmastro, "Design and analysis of a bluetooth-based indoor localization system," in *IFIP International Conference on Personal Wireless Communications*. Springer, 2003, pp. 711–725.

[28] Y. Jiang, X. Pan, K. Li, Q. Lv, R. P. Dick, M. Hannigan, and L. Shang, "Ariel: Automatic wi-fi based room fingerprinting for indoor localization," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 441–450.

[29] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal, "Sentinel: Occupancy based HVAC actuation using existing wifi infrastructure within commercial buildings," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2013, p. 17.

[30] T. Saelim, P. Chumchu, and T. Mayteevarunyoo, "Design and performance evaluation of novel location-based access control algorithm using IEEE 802.11 r," *Journal of Convergence Information Technology*, vol. 10, no. 4, p. 33, 2015.

[31] Android, "Android developer's guide," http://developer.Android.com.

[32] D. Ferraiolo, D. R. Kuhn, and R. Chandramouli, *Role-based Access Control*. Artech House, 2003.

[33] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca, "GEO-RBAC: a spatially aware RBAC," in *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies*. ACM, 2005, pp. 29–37.

[34] S. M. Chandran and J. B. Joshi, "LoT-RBAC: a location and time-based RBAC model," in *Web Information Systems Engineering–WISE 2005*. Springer, 2005, pp. 361–375.

[35] S. Aich, S. Sural, and A. K. Majumdar, "STARBAC: Spatiotemporal role based access control," in *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*. Springer, 2007, pp. 1567–1582.

[36] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, and G. D. Abowd, "Securing context-aware applications using environment roles," in *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '01. ACM, 2001, pp. 10–20.

[37] G. Zhang and M. Parashar, "Context-aware dynamic access control for pervasive applications," in *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004, pp. 21–30.

[38] D. Kulkarni and A. Tripathi, "Context-aware role-based access control in pervasive computing systems," in *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '08. ACM, 2008, pp. 113–122.

[39] R. Sandhu, K. Ranganathan, and X. Zhang, "Secure information sharing enabled by trusted computing and PEI models," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications security*. ACM, 2006, pp. 2–12.

[40] S. K. Gupta, T. Mukheriee, K. Venkatasubramanian, and T. Taylor, "Proximity based access control in smart-emergency departments," in *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. 4th Annual IEEE International Conference*. IEEE, 2006, pp. 5–pp.

[41] M. S. Kirkpatrick, M. L. Damiani, and E. Bertino, "Prox-RBAC: A proximity-based spatially aware RBAC," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2011, pp. 339–348.

[42] J. Park and R. Sandhu, "The UCON ABC usage control model," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 1, pp. 128–174, 2004.

[43] A. Gupta, M. S. Kirkpatrick, and E. Bertino, "A formal proximity model for RBAC systems," *Computers & Security*, vol. 41, pp. 52–67, 2014.

[44] C. Stach and B. Mitschang, "Privacy management for mobile platforms–a review of concepts and approaches," in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference*, vol. 1. IEEE, 2013, pp. 305–313.

[45] M. Nauman, S. Khan, and X. Zhang, "Apex: Extending Android permission model and enforcement with user-defined runtime constraints," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 328–332.

[46] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, "These aren't the droids you're looking for: Retrofitting Android to protect data from imperious applications," in *Proceedings of the 18th ACM conference on Computer and Communications Security*. ACM, 2011, pp. 639–652.

[47] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming information-stealing smartphone applications (on Android)," in *International Conference on Trust and Trustworthy Computing*. Springer, 2011, pp. 93–107.

[48] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, "Mockdroid: Trading privacy for application functionality on smartphones," in *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*. ACM, 2011, pp. 49–54.

[49] G. Russello, B. Crispo, E. Fernandes, and Y. Zhauniarovich, "Yaase: Yet another Android security extension," in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE 3rd International Conference on Social Computing (SocialCom)*, 2011, pp. 1033–1040.

[50] S. Bugiel, S. Heuser, and A.-R. Sadeghi, "Flexible and fine-grained mandatory access control on Android for diverse security and privacy policies," in *The 22nd USENIX Security Symposium (USENIX Security 13)*, 2013, pp. 131–146.

[51] G. Russello, M. Conti, B. Crispo, and E. Fernandes, "MOSES: Supporting operation modes on smartphones," in *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*. ACM, 2012, pp. 3–12.

[52] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically rich application-centric security in Android," *Security and Communication Networks*, vol. 5, no. 6, pp. 658–673, 2012.

[53] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry, "Practical and lightweight domain isolation on Android," in *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, ser. SPSM '11. ACM, 2011, pp. 51–62.

[54] A. Gupta, A. Joshi, and G. Pingali, "Enforcing security policies in mobile devices using multiple personas," in *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 2010, pp. 297–302.

[55] O. Oluwatimi and E. Bertino, "An application restriction system for bring-your-own-device scenarios," in *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*. ACM, 2016, pp. 25–36.

[56] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, p. 5, 2014.

[57] P. Kodeswaran, V. Nandakumar, S. Kapoor, P. Kamaraju, A. Joshi, and S. Mukherjea, "Securing enterprise data on smartphones using run time information flow control," in *2012 IEEE 13th International Conference on Mobile Data Management*. IEEE, 2012, pp. 300–305.

[58] D. Feth and C. Jung, "Context-aware, data-driven policy enforcement for smart mobile devices in business environments," in *International Conference on Security and Privacy in Mobile Information and Communication Systems*. Springer, 2012, pp. 69–80.

[59] J. Leyden, "Your phone may not be spying on you now - but it soon will be," https://www.theregister.co.uk/2013/04/24/kaspersky_mobile_malware_infosec, 2013.

[60] R. Templeman, Z. Rahman, D. Crandall, and A. Kapadia, "Placeraider: Virtual theft in physical spaces with smartphones," *arXiv preprint arXiv:1209.5982*, 2012.

[61] R. Schlegel, K. Zhang, X.-y. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound trojan for smartphones." in *The Network and Distributed System Security Symposium*, vol. 11, 2011, pp. 17–33.

[62] O. G. Consortium, "Open GIS simple features specification for SQL. revision 1.1," 1999.

[63] M. Shehab, G. Cheek, H. Touati, A. C. Squicciarini, and P.-C. Cheng, "User centric policy management in online social networks," in *Policies for Distributed Systems and Networks (POLICY), 2010 IEEE International Symposium*. IEEE, 2010, pp. 9–13.

[64] R. W. Reeder, L. Bauer, L. F. Cranor, M. K. Reiter, and K. Vaniea, "More than skin deep: Measuring effects of the underlying model on access-control system usability," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011, pp. 2065–2074.

[65] L. F. Cranor and S. Garfinkel, *Security and Usability: Designing Secure Systems that People Can Use*. O'Reilly Media, Inc., 2005.

[66] K. Fisler and S. Krishnamurthi, "A model of triangulating environments for policy authoring," in *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*. ACM, 2010, pp. 3–12.

[67] I. F. Progri, "Wireless-enabled GPS indoor geolocation system," in *IEEE/ION Position, Location and Navigation Symposium*, 2010.

[68] C. Feng, W. S. A. Au, S. Valaee, and Z. Tan, "Received-signal-strength-based indoor positioning using compressive sensing," *IEEE Transactions on Mobile Computing*, vol. 11, no. 12, pp. 1983–1993, 2012.

[69] S. Ali-Loytty, T. Perala, V. Honkavirta, and R. Piché, "Fingerprint kalman filter in indoor positioning applications," in *2009 IEEE Control Applications,(CCA) & Intelligent Control,(ISIC)*. IEEE, 2009, pp. 1678–1683.

[70] A. S. Paul and E. A. Wan, "Rssi-based indoor localization and tracking using sigma-point kalman smoothers," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 5, pp. 860–873, 2009.

[71] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: a survey," *ACM Transactions on Sensor Networks (TOSN)*, vol. 8, no. 4, p. 34, 2012.

[72] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A.-R. Sadeghi, and B. Shastry, "Towards taming privilege-escalation attacks on Android." in *The Network and Distributed System Security Symposium*, vol. 17, 2012, p. 19.

[73] L. Lee and W. E. L. Grimson, "Gait analysis for recognition and classification," in *Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, 2002, pp. 148–155.

[74] A. Alzubaidi and J. Kalita, "Authentication of smartphone users using behavioral biometrics," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1998–2026, 2016.

[75] T. T. Ngo, Y. Makihara, H. Nagahara, Y. Mukaigawa, and Y. Yagi, "The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication," *Pattern Recognition*, vol. 47, no. 1, pp. 228–237, 2014.

[76] Y. Ren, Y. Chen, M. C. Chuah, and J. Yang, "User verification leveraging gait recognition for smartphone enabled mobile healthcare systems," *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp. 1961–1974, 2015.

[77] A. Anderson, "XACML profile for role based access control (RBAC)," *OASIS Access Control TC Committee Draft*, vol. 1, p. 13, 2004.

[78] T. Moses *et al.*, "Extensible access control markup language (XACML) version 2.0," *Oasis Standard*, vol. 200502, 2005.

[79] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[80] Y. Kumar, R. Munjal, and H. Sharma, "Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures," *International Journal of Computer Science and Management Studies*, vol. 11, no. 03, 2011.

[81] T. Søndrol, "Using the human gait for authentication," Master's thesis, Gjøvik University College (Gjøvik, Norway), 2005.

[82] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying unique devices through wireless fingerprinting," in *Proceedings of the 1st ACM Conference on Wireless Network Security*. ACM, 2008, pp. 46–55.

[83] S. Banerjee and V. Brik, "Wireless device fingerprinting," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 1388–1390.

[84] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 94–104, 2016.

[85] Z. Kfir and A. Wool, "Picking virtual pockets using relay attacks on contactless smartcard," in *1st International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm*. IEEE, 2005, pp. 47–58.

[86] H. Saevanee and P. Bhattarakosol, "Authenticating user using keystroke dynamics and finger pressure," in *The 6th International Conference on Consumer Communications and Networking Conference (CCNC)*. IEEE, 2009, pp. 1–2.

[87] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos, "Progressive authentication: Deciding when to authenticate on mobile phones." in *USENIX Security Symposium*, 2012, pp. 301–316.

[88] Apple, "iphone x," https://www.apple.com/iphone-x.

[89] B. Haskins, A. Nilssen, and A. Davis, "The evolution of the conference room and the technology behind it," http://cp.wainhouse.com/content/evolution-conference-room.

[90] A. Madhavapeddy, D. Scott, and R. Sharp, "Context-aware computing with sound," in *International Conference on Ubiquitous Computing*. Springer, 2003, pp. 315–332.

[91] B. Carrara and C. Adams, "On acoustic covert channels between air-gapped systems," in *International Symposium on Foundations and Practice of Security*. Springer, 2014, pp. 3–16.

[92] Pronto.ly, "Ultrasonic handsfree authentication technology," http://www.prontoly.com/.

[93] Android, "Android enterprise," https://enterprise.google.com/Android.

[94] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," *SIGOPS Operating Systems Review*, vol. 33, no. 5, pp. 186–201, Dec. 1999.

[95] G. Petracca, Y. Sun, T. Jaeger, and A. Atamli, "Audroid: Preventing attacks on audio channels in mobile devices," in *Proceedings of the 31st Annual Computer Security Applications Conference*. ACM, 2015, pp. 181–190.

[96] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–380, 2006.

[97] C. Medina, J. C. Segura, and S. Holm, "Feasibility of ultrasound positioning based on signal strength," in *The International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2012, pp. 1–9.

[98] B. Draffin, J. Zhu, and J. Zhang, "Keysens: Passive user authentication through micro-behavior modeling of soft keyboard interaction," in *Mobile Computing, Applications, and Services*. Springer, 2013, pp. 184–201.

[99] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of wifi signal based human activity recognition," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 65–76.

[100] B. Smagowska and M. Pawlaczyk-Łuszczyńska, "Effects of ultrasonic noise on the human body—a bibliographic review," *International Journal of Occupational Safety and Ergonomics*, vol. 19, no. 2, pp. 195–202, 2013.

VITA

VITA

Oyindamola Oluwatimi was born and raised in Washington, D.C. He was awarded the Dozoretz National Institute for Mathematics and Science full scholarship to attend Norfolk State University, from which he graduated *magna cum laude* in 2011 with his bachelor's degree in computer science. He was then admitted into the computer science PhD program at Purdue University which was funded by the Purdue Doctoral Fellowship and GEM Fellowship.

Oyindamola's research area was mobile security and privacy. In particular, he focused on developing and enhancing access control techniques for devices within mobile information technology infrastructures. In addition, Oyindamola investigated the use of contextual information extracted from the environment, within the physical and computing realms, to influence access control decisions in context-aware systems.

Oyindamola had participated in several internships at various institutions and companies including Iowa State University, Texas A&M University, University of California Los Angeles, Johns Hopkins University Applied Physics Laboratory, Soonchunhyung University in South Korea, and Harris Corporation. In addition, during the last two years of his PhD program at Purdue University, Oyindamola interned at Analog Devices Inc. as a software engineer.