M.Sc. Thesis
Master of Science in Engineering

DTU Compute
Department of Applied Mathematics and Computer Science

Context Aware Access Control

Dimitrios Iatrou (s150945)

Kongens Lyngby 2017

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

Logical access control aims to protect digital assets from being compromised by unauthorized users. While, in most cases, logical and physical access control have clear boundaries, this is not the case when data is projected on an external device. Even state of the art access control policies cannot ensure data confidentiality if the environment surrounding the computer that contains the data is not secured properly. All traditional access control models are based on the same assumption, that the system is physically secured. However, this is not always true since data can be represented in numerous ways and different devices. The problem becomes more distinct in corporate environments with open plan offices where monitors and printers are stationed in plain view.

In this thesis, we study the possibility of a Sensor Enhanced Access Control(SEAC) model, which is constantly aware of its surroundings, enhancing it with contextual awareness. The aim is to extend logical access control to physical objects with the use of sensors providing all the required information of the system's environment, such as who is currently present and whether someone is approaching. The access decisions are taking into consideration both the logical policies and the sensor readings. The theoretical model evaluates the access levels of data represented on the monitor with the access levels of an approaching user. If the approaching subject is not authorized to access any of the opened files, the system proceeds to make the windows representing them invisible.

A prototype of the model has been developed to provide proof of concept. The prototype secures the displayed data, by modifying their windows, making them invisible to unauthorized subjects nearby. A user tracking algorithm and a camera track users at all times, enabling contextual awareness. The logical access control is regulated by the Unix system and the existing access control policy. The prototype can be extended with additional sensors, that can provide additional features and strengthen security.

Preface

This Master thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a Master degree in Computer Science and Engineering.

Kongens Lyngby, July 23, 2017

[illegible]

Dimitrios Iatrou (s150945)

Acknowledgements

I would like to express my gratitude to everyone that contributed with one way or the other for this project. More specifically i would like to thank my supervisor Christian D. Jensen for all the advice and guidance through the entire project.

Contents

CHAPTER 1

Introduction

Information has and will always be extremely important in every business field or aspect of life. Having more information from your corporate antagonists or enemy nations means everything. It can provide the foundations for superiority or lead to total collapse upon losing essential information parts. Regardless if we are referring to personal, business or national information, information security is one of the most important aspects of system security. Since the number of systems with access to files keeps getting higher and higher, data confidentiality is important to preserve; therefore, user authentication is required almost everywhere. Authentication is the verification process to prove that the user is legitimate while also providing information on how the user interacts with the system, for example, what time did the employee logged in and out of the system. The authentication process which can include a single or multiple sessions can be achieved in many ways from the standard usernames and passwords to physical authentication devices with smart cards or remote authentication with one or more biometric experts. Most systems enforce single authentication sessions such as the traditional password-based approach which authenticates the user at the very begging of the session. However, after they gain access to the system, there is no possible way to verify that it is the same user for the rest of the session which raises serious security concerns, especially for long sessions. For instance, in cases when the user logs in his computer and visits the bathroom shortly after, leaving the computer exposed for someone else to take over his session. Ideally, users will terminate their session once they leave the computer otherwise everyone can access their logged in account.

In reality, users are not concerned enough to terminate their session and tend to share devices with ease, without taking into consideration the security risks of their action. Even two-factor authentication is not sufficient enough and unless the system enforces continuous user verification sessions, there is no way to tell if the user is the same the entire time. Extending the traditional single event authentication models with continuous sessions is inevitable, although it is still in its early stages. Continuous authentication can be established with various techniques according to the business needs and budget, from face recognition to behavior and walking patterns. The authentication system works in the background, monitoring and learning the user's behavior, which will allow it to identify the user through the entire session. Face recognition or keystroke patterns are all unique user identifiers, making them ideal solutions for the purpose of continuous authentication, since they can be continuously monitored and take actions upon failing to identify the user.

Authenticated users are associated with every action they are authorized to perform along with access to files, buildings and areas for which they have rights. These rights are set by various predefined policies, providing Access Control that can also be enforced in physical locations, such as computer rooms and objects, such as computers and printers. Generally, Access Control aims to selectively restrict unauthorized users to access various resources and locations. More specifically, we can describe access control as physical, where the access is limited in buildings, facilities, or physical computer assets and as logical, where we focus on system files, digital information and network connections. Access control must be transparent to the user without interfering with his work, otherwise it will be bypassed or even compromised for their convenience. Exchanging id tags, writing passwords on papers or leaving office doors open are common in many cases where the users want to make things faster and easier. Transparency requires the security mechanisms to operate calmly in the background, which in the times of ubiquitous computing is feasible more than ever, considering that they are located everywhere in our lives constantly collecting data. Collecting and analyzing the data can enhance security providing continuous authentication and combined with access control, can offer a fine-grained security system capable of enforcing the security measures in real time, making it an ideal solution for companies where there are many different persons with various access levels.

Passwords, data, private calls and confidential discussions are easy to be overheard or peeked at in small office spaces. Shoulder surfing is categorized as a type of social engineering technique, which is used to retrieve passwords and other information from victims by peeking at their keyboards when they type or at the information on their screen. It is a major problem in modern businesses where offices are packed as much as possible and especially cubicle offices that are designed without security in mind, having small spaces and exposed screens to everyone that walks by. With conventional means, the computer is not able to detect whether the user is the one pretending to be, if he is alone in the room while typing his password and if an external user is close enough to their screen to retrieve information. What is required is a more advanced security plan where the system is aware of its surroundings, the employees and the changes that occur in real time. To achieve these, the system is required to be able to identify its users and their rights while it keeps track of their position in the company. This thesis focuses on the problem of shoulder surfing and how possible it is to provide a secure and transparent working environment for both the users and the information they are authorized to handle. Our focus lays on the three main topics of shoulder surfing: detect approaching users, identify them and take appropriate measures when the user is unauthorized or has lower privileges. We propose a context-aware access control model capable of identifying and tracking inhabitants within a company building. When needed, the system will protect the current user's work by hiding or covering the current window of the specific monitor. The model is based on the current access control techniques enhanced with various sensors, providing context awareness and extending logical access control to physical objects. To provide a proof of concept, a prototype has been developed that can be integrated into any UNIX system.

1.1 Context Awareness

As the trend of ubiquitous computing keeps expanding to more and more everyday devices, contextual awareness is easier than ever [AlM+03]. Concerning pervasive computing [Cov04], every smart device is capable of being aware of its location and its surroundings. As context we adopt the definition of Anind K. Dey and Gregory D. Abowd in [Abo+99] which is defined as any information that can be used to characterize the situation of a person place or object which is considered relevant to the interaction between the user and the application. The definition helps us realize that context is more than being aware of the users location but there are parameters that may be constantly changing over time. Therefore as described from Schilit, Adams, & Want in 1994 [SAW94] there are three important aspects to consider:

- where you are
- who you are with
- what resources are nearby

These aspects make the adaptation of a context aware system more complex since it eliminates the abstraction factor of the system and needs to be more precise.

The real purpose of context aware computing is to make the user's life easier by triggering actions or providing features according to the readings [Cov+01] and in some cases the user's history. There are various examples of contextual awareness whether the system passes simple information or provokes a process. For instance, simple interface features that provide material according to the user's location such as restaurants and bars nearby or automatic pairing with a Bluetooth speaker when entering a specific room or when invoking the print command for a document the system chooses the nearest printer according to the user's location. Of course there are also more complex cases where we implement conditional actions using the same principle as an If-Else statement. For example If the user is still in the room keep computer working Else lock the screen. Generally, for an effective context aware system we rely upon the system's ability to gather information from the connected devices (cameras, printers, motion sensors etc.) comprehend the gathered information and perform actions according to the user's and system's requirements.

In an office environment, we can understand how the three aspects we described are changing. For example, at a given time the importance of information handled by the user may change from one time to another or another user steps in the office. Therefore we require continuous authentication for the users to be able to keep track of their associated services and their data while protecting them from users who do not have sufficient rights. Identifying the current user can lead the system to categorize the resources surrounding the system at the given point thus taking appropriate measures when and if it is needed.

A seamless persistent authentication model is required that can identify users remotely in order to interfere as little as possible to the office's everyday life. Biometrics are currently the best candidate since they can provide remote authentication on sufficient level. Whether we use walk patterns, face recognition retina scanners, etc. the distinguishing features are unique for every individual.

The system has to be able to keep track of the authenticated and non authenticated users and estimate their location in the building. Having a possible location for most of the time allows us to take appropriate measures whenever the person is in the range of the service or leaves out of it.

1.2 Motivation

Almost every building nowadays features one or more forms of an access control mechanism; it could be a guard at the front door checking IDs, swipe card terminals, password terminals, etc. The purpose is to ensure that only authorized users are entering the building and differentiate them from guests and other unauthorized entities. While this form of physical access control can provide a good line of defence against intruders, we also need to provide further authorization as the user advances in other parts of the company. This can be proved challenging, since the security policies should be transparent to the user without disturbing his everyday life. Therefore, locking everything and everyone behind a door with PIN locks is not a viable solution.

Once we get passed the main authentication points in every entrance, then it is up to logical access control to grant or restrict access to data. Each user has specific policies for the data he can access. Once the user is authenticated, the system assumes that this is the actual user, even if he is not. The policies are enforced after a successful login but from this point on the system cannot provide any protection in cases where the user leaves the computer with an open session to grab a coffee for example.

In large corporate buildings, while a number of employees may work on the same floor, it does not necessarily mean that all the users have the same privileges. A typical office floor can contain several cubicle offices such as the one in Figure 1.1 which makes shoulder surfing a great problem. Employees passing by monitors with open blueprints or reflections on windows with classified designs can be a major leak of information. Even if the company operates with a great amount of trust among its employees, there is also a chance for an authorized user to be accompanied by a guest and it is easy to understand that trust is not an option here.

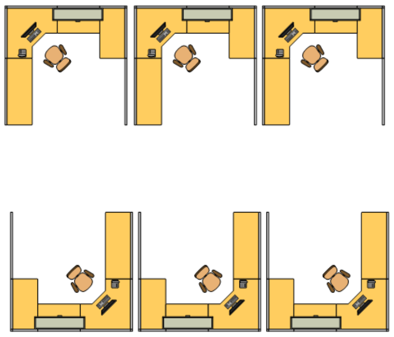


Figure 1.1: Cubicle Office

There is a growing need to actively address the problem of shoulder surfing with an effective, yet easily integrated and low-cost, solution, that will also adapt to the employee's habits, without forcing him to bypass the security measures for his convenience. Our proposal combines traditional access control policies with persistent authentication to provide a smart and more reliable security plan for corporate buildings.

1.3 Objectives

Our goal is to develop a prototype nested on top an existing Unix system. We use the current authentication criteria and techniques in the same manner with the original operating system; therefore, every user is bound with an id and an inode number that identifies processes executed by them and their access rights.

Access control is being handled by a virtual file system mounted on the existing file system and is responsible for retrieving the user levels and associate them with the current processes.

Authentication and tracking are implemented using a dome camera and a simple motion detection algorithm. We consider every approaching user to be authenticated upon his entrance to the corporate building and his identity is known for the whole tracking process.

Window handling is performed with two different propositions which will be evaluated. The first one identifies the currently opened windows and their required access rights and proceeds to change window's opacity making it gradually transparent or visible. The second one monitors all the window's screen location, their size and their required privileges. Then it proceeds to create a window with the same size and at the same location as the one we are trying to cover. The cover window changes its opacity and color gradually.

In every office, we have set visibility zones to determine the distance from the monitor or any other output device. The visibility zones determine the changes in opacity and color of the windows and the windows covering them.

1.4 Contributions

Our contribution is the development of a context aware system that combines data gathered from sensors to determine whether the system's environment is safe to display information and run services. Depending on the readings the system will be able to block content and services whenever an unauthorized user is in range. The persistent authentication enforced by our model will enhance security in company buildings and offices with as little hassle as possible for the users.

We could divide/summarize the contributions in the following:

- A module that extends the current version of Ubuntu Linux to provide context aware access control. The logical access control makes use of the default access control of UNIX with the decisions being based on the data gathered from sensors. The module is built as a virtual file system and can be mounted on demand using the existing policies.
- A simple motion tracking algorithm that will help us determine when a person authenticated or not approaches or leaves a work station. Having set visibility zones, with the user's screen being the center, the algorithm will forward a warning output of a possible intrusion or user absence to the window manager.
- A window manager to handle windows and files according to the predefined access policies set by the system administrator, the data gathered from the user tracking and the matching output from the biometric algorithm.
- Two implementations that extend the window manager and are focused on securing the open windows. The first changes the window transparency making them invisible when the unauthorized subject is close to the screen. The second covers the windows with an overlay area with different transparency levels and background color.

1.5 Thesis Organization

In this chapter, an overview of the thesis was presented along with our objectives and contributions the remaining chapters are organized as follows. Chapter 2 provides an introduction to access control and some of the basic models that are currently used. Chapter 3 presents the persistent authentication model and how sensors could enhance the traditional security models to provide a secure and smart work environment. Chapter 4 describes the basics regarding the Unix filesystem and virtual file systems. Chapter 5 analyzes the need for context aware access control and how to enforce logical access control in physical objects. Chapter 6 and 7 present the design and implementation of the prototype where the prototype's components are identified and how their implementation was approached. In Chapter 8 we have documented all the tests we conducted to determine that all the components work as indented. Finally, in the final chapter, we review the contributions of the thesis along with ideas of further development for the prototype. In the appendix, there are the detailed tests we followed for the evaluation.

CHAPTER 2

Access Control

Every organization or individual has a number of assets both physical such as a computer with files, hard copies of information, a family relic and logical such as sensitive information and data. These assets have a different value for each one and their loss might cause from insignificant to catastrophic results for the owner. This is what security is trying to achieve, the protection of the assets that matter. The cost of the assets is purely dependant on the proprietor. Security requires careful planning and always being up-to-date with new attacks and new technologies.

Securing assets has always been a trade-off between security and usability. A foolproof security plan often has usability drawbacks which cause the users to bypass it for their convenience. Storing all the organization's files for instance on a computer disconnected from everything and everyone is not a viable solution even though it is secure. Disrupting the information flow of the employees will make their work inefficient thus causing more damages to the company. Consequently, before planning a security infrastructure, we have to conduct a net cost analysis of the organization's assets. There is no need for a security system which will cost more than the net cost of the asset. Traditionally every security system focuses on addressing the following three aspects known as CIA:

- Confidentiality: Prevent access to information from unauthorized users.
- Integrity: Prevent any data modification from unauthorized users.
- Availability: Ensure information flow to authorized users when it is required

There have been many solutions to achieve the CIA goals and many attacks to disrupt them. One of the most common solutions and the one we will be focusing on, is to handle access to information mandated by policies, is access control.

2.1 Access Control Overview

Access control is the security constraint responsible for restricting or permitting any activities from authorized users, regulating each attempt for admission to a facility or retrieve system resources. Access control systems can be implemented for any infrastructure with various levels and at any given point. Facilities may use guards or swipe cards to protect physical resources. Operating systems have access policies set for the legitimate users to protect sensitive data and directories. The primary goal of an access control system in an operating system is to mediate how information is shared with the users and the software. A fine-grained access control system can ensure the Confidentiality, the Integrity and the Availability of the exchanged information in a business environment, while when absent or poorly implemented could set information in risk.

Access control could be differentiated between physical and logical. As physical we identify the aim to protect physical assets such as hardware, hard copies and the access in an office or building. On the other hand, logical access control focuses on the users of the system. Whereas in most of the cases we could easily see their boundaries, things get more complex in cases where they overlap each other. Whenever data is projected to a monitor or an output device in general, then its protection is considered physical.

2.1.1 Physical Access Control

In its strictest form, physical access is limited to authorized personnel who have undergone security training and have the appropriate security clearance levels. Furthermore, the asset to protect (computer, facility, computer room) must be enclosed in a secure perimeter with additional physical security controls for further protection. Even though at first glance this plan might fit more in a military environment, it is not far different from a company building. In a business environment, in most cases, we require from the users either to swipe their id cards before entering the building or have their identity card checked from the front desk. This is also a defined perimeter. As we advance further in the building, additional authentication stations are set such as pin code and swipe card locks.

When planning a physical access control system, we must also take into consideration the wiring and the connections used to power elements of the system. Similarly with phone and internet lines and anything else that might disrupt the system's optimal behavior. Consequently, we have to identify all elements that are part of the system and plan accordingly.

Another important aspect is the review of the established physical access controls for each sector. The review must be done both during operating hours and outside the normal schedule. We need to review how effective are the established controls to prevent unauthorized entrance to intruders and how easy is to bypass them during "inactive" hours. In summary, an organization's physical access control plan is considered effective when it can provide solutions to prevent interruptions regarding services, physical damage upon assets, theft, sensitive information leak, system's integrity violation.

2.1.2 Logical Access Control

Logical Access Control is focused on mediating the access of every user to data whether this includes viewing, modifying or using it. Logical access control is not only limited to restrict the access to an asset like physical access control but also to regulate what the user is allowed to do with it. The effectiveness of a Logical Access Control system is based on the organization's needs since the policies required for its implementation has to balance between security and usability. A general rule of thumb is to limit access of each employee only to the data required for their work without disrupting the work flow.

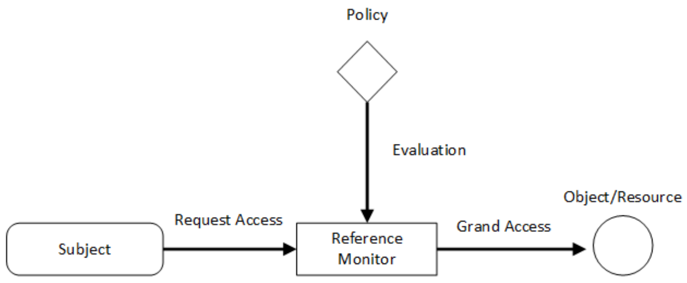


Figure 2.1: Access Control

(revise image) In Figure 2.1 we can see a simple form of an access control system. As a subject, we identify every active entity such as users and processes. Objects are passive entities which can be used or access from an object and could be files and devices such as printers. The reference monitor regulates whether the user has rights to access the specific object according to the predefined policy. For every access attempt, the reference monitor has to evaluate the policy to grant or reject access.

2.1.3 Planning an Access Control System

The planning of a new access control system must be based on three control abstractions: access control policies, models, and mechanisms [HFK06]. Access control policies are responsible for regulating who has access to the information and under which conditions. The policies can be bound to a specific application or user actions within the company. To enforce the policies, we require an implementation of a mechanism that will handle it upon an access request. The implementation depends on the company's needs and may vary from a simple hash-table stored in the system to a database to check the access rights. Security models represent the security limitations of a system. They are used to formally visualize the security policies enforced and help develop access control mechanisms such as Access Control Lists (ACLs). Access Control Matrix is one of the most common examples of an access control model which can be used in every system. It consists of a matrix defined by a set of subjects which are represented by every row, a set of objects represented by each column and for every entry, there is a set of rights which define what operations can be performed from the subjects on the objects as seen in Table 2.1. The matrix is often sparse due to most of the objects that do not have a defined object policy. Therefore it makes its storage and iteration inefficient. Access Control Lists (ACLs) is one of the most common methods of storage which divides the matrix into columns [Lam74].

	Asset 1	Asset 2	Process	File
Process 1	read		read,write	read,write, execute, own
Process 2	read, write	read	read, write, execute	read,write
Process 3			read, write, execute	

Table 2.1: Access Matrix example

Access control systems are categorized either as discretionary or non-discretionary. Generally, as non-discretionary we categorize those based upon a rule to decide which users have access and as discretionary those where the owner or a specific user decides the permissions of a file.

2.1.3.1 Discretionary Access Control (DAC)

Discretionary Access Control grants the owner of the file all the rights to decide who can have access and what he can do with that access such as modify, read, etc. Access Control Lists (ACLs) are the most common example of DAC which is used widely by many operating systems. In UNIX where everything is considered a file, ACLs enforce permissions for three different categories, owner/group/other. The permissions are categorized as read (r), write(w) and execute(x) and in Table 2.2 there is an example which is similar to the access matrix that was described earlier.

Although its flexibility as a policy makes it quite useful for commercial software, there are some serious security issues. Even though the creator of the file or the person authorized to handle it may restrict any user from modifying it and allow them only read access, nothing can stop them from copying the file's contents and use them as they prefer. Then the new file can be modified and shared how it pleases the new owner thus risking information integrity in the system. Additionally, since every process inherits the identity of the user that triggered it, it makes it vulnerable to Trojan attacks. For example, if a user executes a file with malicious software then in the log files when trying to trace the attack, it would look like everything was begun by that particular user.

User	**File 1**	**File 2**	**File 3**	**File 4**
John	rwx	rw		r
Alice	rw	r	rwx	r
Bob	x		rw	

Table 2.2: Access Control Lists example

2.1.3.2 Non-Discretionary Access Control (NDAC)

Every other policy is considered as Non-Discretionary where the user is not authorized to modify or grant any access rights but require administrator privileges thus allowing the company to set universal policies according to their security needs.
We will briefly describe the most commonly used policies.

Mandatory Access Control (MAC)
MAC is one of the most common policies used both in the military and business sector. The access control decisions are part of the system and which is decided by a central administrative authority. Users do not have any right to the policy and cannot change them even if they are the owners of the information. System mediates user access and enforces the defined policies. Generally, we classify data according to their importance and we categorize users according to their position or their security clearance, this way we can set security levels and set policies. The military sector is the best example where we use Secret and Top Secret levels to label information. Users that have security clearance only for Secret cannot access Top Secret information. Clearance levels permit users to access information with the same level or lower than theirs and restrict the access to anything higher. An access control model that uses levels is categorized as a multilevel security model. Bell-La Padula Confidentiality and Biba Integrity models are the most common multilevel security models.

Within a company or the military, there are different departments which do not share the same information. Therefore, we have to make sure that even when a user has Secret as his clearance level, he can access information of his department only unless other departments share the same data. The association of security clearance and departments can be easily described in the following security lattice in Figure 2.2. As a system high of the lattice, we assume the maximum security level and the lowest security level as system low.

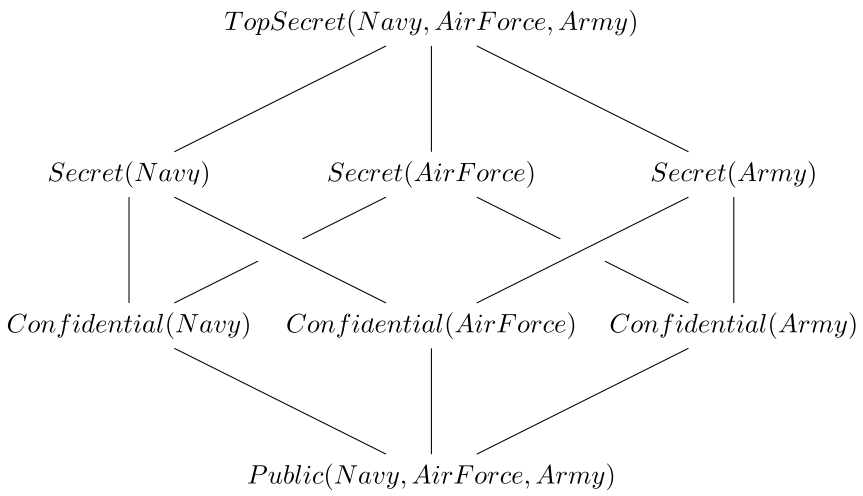


Figure 2.2: Security Lattice Army Example

Bell-Lapadula (BLP)
BLP is a multilevel security model which can be described as a state machine. Its uses are mostly in the military sector. It consists of security label on the objects and security clearance levels for the subjects. Security labels vary according to the sensitivity of the information, from the least sensitive to the most the labels are: ”Public,”Restricted,”Confidential,”Secret,”Top Secret”. For the relation between labels, the term domination has been introduced. If information flow is granted from a label B to a label A then A dominates B. For every security level, a subject can only access objects of the same or lower security level whilst access to higher level objects is restricted. Moreover, there is also the *(star)-property which does not permit any modification from a higher level subject to lower level object. In sort as seen in Figure 2.3, BLP enforces for any given security level No read-up and No read-down properties.

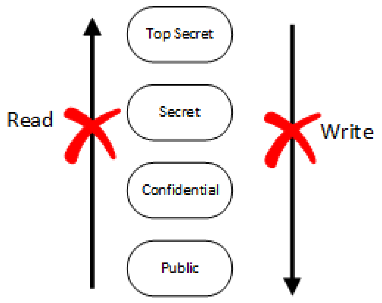


Figure 2.3: Bell-Lapadula (BLP)

Biba Model

Whilst Bell-Lapadula focuses on data confidentiality, Biba aims to preserve the integrity of data and it is the complete opposite of Bell-Lapadula. The key difference between the models is that Biba is more business oriented where information integrity is the key rather than confidentiality and who is authorized to read the information. Biba introduces integrity levels and users are allowed to modify data of the same or lower integrity. On the contrary, users are permitted to view information with the same or higher integrity level. Given a subject with an integrity level A the subject is allowed to modify an object with an integrity level B only if A dominates B and the subject can read the contents of the object only if the integrity level B of the object dominates the integrity level A of the subject. In short, Biba enforces for any given integrity level No read-down and No write-up. The Biba model as we can see from the Figure 2.4 is the direct inverse from the Bell-Lapadula in Figure 2.3.

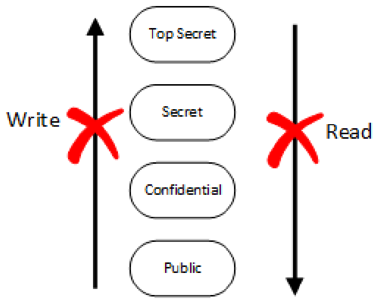


Figure 2.4: Biba Model

As an example, we could describe Biba in a business environment as the need from every company member to be able to read the CEO's announcements while a simple employee is not allowed to modify them.

Chinese Wall

The Chinese Wall policy was developed from Brewer and Nash [BN89] with a more business approach. The aim was to resolve conflicts of interests between consultants within the banking and financial sectors. The focus is mainly on the confidentiality of information. The basic idea is that a client is assigned to a consultant within the company, and consequently for the consultant to do his job, he needs to have the client's data. These data can be considered as insider knowledge in some cases and any potential "leak" of these can disrupt any competitive advantage the company may have or used for ransom.

For every company, there are many clients whose information is considered a conflict-of-interest. The clients are divided into business categories or areas. A consultant upon gaining access to a user's data within a business category is not allowed to have access to another client of the same business category otherwise we have a conflict of interest. The policy must change dynamically at all times keeping track of the user's history on what information he previously had access and restricting future access to information that may cause conflicts of interest. Any consultant can read public information regardless of his history. Another MAC policy may also be enforced by the Chinese Wall policy.

Role-Based Access Control (RBAC)

RBAC is considered one of the three primary policies along side with DAC and MAC which we described earlier. Access permission or rejection is decided upon the role of the user. Every user in the company has a specific role which is used to group up the access policies. A user can have more than one role in the company and each role can be assigned to different users. Granting roles to new users or revoking them whenever the person leaves the company, allows a smooth transition of access rights and responsibilities within the organization. Additionally, the privileges of each role can be easily updated or changed for all the users at the same time with no need to update each user individually making administration easier.

For example, if the organization hires a new person for the HR team then the role of HR is assigned, hence access to the resources required for the job. If an employee is re-positioned to another department, then the role is updated. The roles are updated and granted from the administration or management departments. RBAC requires a clear definition of privileges for every role since some role responsibilities may overlap therefore each role should be restricted only to the essential responsibilities of each job. In general, RBAC follows three primary rules.

Role assignment: The subject has to have a role assigned in order to have access permission.

Role authorization: The active role of a subject must be authorized before it is allocated to the subject. Thus combining this rule with the first one, we are making sure that users get only authorized roles.

Permission authorization: Permissions are granted for each role only if they have been authorized for that specific role. This rule combined with the earlier ones, limits the user permissions to the ones they have been authorized.

Attribute-based Access Control (ABAC)

ABAC is one of the most complex access control models but also the most flexible. It utilizes three different attribute types for the user, for resources and environment changes. The evaluation of these attributes grants or restrict access to users. As an example, a user can only access the employee database if he is employed by the company if his department is within the HR and only between the working hours of the company. We can see that the model supports an IF ELSE Boolean logic which is constantly evaluated, granting the system contextual awareness and a fine grained access control model. Due to its nature, ABAC can combine numerous attributes from different systems regardless of their diversity which makes it flexible to adapt to every company infrastructure.

As the company expands and more users are introduced to the system, there is no need to add more policies or modify the existing ones, as long as each user is assigned the required attribute [Hu+14]. Although due to its complexity, in larger scale companies the number of attribute evaluations scale according to the number of attributes introduced thus making an impact on performance.

Other well-known access control policies include: History-Based Access Control (HBAC) where access is decided upon user's activity history, for example, the access requests the user has performed in a time limit, etc. Identity-Based Access Control (IBAC) is a more personalized model which requires each privilege to be assigned to a user individually according to his needs. Rule-Based Access Control (RAC) in which access is granted or restricted based on a set of rules defined by the administrator or the company. It is similar with the Attribute-Based Access Control models and the rules that require to be met may vary from allowed working hours to user roles. For example, a set of files can be accessed only by teachers while others only from students.

2.2 Identification and Authentication

The scope of authentication is to verify the identity of the user. Authentication can have many forms and is required in various cases. Whenever we need to verify the identity of a person requesting access to a company's service, when we require a person's passport to verify that is truly the one he claims to be or when we require to confirm the authenticity of software by reviewing its digital certificate. Usually, authentication in access control systems may be achieved with the typical combination of a username and password but could also be extended using external means such as biometrics or two different steps such as requiring a smart card swipe. Generally, for a user to be authenticated he needs to be verified using one or more of the following:

- Something he knows (PIN, password, his first dog's name etc)
- Something he possesses (Smart card, USB token or other hardware device)
- Something he is (Fingerprint, retina scan, face etc.)

2.3 Access Control

Access control is the step after the user has successfully been verified. Whenever there is an access request for a logical asset such as a file, a process or a physical asset such as a printer, then the system has to decide whether the authenticated user has sufficient authorization levels to permit the access. While in authentication we require the user's interaction, in authorization the decision is purely from the system and the defined policies. The authorization policies can be granted with various techniques similar to the ones we described earlier. Commonly each system permits three types of access. Read a file or its contents, Write which typically means modify a file, update or change its contents and Execute in cases where users desire to launch a process or a program.

2.4 Accountability

Using logs and various user records we can keep track of the user activity both when accessing a file and any associations with other users within the system. The logs help us trace security violations and possible security breaches. In cases where the system has been infected with a malware then logs can contribute to recreating the event gaining information on how it was begun and how it was spread. Logs can also be used as evidence in a court case if they are handled properly. They can be automatically generated whenever an incident occurs. For instance when a user logs in the system outside the business working hours or from a different location.

CHAPTER 3

Persistent Authentication

User authentication has not changed much over the years even though technology has seen groundbreaking progress especially now that everything moves towards smart devices and pervasive computing. Most of the devices focus mostly on an authentication model that is infrequent and persistent, where the user is required to prove his identity with a password or other similar techniques. The infrequency is a sacrifice we are willing to make for usability since no one would be delighted to use a system with repetitive re-authentications.

Once the initial authentication is achieved the device cannot validate whether the user is the same user throughout the whole session or that he was the actual user in the first place and not someone with his credentials. While in a trusted environment, such as your family house, persistence could be proven as a minor security flaw, the problem scales in larger environments and mobile devices. Cases where a laptop falls into the wrong hands and the user has been logged in his usual accounts, then the intruder gains access, without having to do anything in particular. Cases, where the laptop was the company's laptop, give a chance to the intruder to get his hands on the company's VPN information or classified data. For stationary computers, the fact that in many cases they are in a trusted environment and their high level of physical security can lead us to overlook the security flaws of persistent authentication.

Despite the flaws, we could not require from the user to repetitively authenticate over short periods of time; therefore, we require a more transient authentication model [NC02]. The transiency offered by sensor based security systems whether it is based on biometrics [KG00], motion sensors or RFID cards offer a contextual awareness to the systems authentication model [CN05]. Thus authentication is frequent and requires less human interaction.

In our model, we combine persistent and transient authentication in a system which can extend the initial user authentication session with contextual information gathered from the user's environment. Once the user is authenticated, the system keeps track of the user using CCTV cameras or other sensors thus providing location based security for the company restricting unauthorized access. The model periodically verifies the user's identity using a simple biometric algorithm and continuously authenticating users without interfering with their everyday company habits. An overview of the persistent authentication model can be seen in Figure 3.1.

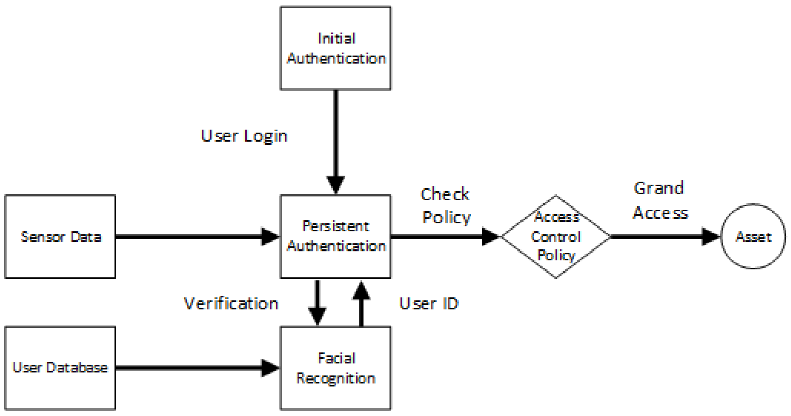


Figure 3.1: Persistent Authentication Model

Both identity verification and location tracking offer contextual awareness to the system which will be sensitive to possible changes to users and their environment. As we see in the picture for persistent authentication, we take into consideration both the sensor readings and the initial authentication session [KH08]. The sensor readings are verified with the biometrics database and if the user is identified, then we return his user ID. When we have the user ID, the system checks the access privileges. If the user has the sufficient rights, then access is granted; otherwise, it is rejected.

We do not provide a new authentication model instead we merely use the existing techniques for the initial authentication while we integrate context awareness to extend it. It is up to the company policies and the access control mechanisms to provide data security.

3.1 Continuous Authentication

As we discussed earlier, several security problems occur due to the persistence of the traditional authentication models; then there is not much you can do. Even with the strongest security policies, users are still unpredictable and in a company, it is more than common to share devices for their convenience. Therefore we need a more transient solution to provide continuous authentication sessions. A promising approach was proposed from Corner and Noble in [CN02] using a hardware token or a small device. The main idea as seen in Figure 3.2 is that the user gets authenticated with some sort of a token or identifier bound to his id. The token could be anything, such as a RFID tag or a Bluetooth device, for instance, the user's smartphone or a smartwatch. As the user walks in the appropriate distance from the terminal, the verification process begins.

Zero-Interaction Authentication (ZIA) as described by Corner and Noble requires an authentication terminal and a token which use a short range wireless communication channel, such as RFID, Bluetooth or Wi-Fi. The channel provides an encrypted and authenticated link for exchanging messages. The messages are encrypted, but since the token does not have the required hardware, the decryption and encryption of the messages occur in the terminal. The token must be bound to the terminal to be able to get authenticated. Between the terminal and the token, an encryption key is shared for this particular session after the token's verification. The key is bound to the token, so only this can decrypt the messages between them. Of course, we have to do the binding before using it for the first time. Once the token gets in range, the terminal verifies it and the session key is exchanged. A challenge response is taking place between the two devices to keep track whether the token is still in range or not. Once the token is out of range, the cache files are encrypted and stored in the terminal until it comes back in range, thus minimizing response times. The challenge response ensures continuous authentication of the users and in the same time get an indication of their location in the building since they have to be in range of the terminal. The challenge response satisfies the transient authentication model since the session is terminated if the user fails to respond or he gets out of range. ZIA could be used for anything that could be automated, from logging in a computer to turning on the lights of a room once your Bluetooth device has come into range.

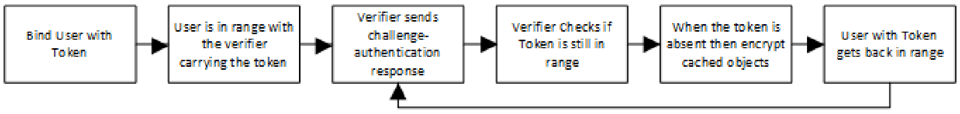

Bind User with Token
User is in range with the verifier carrying the token
Verifier sends challenge-authentication response
Verifier Checks if Token is still in range
When the token is absent then encrypt cached objects
User with Token gets back in range

Figure 3.2: Zero-Interaction Authentication (ZIA)

Although, ZIA by itself does not provide a secure way for authentication-verification. If we establish a system like ZIA with tokens, for example, then, once a token falls into the wrong hands, the intruder can enter the company building or access one of the computers. The token is, in fact, legit and the terminal identifies it as such, but we are not making sure that the user is the one he claims to be. Also in companies is not uncommon for the employees to exchange tokens for simplicity. ZIA's authentication mechanism is based on ”something you have” but combining it with an additional one, such as the ”something you are” method, then we have an even more secure model. The authentication will be based on two factors thus harder to exploit. Using ”something you are” as a mechanism does not limit usability and requires no interaction [SMS16] since it can be something like user's walking pattern or face recognition [Raj]. In our case, we utilize facial recognition which we believe that could cover effectively both the first authentication session of the user and the challenge response required to verify whether the user is still in range. [KG00]

3.2 Biometrics and Facial Recognition

Biometrics is one of the best ways to achieve transparent and persistent authentication. They could be remote and contactless like face detection or might require some sort of interaction or active participation from the user, such as a fingerprint or retina scanning [TLC09]. The greatest advantage of biometrics is that there will always be there for the user, no matter what. They cannot be forgotten or exchanged, thus making them one of the best alternatives to substitute passwords. The use of biometric scanners could be extended in various everyday cases, besides security. They could be used in stores, for example, to gather data about customers' shopping habits, which could be used in advertisement campaigns. They could also be used in houses, to keep track of pets or elderly people in case of an accident. As the demand rises, the technology will evolve further.

Although promising, biometrics need to be evolved further. There are many things that need to be taken into consideration. While some characteristics might be permanent and withstand changes through the years, some others are not that reliable. Fingerprints, for example, tend to change as the years pass, also depending on the subject's type of job. Additionally, younger ages have weaker fingerprint patterns than adults. Voice recognition or walking patterns might not be accurate, if the person has gone through an injury, for example. These are problems that should be taken into consideration, before deploying a biometric authentication system and keep our authentication database up to date, by providing it with any possible new features a user might have every once in a while. A successful biometric system is based on the quality of the training phase where we provide a significant number of test images. These images include both pictures with the object or entity we want to detect and pictures where the object is absent. We use the term positive image for the images where the object is present and as negative we classify the irrelevant pictures.

With regard to the goals we have set we require a non-invasive solution for continuous authentication hence we focused on facial recognition. The main issue with face detection and recognition is the false positives and false negatives. Meaning, the times when a person has mistakenly been classified as legitimate and the times when a legitimate user has mistakenly been rejected. We are also highly dependent on the quality of our equipment. Having high-resolution cameras improves the identification process making it more accurate and from a longer distance.

To minimize the failure rate, an extensive training phase with all the possible cases is required. We need pictures of all the possible angles, illuminations of the face both when wearing an accessory like glasses and without. This is not always possible and that is why probabilities are required. A method to enhance successful identification rates was proposed by Mads Ingwar in [IAJ13]. He proposed the use of multiple biometric experts whose outputs are fused and used to calculate a level of confidence. Based on that confidence a decision is made on whether to accept or reject it.

In summary, we could divide facial recognition into two phases. The first one is the training phase where we have to train the system with positive and negative images. Every user must be registered; otherwise, he is marked as unknown. For the registration, everyone is required to sit in front of a camera and have 20 or more pictures of his face taken. The better the camera quality and light conditions the better the results. All the pictures are converted to black and white. All the pictures are stored in the database under a user's id. Of course, all stored pictures should be encrypted and no one should be able to access them besides the administrators. Various security issues that might be raised here for the secure storage of data is beyond the scope of this project.

As for the second phase, we have the authentication. The system captures video frames of the user once he gets into range and compares each frame with the stored picture. A rectangle is drawn over his face and if a match is found then we indicate his id, otherwise, he is marked as unknown. There are various algorithms to chose from for the verification. For every algorithm, a number of 'confidence' is set which represents the "distance" between the test image and the closest stored set. The value can be optimized in each case accordingly. Another value is 'threshold' which is passed to the algorithm and used in prediction, distances greater than the threshold are ignored.

The actual details and technology of how face detection and recognition is performed is beyond the scope of the thesis.

[KG00]

3.3 Motion and Human Tracking

Motion detection could be easily implemented in various ways and hardware. From its simplest form with a common light sensor to more sophisticated approaches such as weight sensitive floor tiles. As a more realistic approach, we adopt the use of CCTV cameras for motion detection. Since it is already a requirement for our facial recognition algorithm and in most companies, they are already installed in various places, so no additional hardware is required. More importantly, they are relatively easy to maintain and fairly cheap to scale.

Tracking, on the other hand, is more complicated especially when we need to distinguish an individual inside a blob or fill the gaps through his walking path. These gaps where the user might be lost from the tracking program could be filled with the integration of continuous authentication. As we discussed earlier, the system occasionally transmits a challenge response to the user to check if he is still in range. If the user responds back, then we know he is close to the authentication point, therefore, his location.

For the motion detection and tracking, the implementation is based on the calculation of the absolute differences between frames. This is one of the simplest implementations that do not require a training phase for the algorithm to identify persons. Another essential point is that it requires less computation power and equipment which could also be mounted to an external system or single-board computers (such as Raspberry Pi, Arduino). Although simpler, it is more vulnerable to errors since it requires fixed conditions such as strategic camera placement, sufficient illumination at all times etc.As always, having a set of high-quality video feed is the key to better results.

In short, every time we capture a new frame the image is being scaled to a smaller size to decrease calculation time. Once it is scaled down, we convert it to grayscale and add blur for the edges. The color here is not important since we only need to detect the outline of the object. Therefore, color is extra information that increases computational time. For the blurring we use the Gaussian blur [HKZ87] function to decrease image noise and reduce detail. Smoothing the edges improves the results of the algorithm. In Figure 3.3 is an example on how the image is converted to detect the moving objects. The videos for the tests can be found in [PIR15].

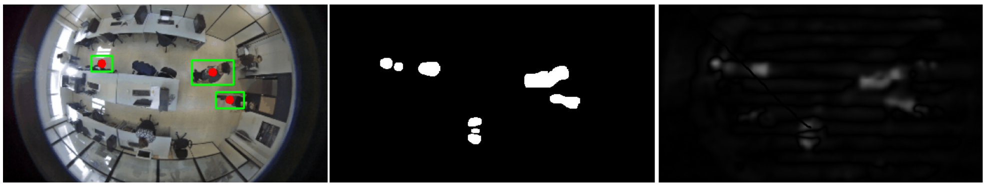


Figure 3.3: Motion Detection

Each frame is processed with the same technique and the decision depends on the differences between the images. The differences can be calculated by applying a simple Background Subtraction [EHD00],[Pic04] as shown in the following equation:

$$P[F(t)] = P[I(t)] - P[B]$$

Where P[I(t)] is the frame we want to test at the time t and P[B] is the background image used as a tester. In Figure 3.3 we can see the algorithm in action.

While tracking is essential for the contextual awareness of the system, constant tracking of individuals is not a requirement. The points of interest are limited in the office cubicle and more specifically the user's monitor. Consequently, a more sophisticated implementation will add unnecessary complexity to the system. The basic requirement is that we need to know whenever a person enters or leaves the cubicle.

3.4 Virtual Walls

The contextual awareness of the system combined with persistent authentication, offer a fine grained security model that can be enforced for both logical and physical access control. For organizations, physical access control is trickier to enforce since you cannot lock everything behind a door. Especially for open plan and cubicle offices where rooms are limited, shoulder surfing and unauthorized users roaming can be a serious security issue. An interesting approach to enforce physical access control in these cases is the proposal of virtual walls [Kap+07].

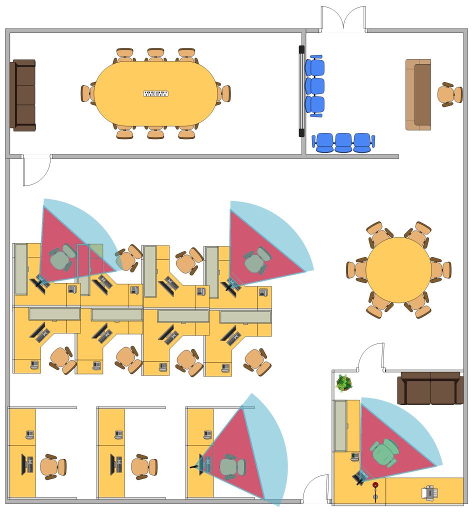


Figure 3.4: Visibility Zones

Virtual walls can be introduced to the system to substitute physical walls to restrict unauthorized or unwanted users from accessing an object or area. Persistent authentication keeps track of the residents without the need of multiple authentication sessions thus the system can easily distinguish between the authorization levels of each individual and unauthorized users. The concept of virtual walls is based on a context aware system that can analyze the organization's environment and decide which actions should be triggered. Various areas could be set within the company with no need of doors or walls where only authorized personnel is allowed. The system monitors the area and the persons who enter; then the alarm is triggered.

The same principle can be used to prevent shoulder surfing and protect privacy in cubicle and open plan offices. As seen in the Figure 3.4 visibility/readability zones have been set instead of virtual walls. These zones are a predefined safe distance from a computer monitor or other output device. The safe distance is such that an intruder cannot extract information. As seen in the aforementioned the safe zone is marked with a blue color and with red is the distance where security is enforced. The distance, of course, is not fixed for every room but needs to be adjusted according to the surrounding environment.

CHAPTER 4

Virtual File System & FUSE

Each operating system requires a method to organize and store data, files, and directories so that users can easily use and access. This method is widely known as file system [Jon07] and there are many different types according to the operating system, for instance, Windows use FAT32, NTFS Linux Ext2, Ext3, Ext4 each with its own features and disadvantages. In this chapter, there will be a brief overview of file systems in general and FUSE which will help to understand the structure of a virtual file system and its role in the project.

For every file in the UNIX file system, there are data that describe it, better known as metadata. These information vary according to the file type and may contain data about the size of the file, the owner, etc. Metadata in UNIX exist at multiple levels of the file system. The characteristics of the filesystem it self are contained in a superblock making it the most important container of metadata in the system. It contains among others data about the filesystem size, the number and location of each inode table, etc. Every request to access a file has to pass through the superblock, meaning that if the superblock cannot be accessed either can the files. Consequently, its preservation is crucial for the entire system, therefore, a copy is stored in multiple locations on the hard disk and memory.

File metadata contain various blocks of information about the file's size, type and also the required access rights. The owner of the file is represented by an id marked as user id and similarly for the group that has access with a unique group ID. All these information are stored in a data structure or better known as inode which is used for every object that resides in the file system be it a file or a directory. For every directory in the system, there is a list that maps each file name with the corresponding inode number an example can be seen in the following Table 4.1.

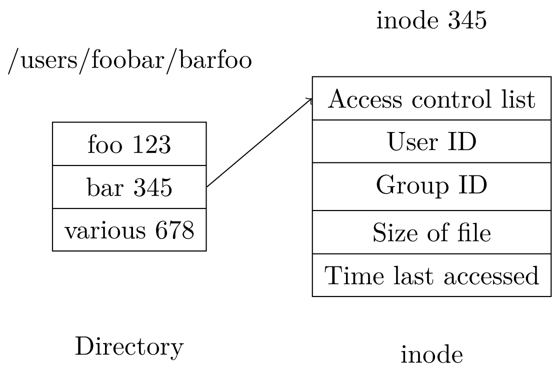


Table 4.1: inode Example

The relation between the inodes and file names is handled by the directory entry or dentry. For convenience and faster access to files, dentry is also responsible for caching the most frequently used directories.

4.1 Kernel and User Space

The kernel is the heart of the operating system which controls everything in the system. Every system has a certain amount of system memory that reserves for its operations. In UNIX the memory is divided into kernel and user space. Kernel space is the amount of memory reserved for the kernel to execute its code and provide the required services. Kernel space requires special privileges to be accessed and processes can access it only through system calls whenever there is a request for a service handled by the kernel such as creating a new process.

User space is the memory reserved for user applications. The user space has access only to its part of the system memory while the kernel can access the entire memory. Processes operating in user space cannot directly access the kernel space thus cannot access system resources like the memory. This limits system crashes and provides a more stable environment to work on.

4.2 Virtual File System

Generally, a Virtual File System (VFS) provides an interface between the kernel and a file system. A better description which is widely used, explains Virtual File System as an abstraction layer on top of a more concrete file system. The main purpose is to provide access to files regardless of the file system, the type of file or the Operating System. VFS ensures that every file and its data are saved correctly, therefore, is caches all the required information in memory when it is mounted.

A VFS has the same structure for metadata as a normal file system with superblocks, inodes and dentries. Since UNIX support multiple file systems mounted at the same time, it keeps a list of every superblock in the system while additional lists containing more information about the mount points and names of the mounted file systems also reside in kernel. Both the VFS superblocks and inodes contain similar information fields as in the traditional file systems about the files and the device identifier of the device the file system is stored.

The inodes in VFS although similar as the ones in traditional file systems, they are in fact different since they are stored in kernel's memory and are preserved in the VFS cache for as long as they are required from the system.

4.3 Filesystem in Userspace

Filesystem in Userspace (FUSE) [IBM14] provides an interface that allows every user to create a virtual file system nested in user space. The fact that the created file system itself operates on user space makes the developed application safer to execute since any possible bugs or crashes will have no effect on the system but only the application. FUSE does not require any special privileges to be mounted and provides an API to enable communication between a user application and the virtual file system.

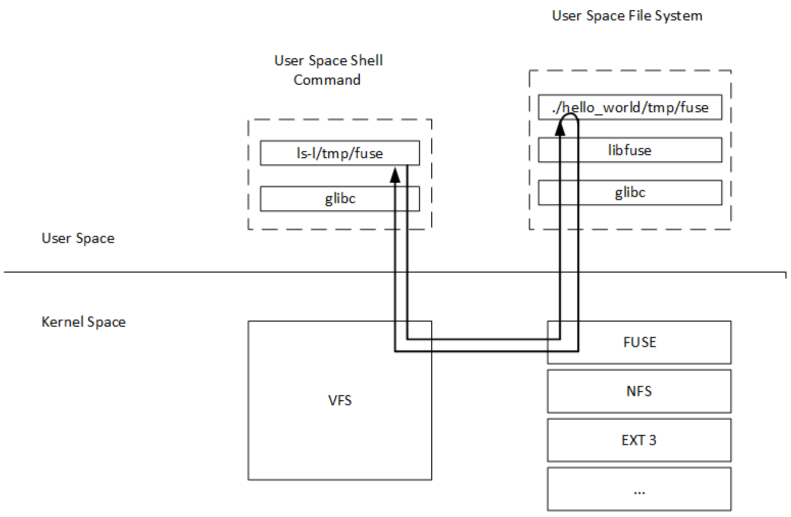


Figure 4.1: FUSE hello_world example

In the Figure 4.1 above there is a simple example of how FUSE works. The ls -l/tmp/fuse command sends a request to the virtual file system (VFS) which is forwarded to FUSE in the kernel and then to the hello_world file in user space. The response follows the same path from the opposite direction. The glibc in the figure is the C standard library and libfuse is the FUSE library.

The FUSE API covers all the methods required for the virtual file system, some examples which were also used in the prototype are the followings:

Filesystem methods

- access: access a file in a specific path.
- chmod: change the access permissions of a file.
- chown: changes the ownership of a file.
- getattr: retrieves the file's extended attributes (uid,gid, access time etc.)
- readlink: retrieves the value of a symbolic link or canonical file name.
- mkdir: creates a new directory in a specific path.

File methods

- open: opens a file.
- write: writes data to a file.
- read: reads data from a file.
- truncate: change the size of a file.
- fsync: synchronize the current state of a file with a storage device.

CHAPTER 5
Context Aware Access Control

Every access control model has its perks and limitations. No out-of-the-box model will work for every organization flawlessly but requires careful planning otherwise, it will have a great impact on security and usability. Our proposition is a Sensor Enhanced Access Control model which combines motion tracking with a traditional logical access control model.

Currently, there are cases where the boundaries between logical and physical access control are not quite distinct. A digital asset is protected with an access control policy, but when the asset is projected to an output device e.g. a monitor, a speaker or a printer, then it is up to physical access control to secure it. Therefore, a context aware system can extend the current physical access control models lessening some of the limitations of existing models and bridge the gap between physical and logical access control.

The developed prototype demonstrates a proposition of a context aware system as an extension to the embedded access control of UNIX which utilizes motion tracking and sensor readings. The current DAC access control model of UNIX is used in which users decide how their content is handled with no further modification upon the logical access control. The prototype focuses on the physical access control which makes it flexible to integrate with any access control policy that may be decided by the organization. In this chapter, the most important components of the system will be described along with an overview of the prototype.

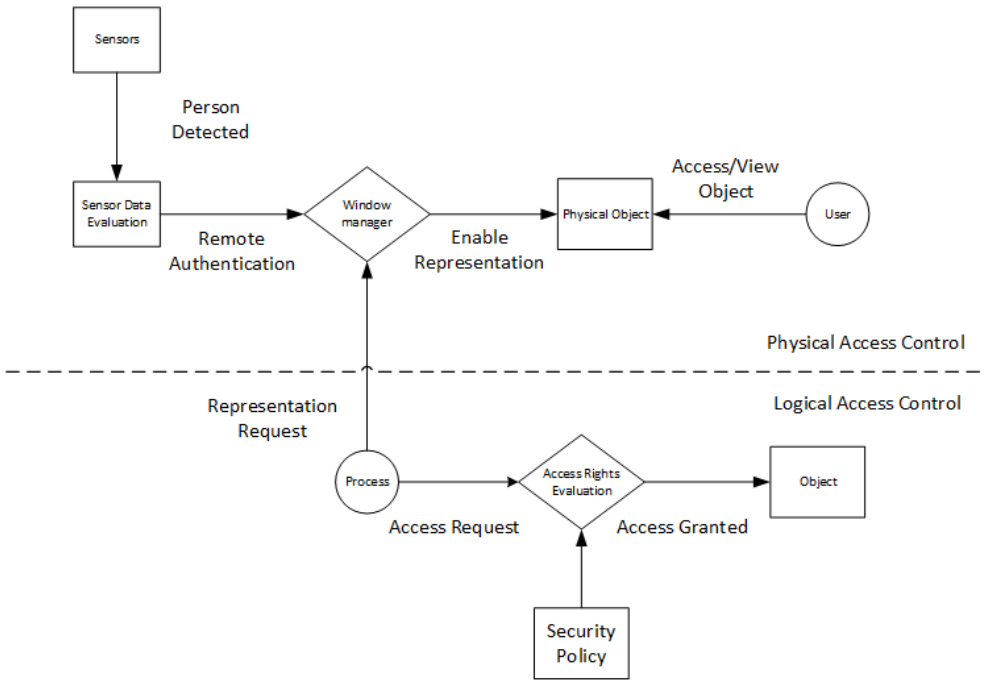


Figure 5.1: Context Aware Access Control

A graphical representation of the model can be seen in Figure Figure 5.1 which is inspired by previous work in [JGW13] and can be divided into two categories logical and physical access control.

5.1 Logical Access Control

Logical access control is handled entirely by the UNIX system using the default DAC policy. The security policy can be replaced any time from the system administrators with no actual impact on the prototype since it requires only the output of the decision made by the system. Regardless of the policy type, there is a set of rules to enforce it which is decided and planned from the administrators. These rules will determine if the system will allow or deny access to an object. The main components of the logical access control include subjects, a reference monitor and objects.

5.1.1 Subjects-Processes

As a subject is considered every active entity that can modify an object. Subjects may vary according to the system; they may be processes running in memory or external users. Even if they may not be physical entities, they represent a user that initiated them. Each user is associated with a unique Id which is granted upon authentication. Each process started from the user inherits the same user Id which is used to evaluate its access rights according to the access policy. Processes invoke all the required tasks in the system and may use various system resources such as CPU, physical memory or data. For every action the process invokes, the system evaluates his authorization levels using the user Id. The access policies will reject or grant access to resources and data. The processes represent the users within the operating system and may indirectly use connected physical objects such as printers.

5.1.2 Objects

As an object can be considered every passive entity that can be accessed and modified by a subject. In an operating system, objects represent files and directories which compose a file system. Every file is bound with a unique inode number which identifies it, while also containing the file's attributes. The attributes contain all the required information used to describe the file or directory e.g. size, type and group. The inode also contains the access policy which is set by the owner of the file and is used by the reference monitor to grant or restrict access when requested.

5.2 Physical Access Control

The second part of the prototype is the physical access control which is responsible for handling the system's output to physical devices. The decisions are based on sensor readings with respect to the logical access control policies. External users, physical objects and a reference monitor represent the physical access control part.

5.2.1 External Users

As an external user is considered every active entity that can access and modify an object. Every user must be authenticated through an initial authentication session in order to interact with the system. Once authenticated, the user is represented by a unique user id which help us assign access rights and invoke processes to carry on the desired tasks. For every user action, the system evaluates the access rights of the specific user id.

5.2.2 Physical Objects

A physical object is every external device connected to the system that represents a logical object. Devices such as monitors and printers convert data to perceivable objects that require physical security. For every represented output the access control changes from logical to physical and additional security measures must be enforced.

For instance, a visual representation of a top secret file on a monitor requires the appropriate access level to be viewed, but once it is projected on the screen, then it is visible to everyone that can see the monitor. Therefore, it is important to control this transition from logical to physical access control.

Windows within a monitor are considered as physical objects since they provide a visual representation of a file or process. Each window is associated with a process and consequently an access level. The prototype analyzes all the open windows along with their inodes, their position on the screen and their size. If an unauthorized user is detected then the reference monitor will secure the appropriate windows by making them transparent or covering them entirely with an overlay window. A similar approach can be implemented to secure other output devices such as speakers and projectors by simply interrupting the representation to users.

5.2.3 Motion Detection & Remote Authentication

Both motion detection and remote authentication provide the contextual awareness to the system. They are categorized as sensors which are responsible to continuously monitor the surrounding environment gathering and analyzing data. The remote authentication can be achieved using a any biometric algorithm through multiple authentication sessions. Motion detection is constantly tracking the users that approach or leave the office.

The outputs of the sensors are combined and evaluated from the reference monitor or window manager. Whenever a user approaching the office is detected then an authentication is required, if the user is not authenticated then he is considered as an unknown user with no access rights thus access is rejected. In our case we consider that an initial authentication session has been established beforehand and follows the user though his entire path. The prototype can be extended with additional sensors according to the organization's security budget and infrastructure.

5.3 Reference Monitor

The reference monitor is responsible to enforce the access control policies on subjects. Each reference monitor has to meet some specifications:

- It must be always invoked thus enforcing complete mediation to the system. Every process is required to have sufficient authorization levels in order to be granted access to the object.
- It must be tamperproof which specifies that it cannot be modified by any process so that no malicious users can modify the security policies.
- It must be small enough to be easily verified that is able to enforce the correct access control policies.

Although, the reference monitor in the prototype which is identified as window manager, does not satisfy all the aforementioned specifications, it meets the basic requirement of handling access requests to the objects. The window manager combines the data gathered from the sensors with the access policy and decides upon the access request. The logical access control policies are handled entirely by the operating system.

As seen in Figure 5.1 when a principal is detected the sensors forward his position and user id to the reference monitor. The reference monitor compares the access rights of the user id with the access rights of the open windows. If the user does not have the required access rights for one of the windows, then the window is hidden as the principal approaches the screen. As the unauthorized principal moves away from the office, the windows are restored back to normal. The window manager can be described with the following algorithm.

Algorithm 1 Window Manager

```
while sensor data do
Track position of principal A
    if A approaches then
        CHECK_WINDOW_PERMISSIONS()
        if A_Permissions < Window_Permissions then
            HIDE_WINDOWS()
        else
            Do nothing
        end if
        if A leaves area then
            REVEAL_WINDOWS()
        end if
    end if
end while
```

CHAPTER 6

Design

The previous chapter was devoted to describe the model of a context aware access control system. The model has various sub components that need to interact and exchange data with each other. In this chapter, the system's components will be described with the visualization of a component diagram. We will analyze each component separately and discuss how they interact with the rest of the system.

The system's functionality is based on the component's ability to successfully exchange data in real time for the security measures to be effective. An overview of how the components exchange data will be presented with the help of sequence diagrams and a discussion why Fuse was chosen as a basis for the prototype.

6.1 System's Overview

An overview of the prototype's architecture can be viewed in Figure 6.1. With the help of the component diagram, there will be a more detailed view of the structure of the context aware access control prototype and how the main components interact. The two main components of the system are the sensor and the reference monitor which include more sub-components that provide various services. Additional components such as the user database and authentication component which provide services to the sensor component.

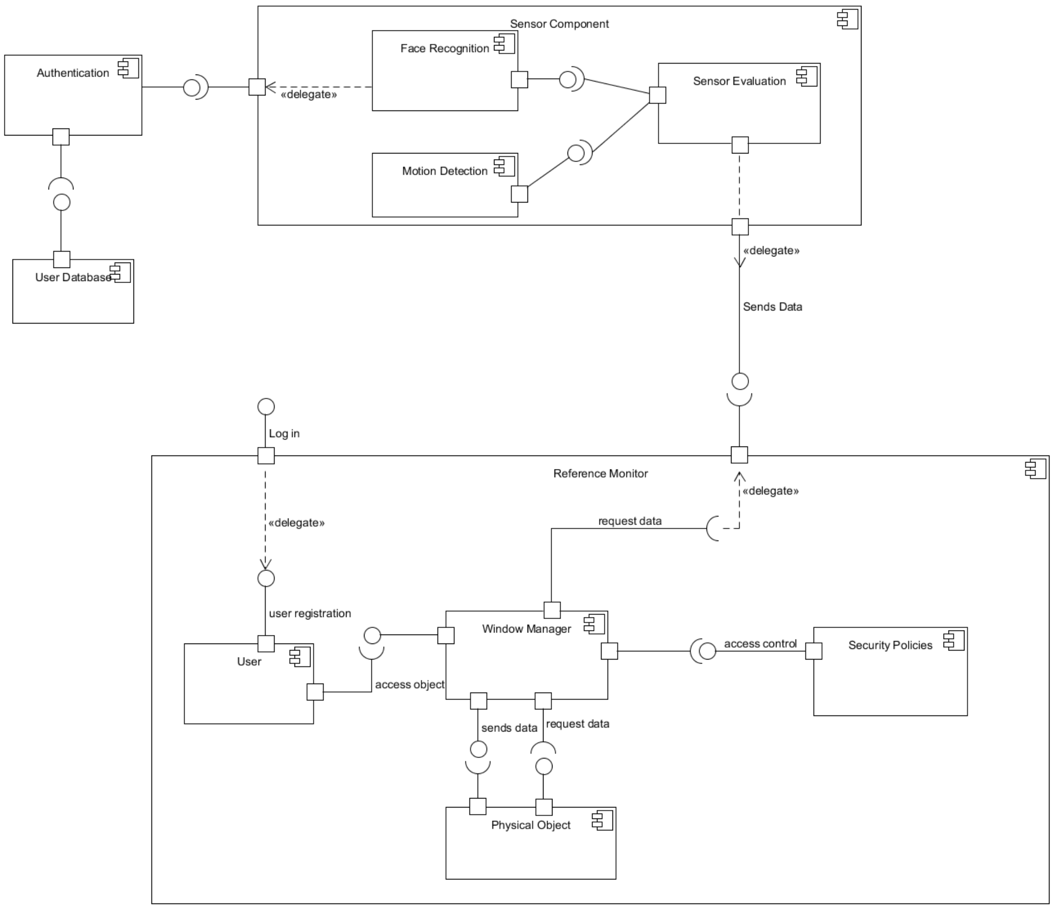


Figure 6.1: Component Diagram

Sensor Component
The sensor component includes the face recognition, human detection and sensor evaluation sub-components. Face recognition and human detection components are responsible for sending data to the sensor evaluation which exchanges data with the reference monitor. The face recognition component has to exchange data with the authentication component which communicates with the user database for the authentication and verification of any detected person. The collected data from the face recognition component include a number of pictures of each face, grabbed from each frame of the live video feed. These pictures will be compared with the stored images of users and attempt to authenticate the approaching individuals.

The database contains a set of images for each employee which will be used for face recognition. The human detection component notifies the sensor evaluation component whenever someone is detected and returns his position from the office. The sensor evaluation must analyze the outputs from the other components and return data to the reference monitor containing the user id of the authenticated principal and his position. In case the principal could not be authenticated then he is marked as unknown and along with his position, the data is forwarded to the reference monitor.

As seen in the Figure Figure 6.1 the client-server paradigm approach was followed for the sensor sub-components. Each sensor is considered a client who provides data to the sensor evaluation which is the server. This allows additional sensors to be integrated in the future making the system easier to scale as the needs expand. The server collects the received data from the sensor clients and evaluates them before passing them on to the window manager. Clients collect and transmit data frequently so that the position and user id of every approaching individual is constantly updated which makes authentication persistent. Introducing more sensors will enhance the performance of persistent authentication and consequently system's performance to successfully enforcing access control.

Reference Monitor
The Reference monitor consists of various subcomponents with the main one being the window manager that communicates with the rest. The window manager is also connected with the sensor component in order to exchange data of the detected principals and decide about the security actions.

User Component
The user component represents the authenticated user currently logged in in a computer station. For every object, the user wants to access whether it is physical e.g. printer or logical e.g. files an access request has to be forwarded to the window manager. The window manager will decide if the request will be granted or not. Each user has a unique user id which is used by the window manager to evaluate the access rights. The authentication and authorization of the user are handled by external components which will not be described since they are part of the UNIX system.

Security Policies
The security policies component is dedicated to enforce and evaluate the access rights of users. The policies are defined from the administrators of the system and are evaluated in every access request. For the evaluation, a user id is required in order to verify what can be accessed by that particular user id. Besides the access control on the registered users, the security policies component evaluate the access levels of the principal detected and authenticated by the sensors. The results in both cases are forwarded to the window manager.

Physical Object Service
As physical object component is denoted every asset which can be accessed from

subjects/users through the window manager after the access rights have been verified and evaluated. Access to the service can be intercepted whenever an unauthorized or unidentified user is detected approaching the office.

Window Manager
The window manager is the main component of the prototype which is responsible to analyze data from various components and make decisions to ensure information security. Every access request goes through the window manager which compares the user id with the access rights in the security policies component. In cases when an external user is detected, the sensor component returns the user id of the detected principal and his current position. The user id is evaluated and compared with the access policies required to access the physical object. If the user id does not have the required access rights, then the output of the physical object e.g. monitor is secured as the principal approaches.

The same method is followed for the principals that have not been identified. Although, failure to identify the approaching individual might also be due to insufficient illumination in the room or not having a clear view of the face. This is where persistent authentication comes in place. The authentication sessions are repeatable over short times which is a requirement to keep track of each person's position as they approach. Therefore, if authentication fails for three out of 5 sessions until the person is close enough to the office, then he has still been identified. If all the authentication sessions fail, then he is not employed in the company thus an intruder or a customer. In this case, the output of the physical object is completely blocked by default. Communication between the sensor component and the window manager is crucial and it should occur in real time.

6.2 Interactions Between Components

In the previous chapter, there was an overall description of the system and the main components. In this section, there will be a more detailed analysis using sequence diagrams on how these components interact and in what order. The sequence diagrams demonstrate various interactions in the system under different scenarios along with what messages are passed through the components.

6.2.1 User Authentication

As seen in Figure 6.2, whenever an individual is detected the motion sensor notifies the face recognition module of a person being in range. The face recognition module sends a number of pictures along with the person's current position to the authentication module in a first identification attempt. The authentication module will search for a match in the user database if the individual is identified then the user.id and position is returned to the sensor evaluation component. The authentication component returns the user.id to the face recognition module as a confirmation of the authentication request. The user.id is also forwarded from the face recognition module to the motion detection module to mark the person with the particular user.id. The motion detection module performs frequent updates of the user's position which will trigger the authentication request again to enforce the persistent authentication. The sensor evaluation receives every new position of the detected user.

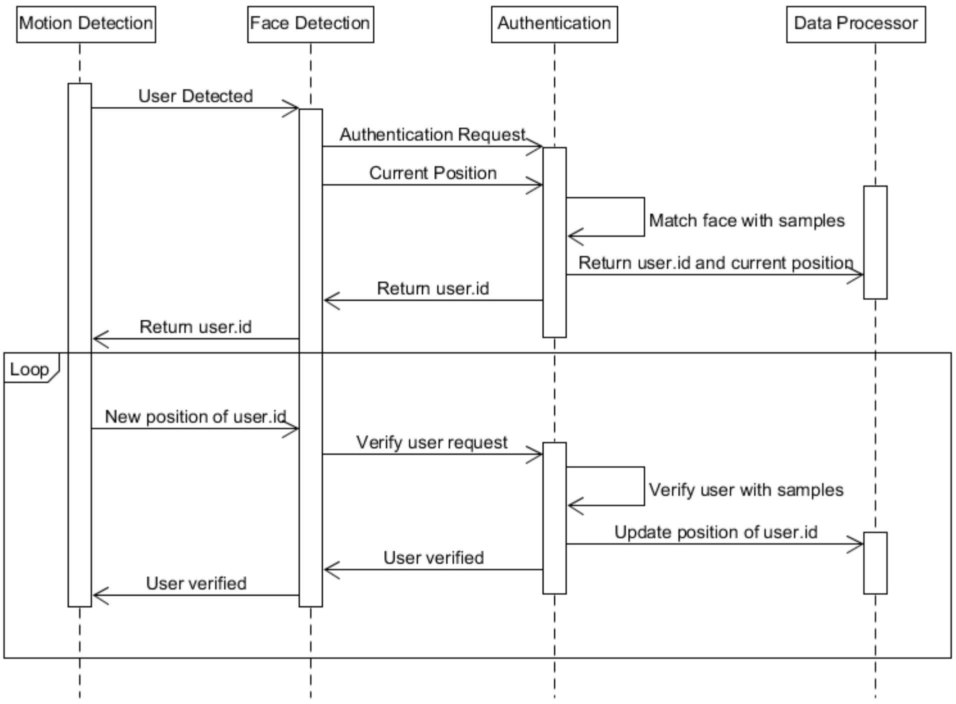


Figure 6.2: Sequence diagram showing the interactions of the sensor components

An alternative scenario would be if the authentication component was not able to identify the detected person. In that case, the person is marked as unknown which is returned to the other modules instead of the user.id.

6.2.2 External User Approaching

In Figure 6.3 is the sequence diagram of the system's behavior when an approaching user is detected.

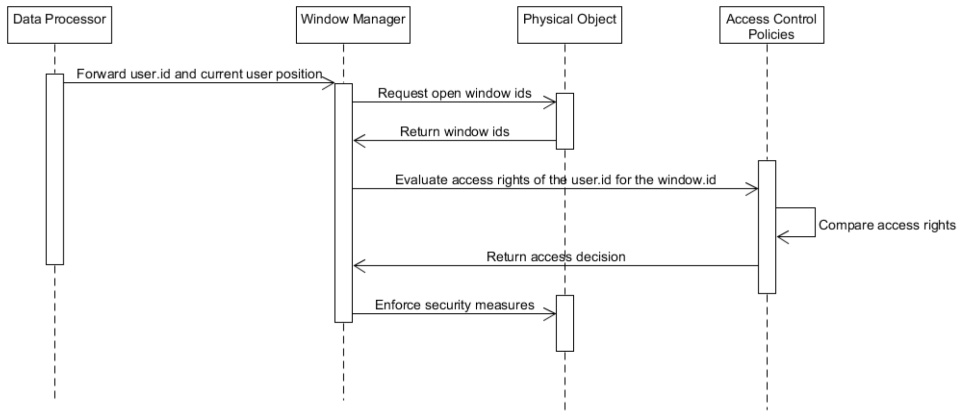


Figure 6.3: Sequence diagram showing the interactions when an intruder approaches

The sensor evaluation sends the user.id and position of the detected person to the window manager. The window manager sends a request to the physical object module to acquire the window.id of all windows currently open. Once the module responds the window manager forwards both the window ids and the detected user id to the access control policy module to verify the required access rights. Once the access control module compares the access rights of every window.id with the user id, it returns the result to the window manager. The window manager proceeds to enforce the security policies for every window that requires higher access rights than the user's while the rest remain intact.

If the sensor evaluation does not forward the user.id, then the approaching user is considered as unknown, therefore, the window manager enforces the security policies directly when he comes in range.

6.3 Window Cover

The proposed security measures to ensure data confidentiality for all opened windows of a monitor include two different approaches while also having in mind usability. Both propositions aim to gradually reduce window visibility when someone approaches the office cubicle. This way it also provides a warning to the user currently viewing the files that someone is in range instead of instantly enforcing the measures causing inconveniences.

Each implementation follows a similar approach that requires each window id to be extracted and evaluate the access rights. The first implementation gradually modifies the opacity of the window until it becomes completely invisible. The second one creates an overlay window that covers the desired windows entirely. The overlay window will have different background colors and opacity levels which will change till they completely hide the window behind it. The opacity and color changes for both implementations will vary according to the distance from the office and monitor of the approaching person.

The implementations were tested on Ubuntu-16.04.1 distribution, but their implementation is not depended on the Linux distribution. In fact, the overlaying windows could be easily integrated into every operating system (Windows, Mac OS) if the window privileges are passed correctly.

6.4 FUSE

The prototype resides in a virtual file system which is mounted on demand over the existing file system. The choice of virtual file system was made solely because of the advantages it provides in flexibility and its support of various file systems. Fuse provides an ideal interface to create a virtual file system which will make the prototype more stable since it operates on User space thus errors are easier to handle without crashing the entire system. Additionally, Fuse's versatility enables the possibility to integrate additional access control policies and handle files in terms of security as the company prefers.

CHAPTER 7

Implementation

This chapter is dedicated to present the implementation of the prototype along with the libraries and algorithms used to achieve it. We start by a simple discussion about the virtual file system and some decisions that were taken. Then the motion detection and user tracking implementation is presented along with how these define the safe and critical zones regarding the distance from a monitor. The rest of the sections describe the core of the prototype and more specifically the window manager, how the access rights are retrieved and handled and the two implemented solutions to preserve the confidentiality of the monitors' output. All the implementations were made with Python 2.7.

7.1 The Virtual File System

The VFS was implemented using the fusepy [Ver] library that provides all the necessaries to construct an API which was used for the implementation of the virtual file system. All the necessary actions of a file system were included and the methods were organized in file methods (i.e. open, create, read) and filesystem methods (i.e. chmod,rmdir,mkdir). The method getattr is called first at least once before any other method is executed, to verify that the file or the file path exists.

Before mounting the virtual file system, there is an option to allow access to the files from external users. By default FUSE allows only the root user to access the mounted files unless we change the allow_other options to True. As root is considered the user who mounted the virtual file system.

7.2 Tracking and Motion Detection

The requirement is that an approaching person should be detected and his position must be tracked through his path. Therefore it is crucial to have real time detection and notification in order to trigger the security controls in time. For the implementation, we used existing computer vision algorithms and techniques with the help of the Open Source Computer Vision Library (OpenCV) [Its15] for Python which is an open source C++ library. The library is ideal for real time applications since it takes advantage of the multi-core processing and if the required hardware is detected, it supports hardware acceleration.

For motion detection, we used the background subtraction method which is a common technique widely used in various computer vision projects. While there are quite a lot of different algorithms and techniques for motion detection and some may be in fact better in performance, we decided to follow this approach since it suited better to the prototype in terms of simplicity and performance.

As it was already explained in section Section 3.3, generally what we need is to model the background and then detect any changes that occur to it during the motion of the subject. The background for the majority of the frames in the video remains the same, therefore as the subject moves the changes in the background can be used to detect this motion and the subject's position in the frame. Although, in case of minor illumination changes then there is also difference in the background which is detected as motion.

The implementation begins with modeling the background of the first frame of the video. This will be the base which we will compare with every new frame detecting any differences and consequently motion. The differences are calculated with a simple subtraction of the model frame and every new frame. An example of the algorithm in use can be seen in Figure 7.1 and Figure 7.2 with different type of camera angles.

$$\Delta = model\_background - frame$$

In order to solve the problem of minor changes in the shadows and illuminations, we have set a threshold parameter for delta. For the prototype the parameter is set to 40 which evaluates every subtraction and compares it with the threshold, if delta is lower than 40 then any differences are ignored, else highlight the area with white. The exact number of the threshold varies according to the position of the camera and the room, hence some calibrations will be required before deploying the system.

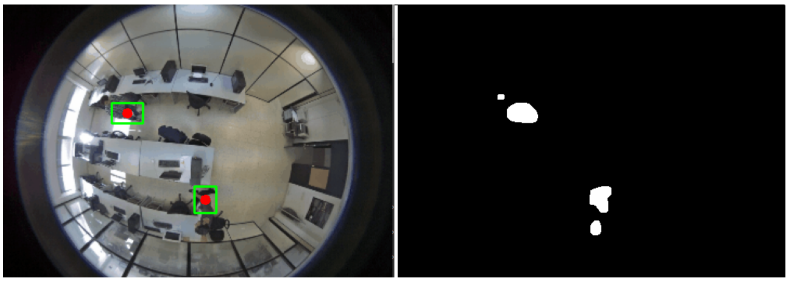


Figure 7.1: Detection in fisheye camera lenses

Once we have the moving object highlighted we just loop through every contour to mark the outlines of the highlighted objects. The contour areas are enclosed in a bounding rectangle and are indicated by a red dot. To minimize the false positives we set another filter for the rectangles, only the detected rectangles that are bigger than 20 pixels width and 20 pixels height are taken into consideration; the others are ignored. The same rule applies for the values as previously, meaning the numbers may vary for different offices and cameras.

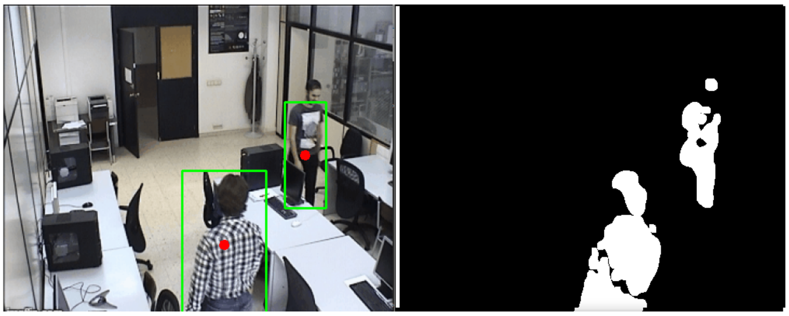


Figure 7.2: Detection in stationary cameras

The bounding rectangle will follow the subject through his entire movement and for as long as it meets the requirements we have set. As we have already discussed, the subject's distance from the point of interest is important for the security measures. While there are a couple of solutions to calculate the distance of the camera lens from an object with known measurements most of them are not universal solutions because camera angles differ from fisheye cameras with ultra wide-angle lens to stationary cameras thus different approaches are required to calculate distance.

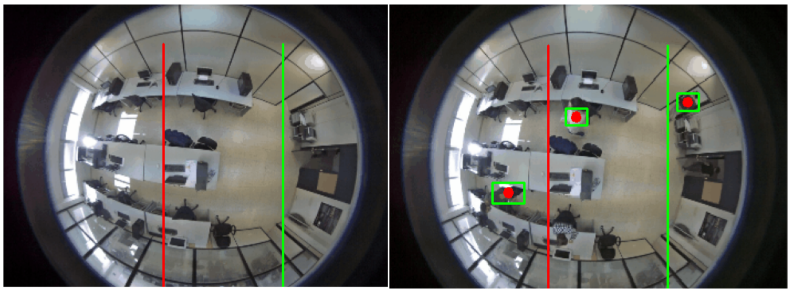


Figure 7.3: Distance estimation in fisheye camera lenses

The prototype provides a universal solution in which instead of calculating the distance of an object, it monitors when the object passes a specific point in the video. More specifically, we have declared two imaginary lines which are drawn on the video, each with different color and at a specific distance from each other. These lines are set according to the safe distance from the monitor which are visible in Figure 7.3. Whenever a subject crosses one of the lines, then we return a notification and the subject's position. Each line represents the distance to trigger a security event. The lines are drawn with the help of the OpenCV library at specific x,y values. In Figure 7.4 because of the camera angle and the office plan we only have one line. If the subject passes the door, then he is already tracked by the system. Additionally, as seen in the picture, the distance from the first monitor is relatively short from the door hence why only one line is required.

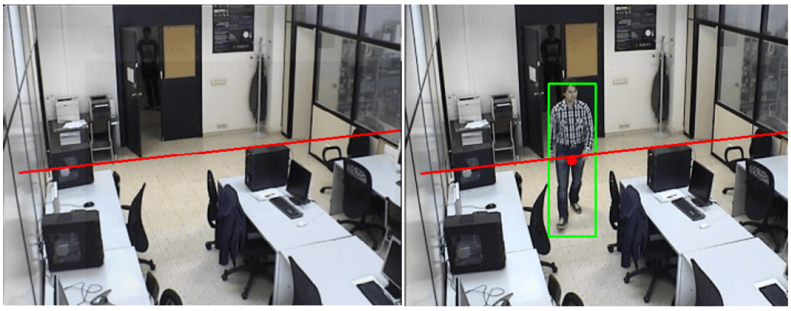


Figure 7.4: Distance estimation in stationary cameras

The functionality of the motion detection could be summarized in the following algorithm.

Algorithm 2 Motion Detection & Distance Estimation

```
1: while first frame is True do
2: Retrieve the current frame convert it to grey-scale and blur it frameDelta = first
   Frame - current frame
3:     FIND_CONTOURS WITH THRESHOLD $> 40()$
4:     for every Contour do
5:         if width>20 and height>45 then
6:             Draw rectangle around subject a and track him
7:             if a crosses green line then
8:                 raise security warning
9:             end if
10:            if a crosses red line then
11:                raise security event
12:            end if
13:        end if
14:    end for
15: end while
```

7.3 Window manager

The window manager is the component responsible for handling the open windows and access rights while also evaluating the output of the sensors. The primary function of the component is to identify which files are currently open from the user, their position on the screen, their actual size and what access rights are required to view them. The implementation is performed with the help of Pygtk [PyG11] and wnck [Lib] libraries.

Although, Pygtk's primary function is to provide the means for a graphical user interface, for this component we will make use of its functions to identify events and refresh the interface. Every window action is a separate event e.g. when a new window is created, when it is moved around the screen or minimized and it is important to refresh the variables containing the information about it. More specifically, the whole process is performed in two functions the first being the events_pending() which monitors the system for any window events and the second one is main_iteration() which loops through the events. If no events occur then it remains inactive until one is triggered. As a side note, the functions are also crucial during the initial execution of the program otherwise the first open windows are not identified.

The second library, wnck provides the mechanisms required for window management. The first step is to identify the default screen and pass it to a variable, once the screen is identified then we identify all the open windows in the session. It is important to have a small delay between the screen identification and the windows otherwise the list of windows will always return zero. We iterate through each identified window and retrieve each window's geometry including its position in the screen saved in x,y values and its width and height in pixels. Additionally, we also keep the window id and the title. Because the retrieved title is the full title of the window including the application's name e.g. test text.odt - LibreOffice Writer we split the returned string and check if it is a file, an application or a directory. If it is a file or a directory, then we utilize the UNIX stat command to return all the identifiers of the file such as the st_mode for the permissions or st_ino for the inode number, etc. These will be used to evaluate the access rights of every open file with the access levels of the identified approaching subjects.

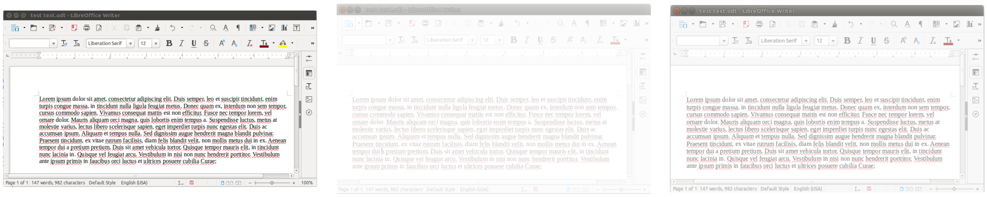
Since the existing file permission system of Unix is used, the access rights are set for files and directories and not applications like Firefox thus the implementation focuses only on handling file and directory windows with access rights. To retrieve the access rights first, we verify that the file exists in the system which allows us to differentiate between files and applications while also retrieving its file-path. To find the file path we have introduced a function which walks through the directories included in "/home" until it matches the given file name and returns the complete path of the file. This approach was preferred because it provides an ubiquitous and flexible solution to identify file paths regardless of the directory they are nested, retrieving their access rights. Additionally, it is capable to handle file names that contain special characters (i.e. <, >, | , :, (,), &) or white spaces. The permissions are extracted from the output of the stat() function and more specifically the st_mode field which contains the file permissions in octal notation (i.e. 0755, 0222 as per Unix Numeric Notation). We decided to follow two different approaches to secure windows and let users evaluate it concerning usability and effectiveness. The two methods are window transparency and overlays. Both allow a calm transition to the security feature disrupting user's work as little as possible, instead of instantly unmapping the working window

7.3.1 Window Transparency

The recent versions of Ubuntu are not flexible enough regarding window customization and especially transparency. The current window manager in Ubuntu is Unity, and the only window modifications it allows through scripting are changing the CSS files directly for each desired theme. Using compositing or window managers such as Compiz were neither reliable enough or stable for the system causing it to crash and did not allow any scripting actions. Additionally, forcing the user to change his user interface altogether is not a viable solution. For this purpose, since we could not find (to our knowledge) any Python based solution that covered usability and scripting to provide window transparency, an open source software was used named Devilspie2.

Devilspie2 is described as a window matching utility written in C++ and LUA. The software allows scripting actions or rules that are defined in a LUA file stored in a folder. Actions include resizing windows, moving them to specific coordinates, changing window opacity, etc. The structure of the rules follow the LUA notation and are applied to a given window or a number of windows passing their XIDs or other identifiers like window or application name. By default when Devilspie is called it searches for the file containing the rules in the /.config/devilspie2/ folder. For every window that is required to be hidden, its XID is passed to a Python list and is appended to the LUA file containing the rules. The rules are simple since the only requirement is to identify all the XIDs of the open windows and look which one of these are included in the list we have appended. The opacity rules are applied only for the matching XIDs, and the values vary from 1.0 being completely opaque to 0.0 being completely invisible. The opacity levels can be applied gradually over time or instantly according to approaching subjects position.

The following pictures present the effect of different opacity levels on the same text and application window (Libre Office).

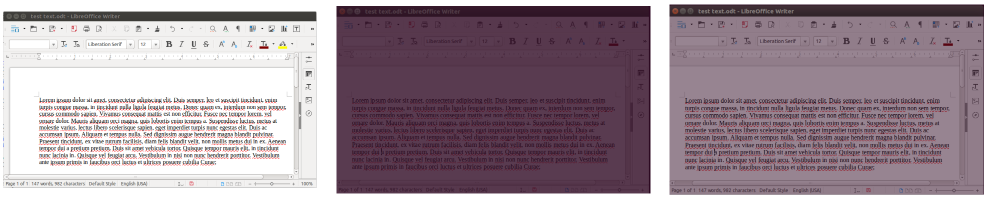


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Page 1 of 1 147 words, 982 characters Default Style English (USA) 100%

Figure 7.5: Windows with 0.2 (middle) and 0.5 (right) opacity levels with on white background

In Figure 7.5 the opacity rules are applied on a Libre Office document which is located on top of the Documents folder with white background. In Figure 7.5 is shown the same window in front of the desktop image which has a purple background. The applications behind the window we need to hide and their background color, change the visibility levels significantly.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Figure 7.6: Windows with 0.2 (middle) and 0.5 (right) opacity levels with on purple background

7.3.2 Window Overlays

Overlaying windows offer a more flexible approach regarding Linux distributions and operating systems in general. The implementation is purely in Python 2.7 with the Pygtk and wnck libraries providing the main functions. Instead of tweaking Ubuntu's Unity window manager or relying on external applications, we create a number of interface windows according to the files and directories that are currently open. The software scans all the open windows and gathers all the appropriate information such as the window size, position on the screen, the name of the open application, etc. Then we proceed to evaluate the access rights required for the window and compare them with the approaching subjects. For the windows the approaching subject is not authorized to see, we create an overlaying window which changes its opacity levels as he approaches the object.

After retrieving the information with Pgtk and wnck as we described earlier the output includes all the identified windows including the windows created from the system on start-up. In Table 7.1 is an example of all the identified windows including the ones from the system e.g. XdndCollectionWindowImp, unity-launcher, Hud which we need to distinguish from the windows created by the user. As we see the system windows have slightly different x,y position values than the others, therefore, when identifying and looping through them we only keep these with positive x,y values.

Window ID	**PID**	**(x,y) position**	**(width,height)**	**Window Name**
46137346	1668	(-1450,-755)	(1350,655)	XdndCollectionWindowImp
46137349	1668	(0,24)	(65 ,631)	unity-launcher
46137352	1668	(0,0)	(1350,24)	unity-panel
46137355	1668	(-420,-300)	(320,200)	unity-dash
46137356	1668	(-1060,-164)	(960,64)	Hud
37748746	1766	(0,0)	(1350,655)	Desktop
29360138	2011	(569,172)	(722,410)	asd@ubuntu: /Desktop/python
62914983	8992	(65,24)	(1285,631)	test text.odt - LibreOffice Writer

Table 7.1: Window Information Output Example

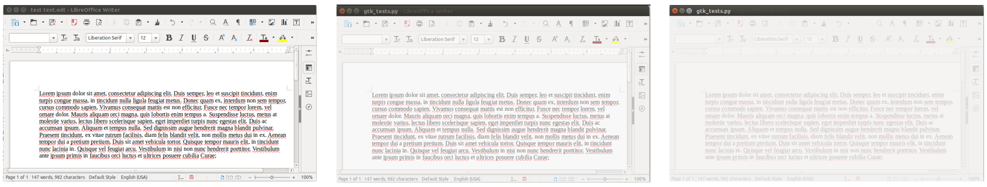
The width and height will be used to create the overlaying windows in order to cover completely the user created windows including the title bar. Although, the returned numbers are slightly wrong for the height values because of the decorations surrounding it i.e. the title bar. The surrounding decorations are defined from _NET_FRAME_EXTENTS(CARDINAL) which includes all the values for left, right, top and bottom frames. By default the values are as follows:

$$\_NET\_FRAME\_EXTENTS(CARDINAL) = 0, 0, 28, 0$$

with the top bar being 28 pixels high. The heights returned from wnck are 28 pixels higher than the original sizes. Therefore, we need to adjust the heights by subtracting the 28 pixels from them before creating the window.

The overlaying windows are created with pygtk and are mainly graphical user interface windows whose opacity levels and background color change according to the parameters we set. The window we need to cover is indicated as parent window and the overlaying as a child window. Both father and child windows are bound together so that when the parent window is closed or minimized, the child window mimics that particular event. The child windows must not interfere with the actions of the user in the parent window, allowing the user to click through the child window and passing the event to the buttons of the father window. In fact, instead of drawing the windows on a pixmap which is defined as a rectangular array of pixel color values, the windows are defined as regions specified by a rectangle. These regions listen for events whenever the mouse pointer is inside the region and passes the click events to the window bellow it. Examples of the overlaying windows can be seen in the following images:

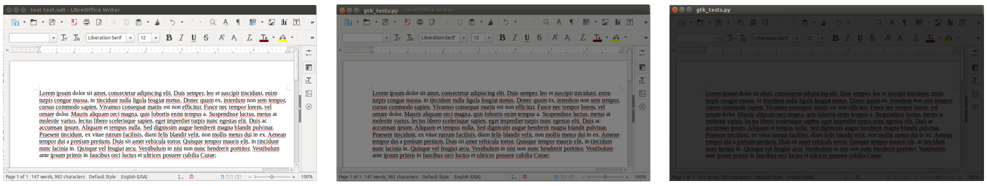
In Figure 7.7 we can see the different opacity levels and how the view is changed. The first picture from the left is a normal window with no opacity levels, the middle represents an overlay window with 0.5 opacity and the last picture with 0.8 opacity level.


test text.odt - LibreOffice Writer

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Page 1 of 1 | 147 words, 982 characters | Default Style | English (USA) | 100%

Figure 7.7: Overlaying windows with 0.5 (middle) and 0.8 (right) opacity levels with white background


test text.odt - LibreOffice Writer

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Page 1 of 1 | 147 words, 982 characters | Default Style | English (USA) | 100%

Figure 7.8: Overlaying windows with 0.5 (middle) and 0.8 (right) opacity levels with black background

In Figure 7.8 we see the difference the background color makes in terms of visibility, while the opacity levels are the same as in Figure 7.7.

CHAPTER 8
Evaluation & Experimental Design

The evaluation will be based on the prototype's ability to successfully provide the basic security criteria of an Access Control system. Starting from the virtual system itself, we have to make sure that it behaves like a traditional system, providing all the required functions, while also maintaining the predefined access policies. Various users will be created with different access levels and we will proceed to set up and access files of the same, higher and lower access rights.

As part of the virtual file system, the window manager should also be tested thoroughly whether its behavior is the one it is intended to be. The window manager should be able to grab the correct access rights of all open windows, analyzing what windows are open and which one is focused. The tests will be conducted with various applications, both individually and in numbers. The windows will differ from Terminal Windows to File Explorer and Text Editors.

In addition, the window manager should also be tested for its ability to enforce window transparency or window overlays, according to users' authorization upon their entrance or when they leave the room. Trying to keep a balance between security and usability, various users were tested both for their ability to read information when the window manager is active and its impact on the authorized user's work.

One critical component of the system is tracking of the identified users. Users are assumed to have been identified upon entering the building and their identity is known through the entire process. Their user.id is required to determine their access levels before reaching the visibility zone of the office. The sensors used to provide the foretold features should be able to successfully pass the information about the approaching users to the window manager. Therefore, various scenarios should be considered to test its performance.

8.1 Access Control in the Virtual File System

The prototype is based on a virtual file system mounted on a traditional Unix system to provide the desired features. To ensure that the core actions of a file system are met, functional tests were used for the virtual file system, e.g., open file, create and delete files. The tests include users and files of various access levels in order to demonstrate the system's ability to maintain and enforce the UNIX MAC when performing actions as the ones mentioned earlier.

Other parts that were tested include user actions, such as changing or modifying other user's files and changing the access rights of files. The tables of the conducted tests can be found in Appendix A.1

8.2 Window Manager

As part of the evaluation, the window manager must be able to identify the access levels of each open window. The test cases include a number of open windows at random positions on the screen. Every window has a specific ID in Hexadecimal digits. These digits are used to identify the required access rights and window's information, e.g., position and size. The importance of this is to evaluate the prototype's capability to retrieve the access rights of the windows and compare them with the approaching user. In the Table 8.1 there is an example of the expected output of the access levels.

In addition to the window rights, the size in x,y and position on the screen must also be identified correctly.

XID	**(x,y) position**	**(width,height)**	**Window Name**	**Access Rights**
79691792	(65,24)	(1301,647)	Mozilla Firefox	-
41945382	(306,24)	(890,494)	Documents	755
58720725	(724,45)	(642,406)	test text.odt - LibreOffice Writer	664
54526084	(65,24)	(1301,647)	foo.txt	740

Table 8.1: Transparent Windows User Testing

An example of the size and position output could be seen in Table 8.1, in which the first window is in full screen, the second one has a medium size and it is located in the middle of the screen and lastly the third one is dragged to the right of the screen and is the smallest of the three. For this test, in order to include an element of randomness, the windows were in various positions and the process included different text editors and applications. The x,y location of the window is necessary for both implementations. Keeping track of the stacked windows behind the main one when it becomes transparent, will prevent unwanted windows to be revealed. As for the overlays, the x,y positions alongside with the height and width of the window to be covered, create the overlaying window.

To determine which one of the solutions, regarding the hiding of windows, was better, a number of people who claimed to have a 20/20 or corrected to 20/20 vision, were asked to test the application. For both implementations (window opacity and window overlay) the users were asked to read a specific text from a screen. All the reading tests were timed and the times were compared to see which solution had the greatest impact on readability. In addition to the time comparison, a grading scale system was introduced to help users grade how easily they could read the texts. The grades used, were a 1–5 point system as the following:

1. Text was unreadable

2. Text was difficult to read

3. The text was not difficult, but it took some time

4. Text was easy to read although, it did not feel perfectly natural

5. Text was easy to read, it felt perfectly natural

The texts used were 100 words each with roughly the same reading level of 6-8 which, according to the Dale–Chall readability formula [DC48] is easy to read and comprehend by an average 9th or 10th-grade student. The text was in Liberation Serif font with 12-point font size. For every user there were random texts each time, to make sure nothing was out of memory. While timed, users were asked to read the first text from 1.15 meters distance under normal circumstances. The specific distance was preferred after tests and user feedback and will be discussed later on. Then they were asked to read a random text for two different opacity levels of the two implementations. For transparent windows, the opacity levels used were 0.5 and 0.2, with 0 being totally transparent and 1 no transparent at all. For the overlay windows the opacity levels were 0.8 and 0.5. with 1 being a fully colored overlay window and 0 completely transparent.

User	Distance	Opacity 0.2 1-5 grade	Opacity 0.5 1-5 grade	Opacity 0.2 Time	Opacity 0.5 Time	No opacity Time
1	1.15m	2 out of 5	3.5 out of 5	83.712 seconds	48 seconds	45 seconds
2	1.15m	3 out of 5	4 out of 5	66.438 seconds	48.51 seconds	46.71 seconds
3	1.15m	2.5 out of 5	4 out of 5	79.764 seconds	56.69 seconds	49,62 seconds

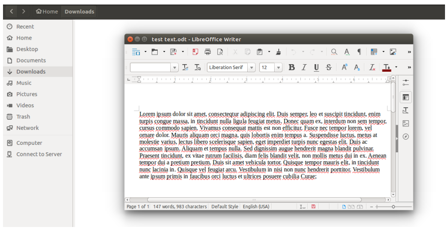
Table 8.2: Transparent Windows User Testing

Some of the results could be seen in the table Table 8.2 and Table 8.3 for a detailed table please refer to the appendix

User	Distance	Opacity 0.5 1-5 grade	Opacity 0.8 1-5 grade	Opacity 0.5 Time	Opacity 0.8 Time	No opacity Time
1	1.15m	3.5 out of 5	4.5 out of 5	49 seconds	45 seconds	45 seconds
2	1.15m	4,5 out of 5	4 out of 5	40 seconds	47,39 seconds	46.71 seconds
3	1.15m	5 out of 5	4 out of 5	38.16 seconds	42.60 seconds	49.62 seconds

Table 8.3: Overlay Windows User Testing

In both cases, there was an actual impact on the readability of the texts and in both cases, the background color could lower or raise the readability of a window. For instance, in the first case, when the opacity of a window was decreased it became completely unreadable if the window behind it included other files or directories, as seen in Figure 8.2. Similarly, as seen in Figure 8.3 and Figure 8.4, visibility varies according to the background.



Lorem ipsum dolor sit amet, consecteqtur adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Figure 8.1: Normal Window

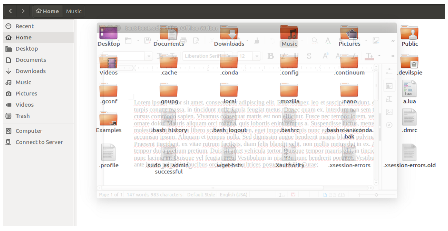


Figure 8.2: Window Transparency with 0.2 Opacity with Folders in the Background

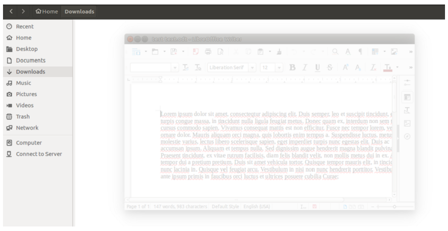


Figure 8.3: Window Transparency with 0.2 Opacity with White Background

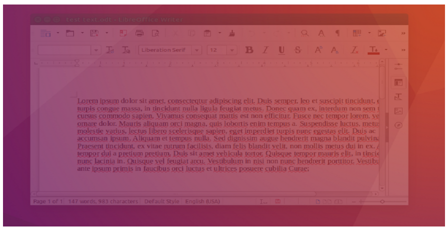
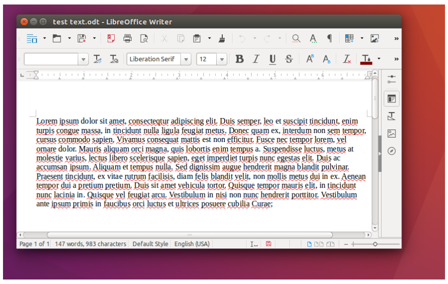


Figure 8.4: Window Transparency with 0.2 Opacity with Desktop Image in the Background

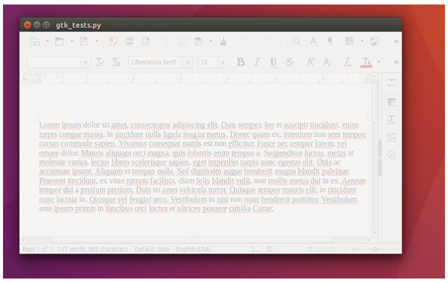
On the other hand, for the overlay windows the readability was increased if the covering window had a white background color (Figure 8.6) and the user was moving further away, but as he closed by, the text became blurry. Furthermore, the users reported that a dark colored overlay window (Figure 8.7) helped to increase the visibility in close distances, but it was reduced dramatically when you moved further away. The distance of the approaching user is a key element to the visibility and ideally, the overlaying windows should have different background colors, but the test users reported that the frequent color changes while working affected their work, which made it more annoying than useful.



Lorem ipsum dolor sit amet, consecteqtur adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Page 1 of 1 | 147 words, 983 characters | Default Style | English (USA)

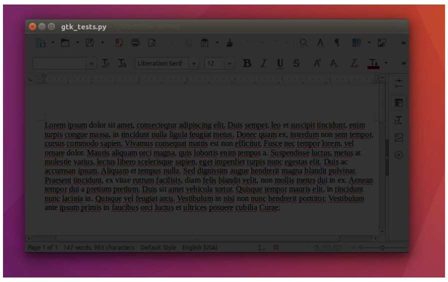
Figure 8.5: Normal Window



Lorem ipsum dolor sit amet, consecteqtur adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Page 1 of 1 | 147 words, 983 characters | Default Style | English (USA)

Figure 8.6: Window Overlay with 0.8 Opacity and white Background

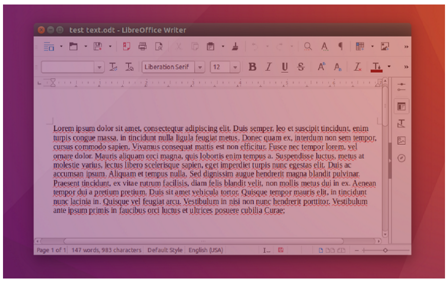


Lorem ipsum dolor sit amet, consecteqtur adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Page 1 of 1 | 147 words, 983 characters | Default Style | English (USA)

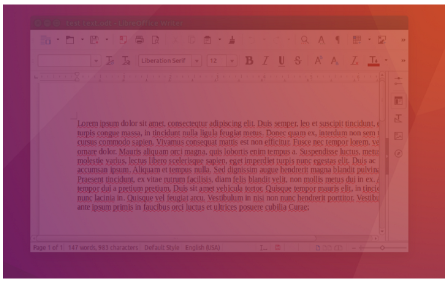
Figure 8.7: Window Overlay with 0.8 Opacity and black Background

In the following figures, there are examples of how the window readability was tested for both implementations. The opacity levels were chosen in such way that for the first test they would provide a slight change to the window and for the second test the window should become almost invisible. The first two figures present a transparent LibreOffice Writer window with 0.5 and 0.2 opacity levels with the default background image of Ubuntu. The bottom pictures present the overlaying windows with 0.5 and 0.8 transparency levels while having a white background.



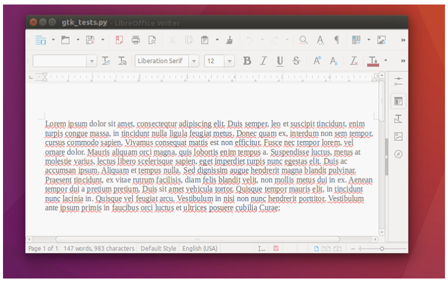
Lorem ipsum dolor sit amet, consectequr adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Figure 8.8: Window Transparency with 0.5 Opacity



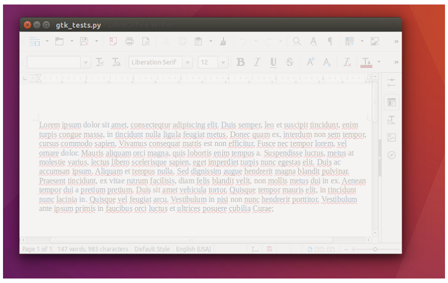
Lorem ipsum dolor sit amet, consectequr adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Figure 8.9: Window Transparency with 0.2 Opacity



Lorem ipsum dolor sit amet, consectequr adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Figure 8.10: Window Overlay with 0.5 Opacity



Lorem ipsum dolor sit amet, consectequr adipiscing elit. Duis semper, leo et suscipit tincidunt, enim turpis congue massa, in tincidunt nulla ligula feugiat metus. Donec quam ex, interdum non sem tempor, cursus commodo sapien. Vivamus consequat mattis est non efficitur. Fusce nec tempor lorem, vel ornare dolor. Mauris aliquam orci magna, quis lobortis enim tempus a. Suspendisse luctus, metus at molestie varius, lectus libero scelerisque sapien, eget imperdiet turpis nunc egestas elit. Duis ac accumsan ipsum. Aliquam et tempus nulla. Sed dignissim augue hendrerit magna blandit pulvinar. Praesent tincidunt, ex vitae rutrum facilisis, diam felis blandit velit, non mollis metus dui in ex. Aenean tempor dui a pretium pretium. Duis sit amet vehicula tortor. Quisque tempor mauris elit, in tincidunt nunc lacinia in. Quisque vel feugiat arcu. Vestibulum in nisi non nunc hendrerit porttitor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Figure 8.11: Window Overlay with 0.8 Opacity

The majority of testers reported that window transparency was more effective to hide windows than the overlaying window. As an addition to the first tests, the users were also asked to find a specific word in a random passage for both implementations and under normal conditions in terms of room lighting. Their attempts were timed once more and even though the difference was not significant, their performance was worse for the transparent windows. The distance was one of the most influential parameters and according to users, the readability zones could be divided into three categories. For distances up to 1 meter from the monitor, the majority of users could read everything with no particular difficulty, unless the window was almost invisible or covered. For distances from 1.15 to 2 meters, the readability was highly influenced from the window's visibility. After 2 meters very few could read the passage even without any active implementation. Therefore, the ideal distance to activate the security feature is once the intruder passes the 2 meters. Of course, the safe distance will vary between monitor and font sizes, but as a basis, the 2-meter border is a realistic distance to trigger the security features.

To determine whether or not the implementations would reduce usability and become yet another annoying security feature, random users were asked to type a random passage, while the implementations were active and once when they were inactive. The majority of them thought windows being transparent while working was interfering a lot with their work. The overlaying windows, on the other hand, offered more flexibility until the focus window was not visible at all.

The feedback received from the users is purely based on their personal preferences e.g. font size, background colors that felt more comfortable and the environmental conditions, e.g. screen brightness, room illumination. Consequently, before deploying any of the implementations, we have to adapt it to the working environment and the authorized user accordingly. While the readability of the passages was relatively easy to test, this is not the case with pictures and images. The amount of information an external user can gain from an image depends on many parameters that could not be predicted. An image of a blueprint, for example, with white lines on blue background could easily be seen from long distances, without taking under consideration cases where the authorized user zooms in.

8.3 Identification and Tracking

Both implementations are based on the system's effectiveness to identify and track individuals. Since the focus of the thesis is not the evaluation of the biometric experts, it is assumed that they function as intended while passing the correct data. The subject is assumed that is authenticated when he enters the visibility zone thus his access levels are known. To check the security features of the implementations, functional tests were used and different cases, such as two persons being detected in the room or one person not having the required permissions. For the tracking system, it is assumed that the tracking can follow users through their entire path, while also remain authenticated. Even though tracking an individual throughout his entire trajectory is out of scope, it is still a requirement to detect whenever someone leaves or enters the office and handle it accordingly. The tests were focused on how the system handles detected persons, while they close by of leave the visibility zones.

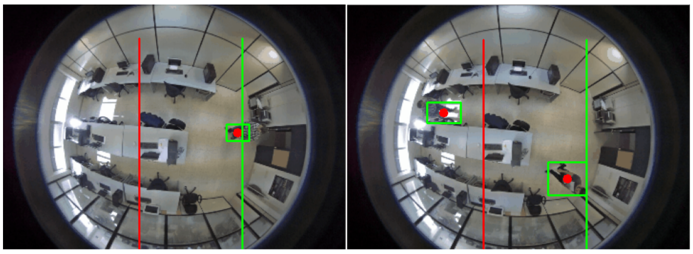


Figure 8.12: Office test

In Figure 8.12 there are two visibility zones with their boundaries drawn. The tracking system must be able to identify whenever someone crosses the green border and notify the system for the approaching person. If the individual gets past the red border, then the security measures are activated. The test was divided in two phases. The first one was to determine if the crossing users were successfully identified both when crossing individually and in groups. The second phase was to test if the security features were triggered correctly.

Cases	Point of Crossing Green	Point of Crossing Red
1 Person	(149, 230)	0
2 Persons	(154, 158)	(199, 251)
3 Persons	(196, 252)	(120, 123),(188, 236)

Table 8.4: User Tracking and Visibility Zones

The Table 8.4 is a small example of the tests conducted to test the tracking system. Whenever someone crosses one of the boundaries, then there is an output of the point of crossing. The boundaries have different outputs to distinguish the distance of the subject from the monitor.

8.4 Window Manager and Motion Tracking

Last but not least, the window manager was tested for all its combined features to realize whether messages are passed correctly between the components. The tests include various scenarios, where subjects with different access rights enter the visibility zone, while multiple windows are open with varying access levels. The Table 8.5 contains the performed tests which follow a similar approach as in the [FW04].

Events	**Test Case**	**Result**	**Status**
A person enters the visibility zone	No open windows	No actions	OK
A person enters the visibility zone	Windows with lower access levels than the approaching user	No actions	OK
A person enters the visibility zone	Windows with higher access levels than the approaching user	Windows are covered/become transparent	OK
The person leaves the visibility zone	Windows with higher access levels than the approaching user	Windows are back to normal	OK
Two persons enter the visibility zone	Windows with higher access levels from one of the approaching user	Windows are covered/become transparent	OK
Two persons enter the visibility zone	Windows with lower or the same access levels from the approaching users	No actions	OK
One person leaves the visibility zone the other stays	Windows with higher access levels from the remaining user	Windows are covered/become transparent	OK

Table 8.5: Window Manager

The system was able to handle users entering or leaving the office area, both individually and in pairs. As long as the users are authenticated beforehand, the tracking system is able to retrieve their access levels successfully. Nevertheless, as the number of people entering the room is increased, the system is more prone to tracking errors.

CHAPTER 9

Conclusion

The primary objective of this thesis was to demonstrate an access control mechanism, that combines logical and physical access control utilizing sensors. The proposed Sensor Enhanced Access Control model aims to counter the limitations of logical access control systems that make them insufficient when data is represented by an external device, while also providing contextual awareness. The prototype was implemented and tested on Ubuntu 16.04.1 but there should be no problems with other Unix distributions, since it can be mounted as a virtual file system. The model follows the traditional discretionary access control policy regulated by the UNIX reference monitor.

The contributions of the system include a module that operates as a virtual file system, which contains the rest of the components. These components include a motion tracking algorithm attached with sensors and a window manager that handles all open windows. The motion tracking algorithm identifies any possible movement and keeps track of any subject in the camera's range. The algorithm evaluates the subject's position and the distance from the computer screen. The window manager operates as an additional reference monitor, that relies on sensor reading provided by the motion tracking algorithm to mediate access to an object. When the subject proceeds towards the screen, then the window manager is responsible for securing any open files or directories. This is achieved either by making the windows invisible or creating an overlay window to cover them. Transparency is changed gradually to the point when windows become invisible as the distance from the computer is decreased, while the overlay windows follow an inverse approach. The overlay windows are created as completely invisible and they become more opaque according to the subject's position. Both solutions were tested by external users, that helped to evaluate their effectiveness in hiding the projected data and their impact on usability.

During the tests, the users provided feedback, with the majority of them reporting that window invisibility was more successful on hiding the windows but the impact in usability was greater, disrupting the users work. Additional, tests were performed to evaluate the prototype's ability to keep track of the individuals walking in the office and if it can identify and handle the window permissions correctly. Overall, the prototype performed well when individuals proceed in numbers of two or three persons at a time, but it failed when more than four people entered at the same time. On the other hand, the windows were identified correctly and access control was enforced successfully when needed.

Either approach for securing the windows is considered viable when planned correctly. The adaptation for each environment requires proper calibration since every environment contains different equipment, e.g. cameras and different conditions, e.g.

lighting. In conclusion, the prototype by itself, provides context awareness to the traditional access control models and can mediate access to both logical and physical objects. When combined with persistent authentication and remote authentication mechanisms, it can extend its capabilities further as an enhanced and ubiquitous security model.

9.1 Future Work

Future work on the prototype will primarily be focused on remote authentication and a state of the art tracking algorithm, that can keep track individuals and handle a bigger number of subjects. Furthermore, there is also the need to replace the access control model of UNIX with Role Based Access Control that can manage applications and processes in addition to files and directories. Additionally, the model can be integrated with more output devices such as printers or speakers providing security functions in corporate and home environments.

APPENDIX A

An Appendix

The appendix includes all the performed tests used to evaluate the prototype and some basic information about the UNIX permission system. To mount the prototype: sudo python fuse_fs.py <The target directory to view at the mount point> <The mount point to view the target directory>

A.1 Evaluation

The prototype was implemented and tested in Ubuntu version 16.04.1. Files and users were created for the tests with different access levels to serve our purposes. There will be an explanation of each file that describes the owner and the access rights. The access rights follow the traditional Unix permission scheme with read, write, execute for owner, group and other. The following tables serve as a reminder of the UNIX permission notation.

Letter Representation	**Octal Representation**	**Explanation**
rwx	7	Read, write and execute
rw-	6	Read, write
r-x	5	Read, and execute
r–	4	Read
-wx	3	Write and execute
-w-	2	Write
–x	1	Execute
—	0	no permissions

Table A.1: Unix Permissions

Permissions	**Octal**	**Field**
rwx——	700	User
—rwx—	070	Group
——rwx	007	Other

Table A.2: Unix Permission Scopes

A.1.1 FUSE and MAC tests

The files used have the following properties:

- foo.txt has owner the user A and its permission bits are 700 which means only user A has permissions.
- bar.txt has owner the user B and its permission bits are 700 which means only user B has permissions.
- test folder is created from user B with default permissions.

User	Command	Test Case	Expected	Status
A	cat > foo.txt	Create file	Create file	OK
A	cat foo.txt	Read File	Read File	OK
A	rm foo.txt	Delete File	Delete File	OK
A	cat bar.txt	Read File of User B	Not Permitted	OK
A	echo 'test test' » bar.txt	Write File of User B	Not Permitted	OK
A	rm bar.txt	Delete File of User B	Not Permitted	OK
B	link test.txt test2.txt	Link File test.txt to test2.txt	Permitted	OK
B	cp foo.txt bar.txt	Copy File of User A	Not Permitted	OK
B	mkdir test folder	Create Directory	Permitted	OK
B	unlink test2.txt	Unlink a File B	Permitted	OK
B	echo 'test test' » bar.txt	Write File of User B	Permitted	OK
B	rm bar.txt	Delete File of User B	Permitted	OK
root	sudo chmod 700	Change File Permissions	Permissions changed to root only	OK
root	sudo chown A bar.txt	Change File Owner	File new owner is A	OK
root	sudo rm bar.txt	Delete File	Permitted	OK
root	sudo stat bar.txt	Retrieve File's extended Attributes	Permitted	OK
root	sudo echo 'test test' » foo.txt	Write File of User A	Permitted	OK
root	sudo echo 'test test' » bar.txt	Write File of User B	Permitted	OK

Table A.3: FUSE and MAC extended tests

A.1.2 Window Manager tests

The table presents various opened windows in various positions and sizes. The permissions are only visible for the files and directories. The permissions are as follows:

- 755: Owner has full permissions. Group and Other can only read and execute
- 664: Owner and Group cannot execute. Other can read only
- 740: Owner has full permissions. Group can only read while Other has no permissions.
- 600: Owner can only read and write. Group and Other have no permissions.
- 700: Owner has full permissions. Group and Other have no permissions.

XID	(x,y) position	(width,height)	Window Name	Access Rights
79691792	(65,24)	(1301,647)	Mozilla Firefox	-
41945382	(306,24)	(890,494)	Documents	755
58720725	(724,45)	(642,406)	test text.odt - LibreOffice Writer	664
54526084	(65,24)	(1301,647)	foo.txt	740
56623114	(686,113)	(722,438)	test@ubuntu: /Desktop/foobar	-
81788936	(65,24)	(1301,647)	test.pdf	600
41945916	(65,24)	(1301,647)	foobar	700
109026	65,43)	(800,628)	Ubuntu Web Browser	-
109171	(65,24)	(1200,647)	Ubuntu Software	-

Table A.4: Windows position and access rights testing

A.1.3 Window Overlay and Window Cover Complete tests

The tables present the user tests to determine which implementation secures windows in the most optimal way.

User	Distance	Opacity 0.2 1-5 grade	Opacity 0.5 1-5 grade	Opacity 0.2 Time	Opacity 0.5 Time	No opacity Time
1	1.15m	2 out of 5	3.5 out of 5	83.712 seconds	48 seconds	45 seconds
2	1.15m	3 out of 5	4 out of 5	66.438 seconds	48.51 seconds	46.71 seconds
3	1.15m	2.5 out of 5	4 out of 5	79.764 seconds	56.69 seconds	49.62 seconds
4	1.15m	2 out of 5	3 out of 5	79.2 seconds	59.62 seconds	46.69 seconds
5	1.15m	2 out of 5	3.5 out of 5	67.72 seconds	53.54 seconds	51.65 seconds
6	1.15m	1 out of 5	3 out of 5	80.22 seconds	51.33 seconds	48.11 seconds
7	1.15m	2.5 out of 5	4 out of 5	70.83 seconds	53.05 seconds	48.34 seconds
8	1.15m	3 out of 5	3.5 out of 5	65.78 seconds	52.9 seconds	50.21 seconds
9	1.15m	2 out of 5	3 out of 5	72.61 seconds	50.47 seconds	47.3 seconds
10	1.15m	2 out of 5	3 out of 5	71.67 seconds	49.57 seconds	46.85 seconds
11	1.15m	2.5 out of 5	4 out of 5	75.05 seconds	52.21 seconds	49.23 seconds
12	1.15m	2.5 out of 5	3.5 out of 5	78 seconds	50.33 seconds	47.9 seconds
13	1.15m	3 out of 5	4 out of 5	74.5 seconds	55.12 seconds	50.06 seconds
14	1.15m	1 out of 5	3 out of 5	79.11 seconds	58.45 seconds	51 seconds
15	1.15m	2 out of 5	3 out of 5	75.9 seconds	49.5 seconds	48.03 seconds
16	1.15m	2 out of 5	3.5 out of 5	72 seconds	50.42 seconds	49.83 seconds
17	1.15m	2.5 out of 5	3.5 out of 5	70.53 seconds	48.87 seconds	46.35 seconds
18	1.15m	2 out of 5	3 out of 5	74.05 seconds	51.65 seconds	48.18 seconds
19	1.15m	2 out of 5	4 out of 5	76.28 seconds	53.09 seconds	50 seconds
20	1.15m	3 out of 5	4 out of 5	71.4 seconds	56.18 seconds	52.60 seconds

Table A.5: Transparent Windows all User Testing

User	Distance	Opacity 0.5 1-5 grade	Opacity 0.8 1-5 grade	Opacity 0.5 Time	Opacity 0.8 Time	No opacity Time
1	1.15m	3.5 out of 5	4.5 out of 5	49 seconds	45 seconds	45 seconds
2	1.15m	4,5 out of 5	4 out of 5	40 seconds	47,39 seconds	46.71 seconds
3	1.15m	5 out of 5	4 out of 5	38.16 seconds	42.60 seconds	49.62 seconds
4	1.15m	4 out of 5	3.5 out of 5	40.24 seconds	52.41 seconds	46.69 seconds
5	1.15m	3.5 out of 5	3 out of 5	39.17 seconds	51.02 seconds	51.65 seconds
6	1.15m	4 out of 5	3.5 out of 5	41.04 seconds	50.33 seconds	48.11 seconds
7	1.15m	4 out of 5	3 out of 5	40.33 seconds	50.15 seconds	48.34 seconds
8	1.15m	4 out of 5	3.5 out of 5	45.81 seconds	52 seconds	50.21 seconds
9	1.15m	3.5 out of 5	3 out of 5	42.01 seconds	50.47 seconds	47.3 seconds
10	1.15m	4 out of 5	3 out of 5	41.56 seconds	48.04 seconds	46.85 seconds
11	1.15m	4 out of 5	4 out of 5	42 seconds	50.11 seconds	49.23 seconds
12	1.15m	5 out of 5	3 out of 5	43.62 seconds	49 seconds	47.9 seconds
13	1.15m	4.5 out of 5	4 out of 5	41.19 seconds	48.86 seconds	50.06 seconds
14	1.15m	4 out of 5	3.5 out of 5	45.48 seconds	48.23 seconds	51 seconds
15	1.15m	4 out of 5	4 out of 5	42.86 seconds	51.68 seconds	48.03 seconds
16	1.15m	4 out of 5	3.5 out of 5	40 seconds	50.26 seconds	49.83 seconds
17	1.15m	4.5 out of 5	3.5 out of 5	39.82 seconds	45.28 seconds	46.35 seconds
18	1.15m	3.5 out of 5	3 out of 5	44.58 seconds	49.7 seconds	48.18 seconds
19	1.15m	4 out of 5	3 out of 5	46.5 seconds	53.15 seconds	50 seconds
20	1.15m	4 out of 5	3.5 out of 5	48.66 seconds	51.92 seconds	52.60 seconds

Table A.6: Overlay Windows all User Testing

Bibliography

[Abo+99] Gregory Abowd et al. “Towards a better understanding of context and context-awareness”. In: *Handheld and ubiquitous computing.* Springer. 1999, pages 304–307.

[AlM+03] Jalal Al-Muhtadi et al. “Cerberus: a context-aware security scheme for smart spaces”. In: *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on.* IEEE. 2003, pages 489–496.

[BN89] David FC Brewer and Michael J Nash. “The chinese wall security policy”. In: *Security and privacy, 1989. proceedings., 1989 ieee symposium on.* IEEE. 1989, pages 206–214.

[CN02] Mark D Corner and Brian D Noble. “Zero-interaction authentication”. In: *Proceedings of the 8th annual international conference on Mobile computing and networking.* ACM. 2002, pages 1–11.

[CN05] Mark D Corner and Brian D Noble. “Protecting file systems with transient authentication”. In: *Wireless Networks* 11.1-2 (2005), pages 7–19.

[Cov+01] Michael J Covington et al. “Securing context-aware applications using environment roles”. In: *Proceedings of the sixth ACM symposium on Access control models and technologies.* ACM. 2001, pages 10–20.

[Cov04] Michael J Covington. “A flexible security architecture for pervasive computing environments”. PhD thesis. Georgia Institute of Technology, 2004.

[DC48] Edgar Dale and Jeanne S Chall. “A formula for predicting readability: Instructions”. In: *Educational research bulletin* (1948), pages 37–54.

[EHD00] Ahmed Elgammal, David Harwood, and Larry Davis. “Non-parametric model for background subtraction”. In: *Computer Vision—ECCV 2000* (2000), pages 751–767.

[FW04] Kristine Frank and Ida C Willemoes-Wissing. “Combining logical and physical access control for smart environments”. Master’s thesis. Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2004.

[HFK06] Vincent C Hu, David Ferraiolo, and D Richard Kuhn. *Assessment of access control systems.* 2006.

[HKZ87] Robert A Hummel, B Kimia, and Steven W Zucker. “Deblurring gaussian blur”. In: *Computer Vision, Graphics, and Image Processing* 38.1 (1987), pages 66–80.

[Hu+14] Vincent C Hu et al. “Guide to Attribute Based Access Control (ABAC) Definition and Considerations”. In: *NIST Special Publication* 800 (2014), page 162.

[IAJ13] Mads I Ingwar, Naveed Ahmed, and Christian D Jensen. “Error-rate-based fusion of biometric experts”. In: *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on.* IEEE. 2013, pages 239–246.

[IBM14] IBM. *Develop your own filesystem with FUSE.* `https://www.ibm.com/developerworks/library/l-fuse/index.html`. 2014.

[Its15] Itseez. *Open Source Computer Vision Library.* `https://github.com/itseez/opencv`. 2015.

[JGW13] Christian Damsgaard Jensen, Kristine Geneser, and Ida C Willemoes-Wissing. “Sensor enhanced access control: extending traditional access control models with context-awareness”. In: *IFIP International Conference on Trust Management.* Springer. 2013, pages 177–192.

[Jon07] Tim Jones. “Anatomy of the linux file system”. In: *IBM developerWorks, October* (2007).

[Kap+07] Apu Kapadia et al. “Virtual walls: Protecting digital privacy in pervasive environments”. In: *International Conference on Pervasive Computing.* Springer. 2007, pages 162–179.

[KG00] Andrew J Klosterman and Gregory R Ganger. “Secure continuous biometric-enhanced authentication (cmu-cs-00-134)”. In: (2000).

[KH08] Martin Kirschmeyer and Mads Syska Hansen. “Persistent authentication in smart environments”. PhD thesis. Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2008.

[Lam74] Butler W Lampson. “Protection”. In: *ACM SIGOPS Operating Systems Review* 8.1 (1974), pages 18–24.

[Lib] Libwnck. *Libwnck Reference Manual.* `https://developer.gnome.org/libwnck/stable//`.

[NC02] Brian D Noble and Mark D Corner. “The case for transient authentication”. In: *Proceedings of the 10th workshop on ACM SIGOPS European workshop.* ACM. 2002, pages 24–29.

[Pic04] Massimo Piccardi. “Background subtraction techniques: a review”. In: *Systems, man and cybernetics, 2004 IEEE international conference on.* Volume 4. IEEE. 2004, pages 3099–3104.

[PIR15] PIROPO. *PIROPO database.* `https://sites.google.com/site/piropodatabase/`. 2015.

[PyG11] PyGTK. *PyGTK 2.0 Reference Manual.* `http://www.pygtk.org/pygtk2reference/`. 2011.

[Raj] Qasim Mahmood Rajpoot. “Enhancing Security and Privacy in Large-Scale Video Surveillance through Role-Oriented Access Control Mechanism”. In: ().

[SAW94] Bill Schilit, Norman Adams, and Roy Want. “Context-aware computing applications”. In: *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on.* IEEE. 1994, pages 85–90.

[SMS16] Babins Shrestha, Manar Mohamed, and Nitesh Saxena. “Walk-Unlock: Zero-Interaction Authentication Protected with Multi-Modal Gait Biometrics”. In: *arXiv preprint arXiv:1605.00766* (2016).

[TLC09] Massimo Tistarelli, Stan Z Li, and Rama Chellappa. *Handbook of remote biometrics.* Volume 1. Springer, 2009.

[Ver] Giorgos Verigakis. *Fusepy Documentation Release 2.0.2.* `https://media.readthedocs.org/pdf/fusepy/latest/fusepy.pdf`.