Sensor Enhanced Access Control: Extending Traditional Access Control Models with Context-Awareness

Christian Damsgaard Jensen, Kristine Geneser*, and Ida C. Willemoes-Wissing

Applied Mathematics & Computer Science
Technical University of Denmark
DK-2800 Kgs. Lyngby, Denmark
Christian.Jensen@imm.dtu.dk

Abstract. Access control models generally distinguish between physical access control that mediates access to physical resources such as buildings, sections of buildings or individual rooms, and logical access control that mediates access to logical objects such as information stored in files or databases. All logical access control models make some, more or less implicit, assumptions about the physical access control model, e.g. that servers are locked in a room with restricted access. However, problems arise when a logical object gets a physical representation, e.g. when a file is displayed on a screen or printed, because the logical access control model has no way to ensure, or even to monitor, that the physical access control policies are being enforced.

Traditionally, physical access control policies are enforced by compartmentalization. Users are separated from other users and resources by placing them in different physical locations such as different offices in a building. Access from one to the other is impossible without passing a guard or a door lock, i.e., guards or distribution of keys/access-cards effectively enforce the physical access control policy. However, these mechanisms are generally coarse-grained, inflexible and expensive.

In this paper, we propose a Sensor Enhanced Access Control (SEAC) model that extends existing logical access control models with context-awareness. This allows the model to incorporate information about the physical environment and to explicitly define and enforce physical access control policies for logical objects that have physical representations. A prototype implementation of the SEAC model has been developed for the Unix platform. The prototype protects file data when displayed on a computer screen by managing the visibility of windows in the X Window System. Context-awareness is provided by a simple motion detection system build using cheap web-cameras. However, the system is designed so that the sensor component easily can be replaced, making it possible to deploy advanced sensor technologies.

1 Introduction

Access control is traditionally restricted to logical access control where access only is granted to authorized users in authorized locations. The authorization of a user is usually determined after the user is identified and authenticated by the system, using for

* Kristine Geneser is the married name of Kristine Frank.

C. Fernandez-Gago et al. (Eds.): IFIPTM 2013, IFIP AICT 401, pp. 177–192, 2013.

example a user name and corresponding password. After such a login procedure, a user ID will be associated with every process the user starts. The user ID will be used to determine which resources the process is permitted to access. This form of access control is logical since subjects and objects are logical representations of physical entities, such as processes running on behalf of users and information stored in a file system. After a user has logged in, an unauthorized person may also be able to obtain physical access to the resource. This could, for instance, occur if the user temporarily leaves his computer unattended. Another, more severe, scenario could be that an intruder uses a weapon to coerce the authorized user into providing access to the resource; the access would then be obtained despite the user's physical presence. In any case, a process that is started on the computer cannot determine who are currently present in the environment. It only knows the user ID of the person who currently is logged in. In a military or commercial setting, it can have severe implications if an unauthorized person obtains access to classified information by circumventing the logical access control.

In this paper, we propose a *Sensor Enhanced Access Control* (SEAC) model that not only takes logical entities into consideration when enforcing access control, but also the persons who are physically present in the environment and the physical representation of logical objects. The model consists of a logical and an environmental access control part. The *logical access control* part is a traditional access control model where e.g. processes and files are considered to be subjects and objects, respectively. The SEAC model does not impose any restrictions on the choice of the underlying access control model, and it can therefore accommodate any mandatory or discretionary access control model. The novel and important aspects of the SEAC model is not the choice of access control model, but the extension with environmental information. The *environmental access control* mediates access to the physical objects by persons. The physical objects are physical representations of logical objects, which are exported to an output device such as a display, a printer or loudspeakers. The identity of persons and other security relevant contextual information is captured by sensors and used in the access mediation.

The SEAC model is designed to counter the granularity gap between traditional logical and physical access control models: By extending logical access control models so that they also encompass objects when they are represented physically, a comparable level of granularity can be obtained. A system based on the SEAC model can be used to evade the paradox situation where fine-grained logical access control protects objects, but the physical representation of objects is only protected by coarse-grained access control such as a guard at the main entrance of a building.

The extension of a traditional access control models with context-aware computing [23] has (to our knowledge) not been explored in the past. The current access control models for context-aware computing are mainly concerned with securing applications in a pervasive computing environment[7,9,8,2,13,26]. None of these models address how access mediation in a traditional computer system can benefit from contextual information. On the other hand, models have been developed that protect information when it obtains physical form on an output device[1,4]. These models are, however, not context-aware and are merely concerned with labelling the output on peripheral devices, e.g. a display or paper, with its security classification. Furthermore, these models are

mainly used by the military, and strong physical access control to the premise of the computers is therefore also deployed. In organizations with lower security requirements (and budgets), it may not be feasible to deploy fine-grained physical access control to the premise of the computers. These organizations may find an access control solution based on context-awareness useful. For example, an open-plan office in a company with many employees and visitors can use it to enforce that windows become invisible when unauthorized persons are present in front of the display.

To demonstrate the validity of the SEAC model, a prototype system based on the model has been developed. The system is designed so that it can be integrated with an existing Unix system. The logical access control is enforced by a file system that, in addition to storing files, acts as a reference monitor when it mediates access to files by processes. The environmental access control is implemented using services provided by the X Window System and two web-cameras that are used as simplified sensors. The rules that are used when mediating access to objects by subjects are based on the Bell-LaPadula confidentiality model. The environmental access control will enforce a no-view up rule, that is, persons can only read data in windows at their own level or a lower level. If a person who is not authorized to see the information displayed in a window, this window will disappear from the display so that the content is no longer is human-readable.

The reminder of this paper is organized as follows: The granularity gap between traditional logical and physical access control models are discussed in Section 2. The SEAC model is presented in Section 3. The design and implementation of our prototype is presented in Section 4. The evaluation of the prototype is discussed in Section 5, and related work is reviewed in Section 6. Finally, our conclusions and guidelines for future work are presented in Section 7.

2 The Granularity Gap

The protection of logical objects within computers has been widely explored during the past decades, resulting in a wide range of access control models that meet different security requirements. In contrast to the availability of logical access control models, relatively few access control mechanisms are available to regulate access to the premise of the computers. The oldest physical access control mechanism is to post guards who control access to a room or equipment. The disadvantages with guards are that they must be on duty continuously, are expensive, and must personally recognize the authorized persons or an access token such as a badge. The second oldest physical access control mechanism is to use a lock. This mechanism is cheaper and simpler than using a guard. The main issues with the use of keys and other access tokens are that they can be lost or forged. Furthermore, the revocation or modification of access rights can be troublesome with access tokens.

When considering the granularity aspects of physical access control mechanisms, the use of locks is relatively coarse-grained. Guards enforce more fine-grained policies, but they have to be paid, so this mechanism can be quite expensive. Furthermore, if a fine-grained solution is required, it will typically cause more inconvenience for the authorized users who frequently have to present their credentials.

A coarse-grained mechanism may be sufficient if the physical access control is only required to prohibit unauthorized outsiders from obtaining access to the premise of the computers. Many companies do, however, require that the access of insiders also is controlled. For instance, it may be a requirement that only top-executives have permission to the computer room where confidential corporate information is processed. A fine-grained access control solution can be obtained by using devices that interface with a computer. Such devices can be smart cards, biometric mechanisms, RFID tags, etc.

The granularity gap between the protection provided by traditional physical and logical access control models can result in some paradox security situations. For example, in a corporate setting where strict logical access control is enforced, the physical representation of objects may be accessible to all within the building. The only physical access control may be a guard at the main entrance, and everyone inside the building will consequently have access to all physical objects, e.g. an object rendered on a computer display will typically be unprotected while the owner is out for a cup of coffee. It is a paradoxical situation that much effort is put into the protection of logical objects, whereas physical objects only are protected by old-fashioned means such as guards and locks.

3 Sensor Enhanced Access Control

The limitations of traditional physical access control mechanisms can be addressed by using advancements in context-aware computing. In particular, the use of sensors to detect persons in an environment can make other physical access control mechanisms superfluous. Furthermore, the use of sensors can provide a very fine-grained and flexible solution. The *Sensor Enhanced Access Control* (SEAC) model is an extended access control model which in addition to logical entities also encompasses physical entities. The SEAC model consists of a logical access control part that is extended with an environmental access control part. Figure 1 illustrates the logical and physical entities in the SEAC model and their relationships. The logical access control part is a traditional access control model such as the Bell-LaPadula[16], RBAC[11,22], or Chinese Wall[6] model. The important part in the SEAC model is not the choice of logical access control model, but the extension with context-awareness via the environmental access control part.

The environmental access control is concerned with mediating access to physical objects by persons. A *physical object* is a physical representation of a logical object. More precisely, a physical object is presented to persons using an output device that makes it perceivable to one of the human senses. Most computers present output in a viewable format where the output device is a computer display or a printer and the corresponding physical objects are a window on the display or a printed paper, respectively. Loudspeakers is an example of an output device that generates a physical object which is perceived by the hearing sense and in this case the physical object is the pressure waves that constitutes sound. An example where it is useful to control access to sound is if the loudspeakers play a recording of a confidential meeting or medical information recorded by a doctor.

A *person* is defined as any human principal, and the access operation is the human sense used to perceive the physical object. The *environment* is defined as the physical

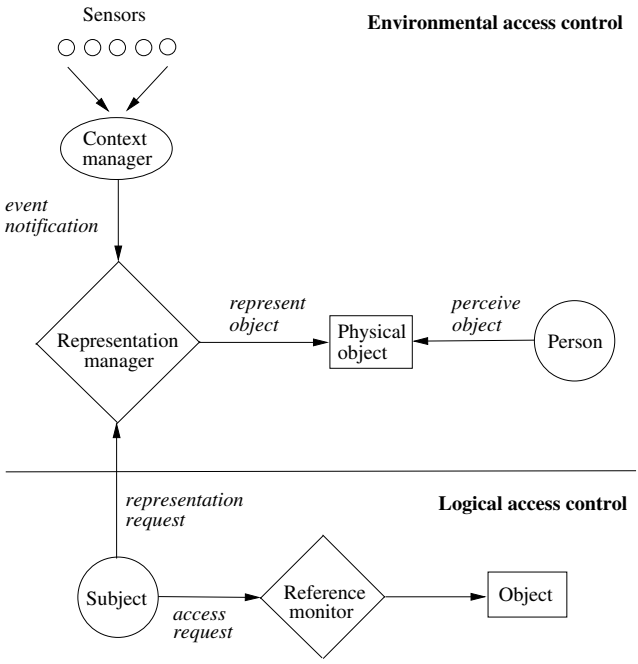

Fig. 1. The Sensor Enhanced Access Control model

location from where a person can perceive the physical object. If the output device is a display or a printer, the environment is the area in front of the display or the printer room, respectively. The environment where sound can be perceived is relatively large, but headphones or loudspeakers that produce directional sound can be used to limit the size of the environment.

The *representation manager* enforces the environmental access control: The persons in the environment are only granted access to a physical object if this is permitted by the representation manager. For example, if the information in a window or on a piece of paper must not be seen by the persons, the representation manager can deny access by removing the window or cancelling the printing job. Since the representation manager mediates access to physical objects by physical subjects (that is, persons), it acts as a reference monitor for the environmental access control. When a person requests access to an object, a user-initiated subject must therefore first request the reference monitor for access to the object and, secondly, it must request the representation manager for access to represent the object as a physical object.

The security policy enforced by the representation manager will (usually) resemble that of the logical access control model, ensuring that the same level of granularity is enforced both logically and physically. This policy is defined in terms of properties related to physical objects and contextual information such as time and identities of persons in the environment. For example, a policy can state that certain persons can only access a physical object during working hours.

All contextual information is captured by sensors. In general, sensors can capture a variety of information such as noise, temperature, or motion. When mediating access to physical objects, the most relevant information is the identity and location of persons. Depending on the capabilities of the sensors, they may capture fine-grained biometric information such as facial features, or they may be very basic and only detect whether someone is present or not. Another possibility is to let the sensors read an identity from a badge; however, this solution is not secure because a person can gain another persons identity by simply wearing his badge.

The *context manager* aggregates information from all the deployed sensors. Whenever the context manager has detected a security relevant event, the representation manager is notified about this event, ensuring that the environmental access control is mediated dynamically. A person who is detected by sensors, but cannot be identified, must be associated with a default clearance, which reflects the underlying physical access control mode. For example, if only persons cleared to "secret" can enter a particular part of a building, it is safe to assume that the clearance of the unidentified person is at least "secret". For persons who can be identified, the captured identity must be verified using an authentication service.

The SEAC model is mainly concerned with providing confidentiality as it enforces access control to output devices and prevents that unauthorized persons can perceive classified information. The model does not encompass other security requirements such as integrity and availability. It is, however, not difficult to develop a dual SEAC model which provides integrity: Whenever a physical object is presented on an input device, the dual SEAC model must prevent that unwanted physical objects are represented internally as a logical object. This duality is reminiscent of the duality between the Bell LaPadula confidentially model and Biba integrity model.

4 The SEAC Prototype

To demonstrate the feasibility of the SEAC model, we have developed a simple prototype system[12], which is integrated with an existing Unix system.

4.1 Adapting the SEAC Model to Unix

An overview of our adaption of the SEAC model can be seen in Figure 2. The separation of user and kernel space found in a Unix system has been included in the figure since it is important from a security perspective.

Logical Access Control. The logical access control model that we have chosen to use in our system is a simplified version of the Bell-LaPadula model[16]. The reference monitor enforces the no read up and no write down rules a.k.a. the simple-security property and *-property in the Bell-LaPadula model. In a Unix system where information flow must be prevented, the only system calls that are subject to access control restrictions are read and write.

In our adaption of the Bell-LaPadula model, only process and files are considered to be subjects and objects, respectively. A *file* is a passive entity that stores information on

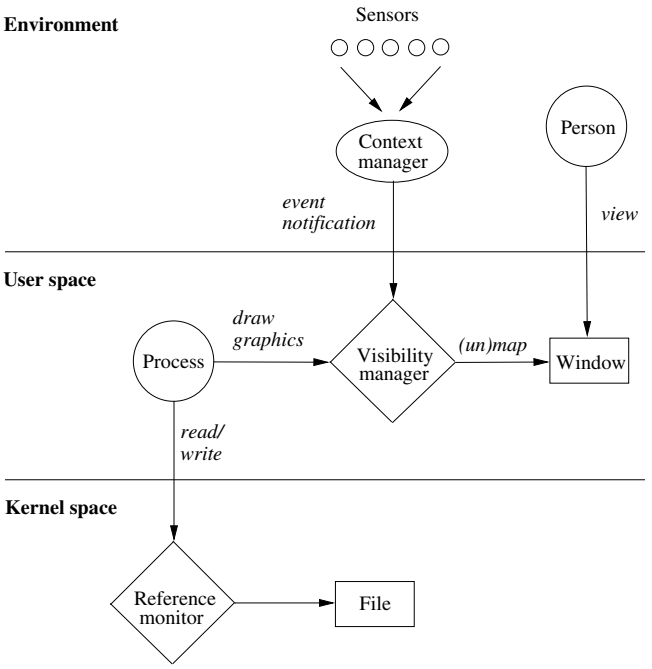

Fig. 2. Our adaption of the SEAC model to the Unix platform

a computer and is uniquely identified by an *inode number*. There exists many types of files, such as regular files, directories, or symbolic links. A *file level* is associated with each inode number, and it is used by the reference monitor when access to a file with a given inode number is mediated. File levels correspond to classification labels in the the Bell LaPadula model.

A *user* is an active entity that can access files. Each user has an account, and a user must identify herself and be authenticated as part of a login procedure before she can access the system. When a file is accessed by a user, it is actually not the user who directly accesses the file, but a process that runs on behalf of the user. The logical subject is therefore a user-started *process*. A user is uniquely identified by a *user ID*, and this user ID is associated with all processes that are started by the user. A *user level* will be associated with each user ID, and it constitutes the basis for deciding what files a given user has access to. User levels correspond to clearance labels in the the Bell LaPadula model.

Environmental Access Control. Without loss of generality, we only consider windows on displays as the physical objects in the environmental access control. In a Unix system, windows are typically managed by the X Window System, which we rely on in our prototype. The X Window System is based on a client-server architecture where the X server offers graphics display services to the X clients. An X client is an application, such as a text editor or an Internet browser, that sends drawing requests to the X server. It is the responsibility of the X server to render the appropriate bits on the user's display.

A *window* is as an area of the display that is used to represent a computation graphically. In Figure 2, a window has been drawn in user space since it is represented internally by a data structure which is manipulated by the X server. A window will, however, be visible in the environment when the X server sends drawing requests to the computer's graphics card. Each window on a display must be uniquely identified by a *window ID*. Most GUI applications will create one *top-level window* and possibly some sub-windows inside this special window. A sub-window can, for instance, be a button, a menu, or a scrollbar. The SEAC system only uses the first top-level window created by an application and ignores any further created top-level windows as well as all the sub-windows of top-level windows. This ensures that an application can be uniquely identified by a window ID.

The access operation for a window is *viewing*. If access to viewing a window is denied, the window will be *unmapped* so that it no longer is visible on a computer display. Otherwise, if access to a window is granted and it currently is unmapped, the window will be *mapped*.

Like the logical access control, the environmental access control is based on a multilevel security model. A *window level* is associated with each top-level window, and its value is determined using a rule based on the high-water mark principle[15]: If two files are opened by a GUI application, the window level of the application becomes the higher of the two file levels. Consequently, the level of a window can only increase.

The term *environment level* will denote the level of a person as it is detected by a sensor. If a person cannot be identified, i.e., no environment level can be associated with a human principal known to be present, a default level is associated with that person. The *clearance level* models a combined level for all the persons in a given environment and the user. It is equal to the minimum of all the detected environment levels and the user level.

The representation manager is denoted the *visibility manager* and it uses the clearance level and the window levels to enforces a no view up rule: if a window level is greater than the clearance level, the window is unmapped. The clearance level is also used by the reference monitor as our adaption of the Bell LaPadula model compares the clearance level, and not the user level, with a file level when access to a file is mediated.

4.2 Design and Implementation

The system we have developed encompasses many different types of functionality, ranging from storing file and user levels to detecting when persons enter or leave a given environment. The system implements a layered architecture, where the lowest layer is a file system which, in addition to storing files, enforces logical access control. The middle layer enforces the environmental access control, i.e., it is responsible for changing the visibility of windows on a display. The services provided by this Visibility Management layer are used by sensors in the Sensors layer whenever persons are detected in the environment. The layered architecture is closed in the sense that each layer only uses the services provided by the layer immediately below it. If only logical access control is required, the MAC file system can be used independently of the higher layers. In the remainder of this section, the implementation of these three layers will be described.

To support information classification of stored data in the Unix system, we had to implement a special file system that associates a file level with every file stored in the file system. The development of a new file system is, however, a difficult and tedious process. In order to focus on the access control aspects of the file system, we decided to use a stackable file system. A *stackable file system*[21] is a layer that resides above a native file system and below the Virtual File System (VFS). This is illustrated in Figure 3. Because existing file systems and interfaces are used, it is much easier to develop a stackable file system than a traditional file system.

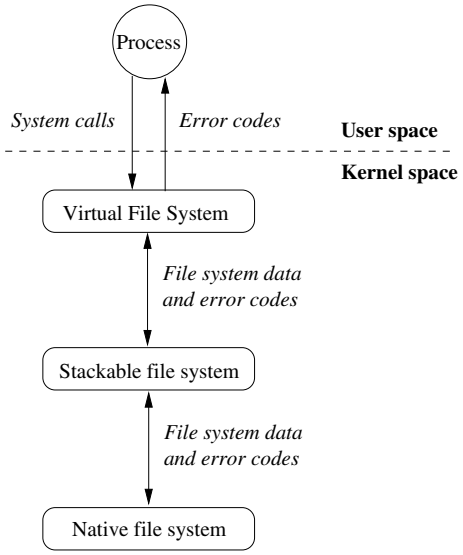

Fig. 3. A stackable file system can be used as a reference monitor if it mediates access to files in a native file system by processes

The stackable file system used in our system is denoted `macfs` (mandatory access control file system), and it is developed using the FiST (File System Translator) system[24,25]. FiST enables programmers to write stackable file systems for several versions of Linux, Solaris, and FreeBSD. The performance overhead when using FiST generated stackable file system is only 1-2% [25].

The main objective of the `macfs` is to enforce the logical access control part of the SEAC model. For this purpose, `macfs` stores a file level associated with each file in the underlying native file system and a user level corresponding to each user who has access to the system. Using these file and user levels, `macfs` implements a reference monitor that mediates access to files by user processes.

A FiST generated stackable file system is a kernel module. When compared to static kernel code, a kernel module has the advantage that it can be loaded and unloaded from memory separately from the main body of the kernel. It is therefore not necessary to rebuild and reboot the kernel every time a new module is to be added.

It is very important from a security perspective that the stackable file system is part of the kernel as it then is protected from non-privileged users via the operating system user/kernel modes. Furthermore, including access control mechanism in the kernel gives better performance because fewer context switches have to be made.

Visibility Management. The X server and X clients run in the user space part of a Unix system. Consequently, the part of our system that manages the visibility of windows must also reside in user space so that services provided by the X server can be used. We have developed an X client, denoted `visibility manager`, that sends requests to the X server whenever a window must be mapped or unmapped.

Our prototype works with both text editors and pure file readers such as Internet browsers. These GUI applications use library functions provided by the X Window System when making drawing requests. We have developed a preloaded shared library which overloads the function that creates a new window. Whenever an application requests the creation of a top-level window, our library will ensure that a message is sent to the `visibility manager` process before the real function is invoked. The message contains the process ID of the application and the window ID of the top-level window. Message 1 in Figure 4 illustrates this message exchange. The `visibility manager` process maintains a table with information about all created top-level windows and the corresponding applications that have created them.

The level of a window is equal to the maximum file level which is associated with a file opened by the application. Because the `visibility manager` mediates access to windows, it must be informed about all the files that an application opens. This part is implemented using an auxiliary process, `file open monitor`, which blocks on a semaphore in `macfs`. Whenever a file is opened in `macfs`, the semaphore is signalled and the `file open monitor` process unblocks and sends a message to the `visibility manager`. This message contains the process ID of the application, the inode number and the file level associated with the newly opened file (see Message 2 in Figure 4).

Besides information about the window levels, the `visibility manager` must also know the levels of all the persons who are present in the environment so that the clearance level can be updated accordingly. Therefore, a second auxiliary process, `sensor server`, is included in the system. This process aggregates data from all the deployed sensors and it will thus function as a simple context manager. The aggregated data includes the environment level of a detected person and the movement direction of the person. The term *direction* denotes the movement direction of the person. It is a binary value which models that the person either enters or leaves the environment.

The three processes `visibility manager`, `file open monitor`, and `sensor server` must be started by the super user and subsequently run as demon processes. It is very important from a security perspective that they are started by the super user since they then will be protected by the Unix security mechanisms. In particular, a non-privileged user must not be allowed to kill them by sending them a signal, regardless of whether this is done deliberately or not.

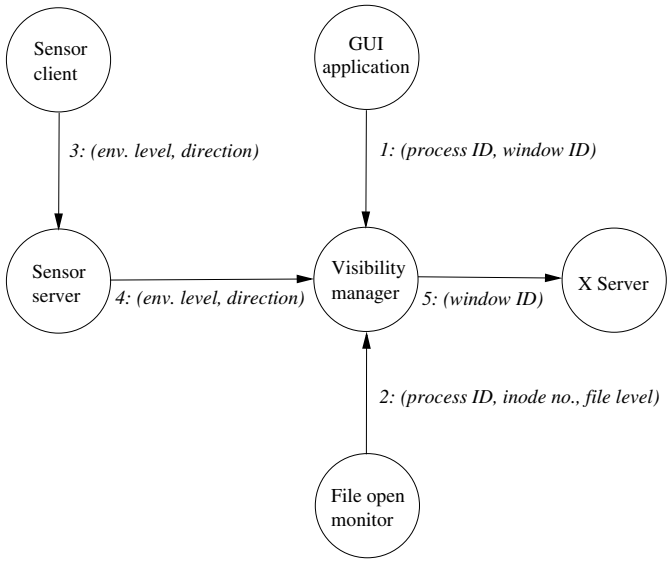

Fig. 4. The `visibility manager` process is a representation manager. It receives information about created windows, opened files, and detected persons and uses these data when it requests the X server to map or unmap windows.

Sensors. As several sensors may be used simultaneously, we have chosen to implement the sensor part of the system using the client/server paradigm. The `sensor server` receives data from all the deployed sensors, which are denoted `sensor client`, and relays the data to the `visibility manager` (see Message 3 and 4 in Figure 4). The window levels and the clearance level are used when the `visibility manager` sends requests to the X server about which windows should be mapped or unmapped (see Message 5).

The `sensor client` used in our prototype system is based on two web-cameras and a motion detection program. The web-cameras are only used to determine whether someone enters or leaves the environment and a common level is therefore assigned to all the detected persons. This common level depends on the physical access control that is enforced in the monitored location. A more fine grained physical access control policy can be enforced if it is possible to associate a specific *user ID* to the person entering the area, e.g. through the use of *persistent authentication* [18,14], but this is not implemented in the current prototype.

5 Evaluation

The main security requirement that the SEAC system has been designed to meet is access control. In this section, we will describe how well the prototype meets this requirement by evaluating the mediation of access to windows by persons. Since a system

only is as strong as its weakest link, we will also evaluate other security properties of the SEAC system.

5.1 Access Mediation to Windows by Persons

The environmental access control in the SEAC system must enforce that GUI applications are mapped and unmapped in accordance with the authorizations of the persons who enter or leave the environment. The system has been designed to work with GUI applications that only create one top-level window since this avoids ambiguity about which windows should be mapped or unmapped. However, not all applications work this way, and the Inter-Client Communication Conventions Manual[20] used in the X Window System does not specify that all applications should use the same window hierarchy. The prototype can therefore not be guaranteed to work with all applications, and users should be restricted to using applications that successfully can be unmapped.

The Unix version used the GNOME desktop environment with the `metacity` window manger. The test showed that typical applications, such as the `emacs`, `nedit`, and `mozilla`, fully integrate with our system, i.e. windows were mapped or unmapped according to the no-view up rule. The system could, however, not unmap the `gedit` editor because the communication between `gedit` and the window manager `metacity` was very unpredictable. We think that the main reason for this unpredictable behaviour is that `gedit` is tightly integrated with GNOME and therefore may use a custom protocol when it communicates with `metacity`. After all, this unspecified behaviour is permitted since the X Window System provides mechanism, not policy.

5.2 Security Analysis

The security provided by the logical access control part of the prototype depends on the used Unix system and the security it provides. In particular, the `macfs` file system is only a kernel module that interacts with other parts of the Unix system. Security procedures such as user identification and authentication are therefore beyond the boundaries of our system, and we must just assume that they are secure.

The environmental access control in the SEAC model is mainly concerned with access control to output devices such as displays and printers, but when constructing a secure system one should also remember to secure input devices and all communication channels. In particular, the communication between the sensors and the `visibility manager` must be secured so that traditional security requirements such as confidentiality, integrity, and availability are met.

In general, when deploying a security mechanism in an environment with high security requirements, one should notice that these requirements cannot be met with a mainstream operating system[17]. Our system is no exception since it is based on a regular Unix version. For example, the X Window System is not designed with security in mind and it does not meet any of the basic security goals confidentiality, integrity and availability[10]. The visibility management part of our system is actually exploiting the lack of availability: the X client `visibility manager` unmaps the windows of other X clients, and no permission checks are made before this occurs.

Even if all software and hardware parts of the system are assumed to be secure, the security provided by the system can be breached if the super user (a.k.a. root in Unix) cannot be trusted. The actions of the super user are not restricted by any access control, and he can therefore easily circumvent all access control mechanisms that protect logical objects. The environmental access control can also be evaded by the super user, for instance by killing the `visibility manager` process. This vulnerability implies that the person who is able to log in as the super user should use some form of physical access control, such as locking the door, before using the system. Otherwise, an unauthorized person might enter the environment and coerce the person into misusing the super user privileges, bypassing all logical access control mechanisms provided by the system.

6 Related Work

The use of environment information in access control mediation has previously been explored by Covington et al.[9,7] and Bertino et al.[5]. Covington et al. propose GRBAC, which is an extended RBAC model that, in addition to the traditional subject roles, includes object roles and environment roles. Object roles capture properties of resources in the system, and environment roles capture context information. All types of roles can be used to define fine-grained and flexible security policies. Bertino et al. propose GEO-RBAC, where spatial entities are used to model objects, user positions, and geographically bounded roles. Roles are activated based on the physical or virtual position of the user. An important difference between SEAC and both GRBAC and GEO-RBAC is that SEAC also includes persons who cannot be identified. If a person is detected, but cannot be identified in the SEAC model, the default clearance is associated with the person. This will prevent that unidentified persons obtain access to a physical object, for which they are not authorized.

Context-aware security mechanisms have been explored by Aziz and Jensen[3], who describe a mechanism that adapts the security properties of client-server communication in CORBA based on context information at runtime. The mechanism uses a context server to enforce multiple security policies defined for the current context, e.g., to encrypt all communications when a mobile client is away from the home network. While this work explores ideas that are similar to the ideas presented in this paper, their work focuses on communications security, but access control is not addressed. Moreover, their system is based on simple software based sensors, e.g., to determine whether the client is at home by comparing the current IP address identical to the home IP address and it does not explicitly address the use of external sensors.

7 Conclusion

Traditional access control models protect logical objects within a computer; however, these models are inadequate when the objects are represented as physical objects to persons, e.g. in the form of windows on a computer screen or printed papers. To protect physical objects, physical access control is therefore typically deployed in the form of

guards or locks. This access control mechanism is typically coarse-grained, expensive, and inflexible.

To counter the limitations of traditional access control models, we present a novel access control mechanism: any traditional logical access control model can be extended to encompass physical representations of logical objects. Corresponding to the reference monitor used in logical access control models, we propose a representation manager that mediates access to physical objects by persons. The representation manager is context-aware and uses information about the authorizations of the persons in the environment when it mediates access. The presented model is denoted the Sensor Enhanced Access Control (SEAC) model because it relies on sensors to capture environmental information. A system based on the SEAC model can be used as a replacement or enforcement of traditional physical access control mechanisms. In a context-aware environment where sensors are deployed, such a system can provide a fine-grained and flexible access control mechanism.

A prototype system based on the SEAC model has been developed for the Unix platform. It consists of three main parts: Firstly, a stackable file system acts as a reference monitor when it mediates access by processes to files. Secondly, services provided by the X Window System are used to ensure that the visibility of windows change according to the common authorization of the persons in the environment. Finally, a simple movement sensor based on two web-cameras are used to detect persons in the environment.

7.1 Future Work

The development of the prototype demonstrates the applicability of the SEAC model. In particular, it demonstrates that access control to physical objects in the form of windows can be enforced successfully. Future work on our prototype can include enforcement of access control to printers and other output devices. This can, for instance, be implemented by utilizing that many types of resources are accessed via special files in a Unix system. In particular, the `/dev` directory will (usually) contain files for accessing printers, hard disks, modems, terminals on remote computers, etc. Access control to a printer can be enforced by restricting write access to the printer special file, which typically is the file `/dev/lp0`. FiST-generated stackable file systems can (at the time of writing) not be stacked on top of directories containing special files, so this type of access control is not supported by our prototype.

The `sensor client` implemented in the current prototype does not associate a unique *user ID* with the different people moving around in the area. We plan to address this limitation by integrating the SEAC prototype with a fine grained remote authentication mechanism, such as persistent authentication [18], possibly enhanced with remote biometrics [19] to improve robustness.

As with most other security mechanisms, the SEAC model can also result in some inconvenience for the users: if a user e.g. sits in an open-plan office where unauthorized persons frequently pass the computer, the user will (eventually) be very annoyed because the windows disappear all the time. The windows will, of course, reappear on the screen again once the persons are gone, but it will nevertheless be irritating. An area of

future work within usability and HCI is the provision of availability of physical objects for authorized persons.

Developments in sensor technologies for pervasive computing environments will provide a basis for interesting work in the near future. These sensors will be able to capture detailed contextual information, which can be used in a fine-grained and flexible access control model where all physical representations of logical objects are protected. We therefore plan to explore the integration of our work on *persistent authentication* with the SEAC model presented in this paper.

References

1. Department of defence trusted computer system evaluation criteria. The Orange Book (December 1985), http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html
2. Al-Muhtadi, J., Ranganathan, A., Campbell, R., Dennis Mickunas, M.: Cerberus: A context-aware security scheme for smart spaces. In: PERCOM 2003: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, p. 489. IEEE Computer Society (2003)
3. Aziz, B., Jensen, C.: Adaptability in corba: The mobile proxy approach. In: International Symposium on Distributed Objects and Applications, pp. 295–304 (2000)
4. Berger, J.L., Picciotto, J., Woodward, J.P.L., Cummings, P.T.: Compartmented mode workstation: Prototype highlights. IEEE Trans. Softw. Eng. 16(6), 608–618 (1990)
5. Bertino, E., Catania, B., Damiani, P.P.M.L.: Geo-rbac: a spatially aware rbac. In: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, pp. 29–37 (June 2005)
6. Brewer, D., Nash, M.: The chinese wall security policy. In: Proceedings of the 1989 IEEE Symposium on Security and Privacy, pp. 206–214. IEEE Computer Society Press (May 1989)
7. Covington, M.J.: A Flexible Security Architecture for Pervasive Computing Environments. PhD thesis, College of Computing, Georgia Institute of Technology (April 2004)
8. Covington, M.J., Long, W., Srinivasan, S., Dev, A.K., Ahamad, M., Abowd, G.D.: Securing context-aware applications using environment roles. In: SACMAT 2001: Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies, pp. 10–20. ACM Press (2001)
9. Covington, M.J., Moyer, M.J., Ahamad, M.: Generalized role-based access control for securing future applications. In: Proceedings of the National Information Systems Security Conference, NISSC (2000)
10. Kilpatrick, C.V.D., Salamon, W.: Securing the x window system with selinux. Technical report, NAI Labs (2003)
11. Ferraiolo, D., Kuhn, R.: Role-based access controls. In: 15th NIST-NCSC National Computer Security Conference, pp. 554–563 (1992)
12. Frank, K., Willemoes-Wissing, I.C.: Combining logical and physical access control for smart environments. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark (2004), http://www.imm.dtu.dk/pubdb/p.php?3401
13. Hengartner, U., Steenkiste, P.: Access control to information in pervasive computing environments. In: Proceedings of 9th Workshop on Hot Topics in Operating Systems (HotOS IX), Usenix (2003)
14. Kirschmeyer, M., Hansen, M.S.: Persistent authentication in smart environments. Imm-thesis-2008-16, Department of Informatics & Mathematical Modelling, Technical University of Denmark (2008)

15. Landwehr, C.E.: Formal models for computer security. ACM Computing Serveys 13(3), 247–278 (1981)
16. LaPadula, L., Bell, D.E.: Secure computer systems: A mathematical model. MITRE Technical Report 2547, II, May 1973. An electronic reconstruction by Len LaPadula (November 1996)
17. Loscocco, P.A., Smalley, S.D., Muckelbauer, P.A., Taylor, R.C., Turner, S.J., Farrell, J.F.: The inevitability of failure: The flawed assumption of security in modern computing environments. In: Proceedings of the 21st National Information Systems Security Conference, pp. 303–314 (1998)
18. Jensen, C.D., Hansen, M.S., Kirshmeyer, M.: Persistent authentication in smart environments. In: Proceedings of the 2nd International Workshop on Combining Context with Trust, Security, and Privacy, Trondheim, Norway, pp. 31–44 (June 2008)
19. Tistarelli, M., Li, S.Z., Chellappa, R. (eds.): Handbook of Remote Biometrics. Springer, New York (2009)
20. Rosenthal, D.: Inter-Client Communication Conventions Manual. Sun Microsystems, Inc., version 2.0 edition (1994), `ftp://ftp.x.org/pub/R6.6/xc/doc/hardcopy/ICCCM/icccm.PS.gz`
21. Rosenthal, D.S.H.: Evolving the vnode interface. In: Proceedings of the Summer USENIX Technical Conference, pp. 107–118 (1990)
22. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
23. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, pp. 85–90 (1994)
24. Zadok, E.: FiST: A System for Stackable File System Code Generation. PhD thesis, Computer Science Department, Columbia University (May 2001), `http://www.cs.columbia.edu/~ezk/research/thesis`
25. Zadok, E., Nieh, J.: Fist: A language for stackable file systems. In: Proceedings of the Annual USENIX Technical Conference, pp. 55–77 (June 2000)
26. Zhang, G., Parashar, M.: Context-aware dynamic access control for pervasive applications. In: Communication Networks and Distributed Systems Modeling and Simulation Conference, CNDS 2004 (2004)