available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Adapting usage control as a deterrent to address the inadequacies of access controls

Keshnee Padayachee[a,*], *J.H.P. Eloff*[b]

[a]*University of South Africa, School of Computing, PO Box 392, Unisa, Pretoria, Gauteng 0003, South Africa*
[b]*Information & Computer Security Architectures Research Group, Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa*

A R T I C L E I N F O

Article history:
Received 4 April 2008
Received in revised form
12 March 2009
Accepted 13 March 2009

Keywords:
Usage Control
Optimistic Access Control
Deterrent Control
Access Control
Aspect-Oriented Programming

A B S T R A C T

Access controls are difficult to implement and evidently deficient under certain conditions. Traditional controls offer no protection for unclassified information, such as a telephone list of employees that is unrestricted, yet available only to members of the company. On the opposing side of the continuum, organizations such as hospitals that manage highly sensitive information require stricter access control measures. Yet, traditional access control may well have inadvertent consequences in such a context. Often, in unpredictable circumstances, users that are denied access could have prevented a calamity had they been allowed access. It has been proposed that controls such as auditing and accountability policies be enforced to deter rather than prevent unauthorized usage. In dynamic environments preconfigured access control policies may change dramatically depending on the context. Moreover, the cost of implementing and maintaining complex preconfigured access control policies sometimes far outweighs the benefits. This paper considers an adaptation of usage control as a proactive means of deterrence control to protect information that cannot be adequately or reasonably protected by access control.

1. Introduction

Industry surveys prove that a substantial portion of computer security incidents are due to the intentional actions of legitimate users, the consequences of which include negative publicity, competitive disadvantage and loss of consumer confidence (D'Arcy and Hovav, 2007). Traditional access control models are evidently deficient under certain conditions. For instance, a particular organization may necessitate access controls to be less prescriptive for the purposes of intra-organizational cooperation (Stevens and Wulf, 2002; Etalle and Winsborough, 2007). Traditional access controls, such as mandatory, discretionary or role-based access control, offer no protection for information that is unclassified and freely available in the public domain. In the recruitment industry, for example, information such as client lists and candidate lists has to be shared freely for purposes of collaborative job matching. As there are no controls over this information, an employee may download and distribute it to competitors.

On the opposing side of the continuum, organizations (e.g. hospitals) that manage highly sensitive information insist on stricter access control measures. Yet, traditional access controls may sometimes have an undesired effect in these circumstances. Consider as an example a nurse – at a hospital that has been isolated during a tornado – who needs access to a patient's records but cannot access them because nurses are not authorized to access this information (Povey, 1999). The

* *Corresponding author*. Tel.: +27 124296460.
E-mail addresses: padayk@unisa.ac.za (K. Padayachee), eloff@cs.up.ac.za (J.H.P. Eloff).
0167-4048/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2009.03.003

meaningful implementation of access control remains a difficult task and preconfigured access control policies may at times change dramatically in dynamic environments, depending on the context. Often, in unpredicted circumstances, users that are denied access could perhaps have prevented a catastrophe had they been allowed access. Moreover, the cost of implementing and maintaining complex preconfigured access control policies sometimes far outweighs the benefits. It has been proposed that auditing and accountability measures be enforced to deter unauthorized users, rather than to completely prevent them from gaining access (Etalle and Winsborough, 2007). This paper considers a proactive means of deterrence control to protect information that cannot be adequately or reasonably protected by access control.

The deterrence control approach is an application of optimistic access control. Optimistic access control is useful in cases where openness and availability are more important than complete confidentiality (Povey, 1999). Optimistic access control involves a combination of audit and accountability as deterrent mechanisms to encourage trustworthy behavior. This approach is characteristically more retrospective rather proactive. However, the application of usage control within an optimistic access control context may provide proactive means of deterrent control. Usage control enables finer-grained control over the use of objects than do traditional access control models (Sandhu and Park, 2003). Within traditional access control models, usage control would offer an extra layer of restriction against unauthorized usage. However, under the optimistic access control paradigm it would not restrict users but rather deter them from accessing and misusing information. As is defined in terms of the optimistic access control paradigm, the user must ultimately be able to access the required information.

This paper investigates the possibility of reformulating usage control in terms of the optimistic access control paradigm. Section 2 explores traditional access control models and presents the concept of optimistic access control. Section 3 elaborates on the usage control model and its applicability. Section 4 describes the adaptation of usage control as an application of deterrence control to enhance traditional access control. Furthermore, a possible technique to implement the model is proposed, using aspect-oriented programming. The objective is to seamlessly augment traditional access control with deterrence control. Section 5 concludes with directions for future work.

2. Background work on access control

Discretionary access control (DAC) is an access policy that restricts access to files and other system objects such as directories and devices on the basis of the identity of the users and/or the groups to which they belong (Russell and Gangemi, 1991). Discretionary access control is very flexible but highly vulnerable to Trojan Horses. As a result of this inadequacy, mandatory access policies are proposed. Mandatory access control (MAC) (Ramachandran et al., 2006) refers to access control policy decisions that are made beyond the control of the individual owner of the object. A central authority determines what information is to be accessible by whom, and the user cannot change access rights (Pfleeger, 1997). The most dominant model of recent times is the role-based access control model. Within role-based access control (RBAC), system administrators create roles according to the job responsibilities performed in a company. They grant permissions (access authorization) to those roles, and then assign users to the roles on the basis of their specific job responsibilities (Sandhu et al., 1996).

Access control models such as DAC, MAC and RBAC often assume what users want and are able to determine permissions before the actual access is made. They require *a priori* settings of permissions that are difficult to specify and maintain in highly dynamic environments where access policies may fluctuate on a regular basis. These types of control lack flexibility as they rely entirely on denying access. Access control models assume that human beings cannot behave in a trustworthy manner and the system has to prevent them from behaving in an undesirable manner. For example, within a typical mandatory access control model, *doctors* may have the privilege to access a patient's medical information, whereas *clerks* would have the privilege to access a patient's account information only. Unfortunately, this does not guarantee that an authorized user will in fact demonstrate integrity or act professionally with the designated information.

Optimistic access controls address this niche, where the access control is not preconfigured and the user is essentially trusted to behave ethically. While traditional access controls such as DAC, MAC and RBAC may be highly appropriate in certain contexts, optimistic access controls may be more appropriate in other circumstances (Padayachee and Eloff, 2007). A field study conducted by Stevens and Wulf (2002), who considered the cooperation between two engineering offices and a steel mill, is a case in point. Within this real-world inter-organizational cooperation scenario, it was found that traditional access controls did not comply with the organization's requirements and that cooperation and competitive reasons motivate the use of interactive and optimistic access controls (Stevens and Wulf, 2002). A posteriori policy enforcement offers interoperability, flexibility and scalability, which is crucial in collaborative environments (Etalle and Winsborough, 2007). Moreover, the call for privacy-sensitive systems to have a range of control and feedback mechanisms for building pessimistic, optimistic and mixed-initiative applications has also been recognized (Hong and Landay, 2004).

The optimistic access control scheme allows users to exceed their normal privileges in a way that is constrained so that it is securely audited and may be rolled back (i.e. the system is restored to its original state before the breach). As optimistic accesses are subject to *ex ante* controls to ensure that the organization's security policies are maintained, it is contingent on administrators to detect unreasonable access and take steps to compensate for the action. Such steps might include the following (Povey, 1999):

- Undoing illegitimate modifications
- Taking punitive action (e.g. firing or prosecuting individuals)
- Removing privileges

Optimistic access controls trust human beings to perform legitimate accesses and take retrospective action after such trust has been breached. The initial cost of implementing optimistic access control methods is minimal as there are no clearly defined mechanisms for restricting unauthorized users. However, the consequences of a breach in trust could be disastrous. If such a breach is discovered, it could involve prosecution or require the performing of a roll-back procedure. Although the roll-back procedure may be able to restore the system to its original state prior to the breach, it is highly likely that it may not be able to undo the damage done. Padayachee and Eloff (2007) propose that this type of access control should be supplemented with usage control to ensure that humans behave ethically. However, the proposed model had limited appeal as it suggested that all information should be subject to optimistic access control. It did not consider the real-world context where traditional access control measures are also obligatory. In this paper, we expand this concept to constitute a model for deterrence control as an enhancement of traditional access control.

3. Background on the usage control (UCON)

Usage control (UCON) is intended to address the inadequacies of traditional access control. Consequently, there has been a trend towards complementing access control methods such as role-based access control with usage control (see Li et al., 2005; Xu et al., 2003). The UCON model encompasses emerging applications such as trust management in a unified framework. It considers the missing components of traditional access control, such as the concepts of *obligations* and *conditions*. *Obligations* require some action by the subject (user) so as to gain or sustain access, e.g. by clicking the ACCEPT button on a license agreement or agreeing not to distribute a confidential document. *Conditions* represent system-oriented factors such as time-of-day, where subjects are allowed access only within a specific time period. With traditional access control, authorization is assumed to be done before access is allowed (pre). However, the UCON model extends pre-authorization by re-evaluating usage requirements throughout usages. This property is called ''continuity'' and has to be captured in modern access control for the control of relatively long-lived usage or for immediate revocation of usage. The 'continuity' property implies that access may be revoked instantaneously during access. Hence ongoing authorization is active throughout the usage of the requested right, and is repeatedly checked for sustaining access. Technically, these checks are performed periodically based on time or events. For instance, suppose an ongoing obligation condition stipulates that a window declaring the '*Terms and Conditions of Use*' should remain open during access. Accordingly, if the user ignores this stipulation and closes the window during access, then the usage is revoked immediately (Sandhu and Park, 2003).

Usage control is relevant in many contexts, including privacy, Digital Rights Management, management of Internet Protocols and that of trade and administration secrets (Pretschner and Walter, 2008). One of the motivations for applying usage control is that it considers ongoing controls for extended access or for revocation. For example, Zhang and Nakae (2006) used UCON as the access control model for collaborative systems by leveraging the features of decision continuity and attribute mutability. They claim that traditional access control approaches do not consider the usage status of a shared object in authorization. They developed a prototype and found that the main overhead of the system introduced by usage control involved mutable attribute acquisitions, policy interpretations and evaluations, and the updates of mutable attributes. Wang et al. (2006) also motivated the use of the UCON model for extended access, as it would be useful in ubiquitous environments where the information can be accessed anywhere and at any time, which is potentially unsafe. The ongoing continuity for authorizations, obligations and conditions found in the UCON model can be used to control objects in a dynamic environment, since they provide more robust access control for ubiquitous computing environments and can protect sensitive messages from dissemination.

Although the UCON model is comprehensive, it has been extended in several ways. For instance, according to Lee et al. (2004) this framework lacks an important component in terms of access control. Lee et al. (2004) maintain that the element of '*consent*' should also be included in an access control system, thereby increasing society's trust towards a software system. They consider consent to be diametric to the 'concept of obligation' within the Usage Control Model and state that '[w]hile the obligation is obeyed by the customer, *consent* is observed by the provider' (Lee et al., 2004). The proposed method can extend the coverage of the UCON model in a security area, and will enhance the right of the provider and customer. It also provides a solution for trust relationships on e-Commerce and protection for individual privacy. In a position paper, Pretschner and Walter (2008) considered usage control in the context of distributed systems that are composed of different actors taking the roles of data providers (who give data away) and data consumers (who request and receive data). In this position paper, they considered the element of negotiation for usage control. The term negotiation suggests that multi-step bidirectional communication takes place. Shin and Yoo (2007) extended the usage control (UCON) model by adding a further component, namely *delegation*, for the effective modeling of delegating access rights in ubiquitous computing.

Sandhu and Park (Sandhu and Park, 2003; Park and Sandhu, 2004) have expanded usage control into a family of models for usage control, involving pre-authorizations and ongoing authorizations. The implementation of pre-authorization is relatively simple, as it warrants checking the conditions and obligations before the user may proceed. However, the implementation of ongoing authorization is non-trivial. Sandhu and Park (Sandhu and Park, 2003; Park and Sandhu, 2004) furthermore do not offer a proposition towards how ongoing authorizations may be implemented. In an previous paper (Padayachee and Eloff, 2007) we proposed the use of multithreading to implement ongoing authorizations. If a subject (user) requests an object (such as a file), the preconditions and pre-obligations are checked, then two separate threads are invoked representing ongoing conditions and ongoing obligations respectively. The ongoing obligations and ongoing conditions will be tested intermittently during the access. In the next section, the model for usage control is adapted under the optimistic access control paradigm. The mechanisms for pre-obligations, pre-conditions, ongoing obligations and ongoing conditions are used as mechanisms of deterrence control.

4. Adapting usage control as a mechanism for deterrence control

According to Jones and Rastogi, security controls may be found in one of four categories: *corrective control*, *deterrent control*, *detective control* and *preventative control*. Access control falls in the preventative control category. Typically, information under this protection is secured in terms of roles or attributes, whereas information in the public domain is not. Two aspects are at issue in the securing of a distributed computing environment against malicious or otherwise disruptive use: a social aspect, where the safeguarding of a computer system relies on social deterrents such as shameful exposure or prosecution, and a technical aspect, where the system is protected by technical means such as encryption algorithms and access controls (Georgiev and Georgiev, 2001). Detective functions attempt to identify unwanted events even as they are occurring or after they have occurred. Deterrent controls are intended to discourage individuals from intentionally violating information security policies or procedures. Typically, organizations implement deterrents such as anti-virus systems and passwords, or by fostering security awareness. Recovery controls restore lost computing resources or capabilities and help the organization to recover monetary losses caused by a security violation. Corrective controls either remedy the circumstances that allowed the unauthorized activity or revert conditions to what they were before the violation (Kim and Leem, 2004).

4.1. *A motivating example*

Suppose company ABC is an e-recruitment company where clients and prospective candidates (job seekers) place job orders and applications online. Company ABC then maintains databases of candidates and clients. While internally the company places access controls on sensitive information such as salaries, this information is unrestricted for purposes of collaborative job matching. Suppose an employee of company ABC decides to download all the telephone numbers that are available on these databases and sell it to a telemarketer. Due to the lack of deterrents, this act is relatively easy to carry out. Furthermore, the employee may claim that he was unaware of the fact that his actions were unethical. This type of security breach is usually blamed on the user and on a lack of user training. Such an understanding of a security breach is contradictory to the requirements of security usability. According to Zurko (2005) we have to ask, 'why did the system make the insecure option so easy and attractive' and in this case ultimately lucrative? Perhaps if system provided for synchronous deterrents, this employee may have been deterred from carrying out the act.

So, in the above scenario, the following stipulations as mechanisms could be used as usage control deterrents:

Pre-obligation: The user must click on a button in a window, thereby indicating that he/she agrees not to distribute this information.
Ongoing obligation: A window with the following warning "This data set must be used EXCLUSIVELY for work-related purposes" is to remain open at all times.
Pre-condition: This information may be accessed during business hours only.
Ongoing condition: This information may be accessed during business hours only (same as pre-condition, as it is time dependent). While the pre-condition may have been valid at the time of access, pre-condition may become invalid during the access.
Post-obligation: The user must send an e-mail to the administrator if he accessed these databases outside of business hours.

The post-obligation implies that an employee may in fact access the databases after hours. Under the optimistic paradigm, the employee should ultimately be able to download the data in the case of an emergency. This is permitted as the employee should not be prevented from performing his/her duties. In the following section we consider how such deterrents may be implemented practically under the optimistic paradigm.

4.2. *Design*

Fig. 1 shows the activity diagram for usage deterrence. When a user requests access to an object, authorization is performed utilizing subject and object information (attributes). Usage rules are used to check whether the request is allowed or not and whether the data is classified and subject to access control. If the data is in the public domain and hence unprotected by access controls, then the usage deterrence mechanisms are deployed. Otherwise the access control proceeds as expected with traditional access control based on user attributes.

The `Pre-Conditions` and `Ongoing Conditions` activities decide whether conditional requirements are satisfied by using the contextual information (such as current time, IP address, etc.) before and during the access. The `Pre-obligation`, `Ongoing Obligation` and `Post Obligation` activities decide whether certain obligations have to be performed or not – either before, during or after the requested usage has been executed. If there exists any post-obligation that has to be performed, this must be monitored and the result has to be resolved by the `Post Update` activity. If the user does not accept the `Pre-Obligations`, he/she is not allowed to access the information at all. If the user does not accept the `Ongoing Obligations` while accessing the information, then access is ceased. These problems are not considered as breaches. However, if the user has accessed the information and refuses to accept the post-obligations, this is considered to be a breach of trust. Here the user will be penalized and he/she will be not trusted to access this information in the future. While the user will not be permitted to access the information unless the obligations are satisfied, he/she will under special circumstances be allowed to access it by utilizing the `Break the Glass` facility.

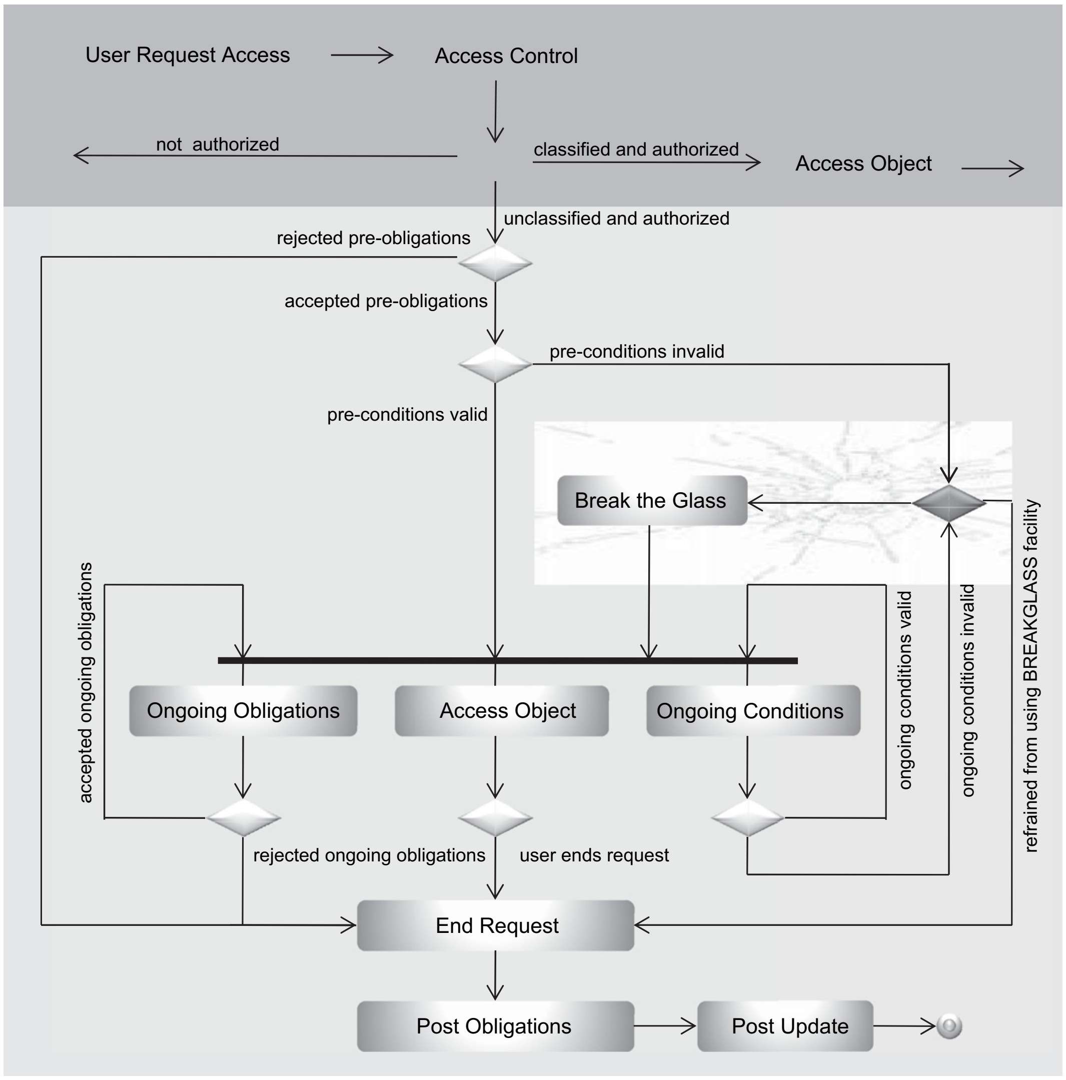

Fig. 1 – Adapting usage control with respect to deterrent control.

The `Break the Glass` facility is a novel addition to the usage control model. While it is a facility to deter the user from proceeding with access when the conditions are not satisfied, it provides a mechanism to override the system. As in Ferreira et al. (2006), the user is informed about the consequences of accessing this information illegitimately and the access is red flagged for auditing purposes. The difference here is that the `Break the Glass` facility is deployed when the system conditions are not satisfied. Essentially the `Break the Glass` facility is for emergencies only. During auditing, the `Post Update` process will also be referenced to identify unfulfilled post-obligations. If the user cannot justify an illegal access or an unfulfilled post-obligation, then the access policy will be updated to restrict this user's access rights to public domain information in the future. This may also involve punitive action. If the user has performed some illegal modification to the data, the roll-back procedure will be attempted to return the data back to its original state. Immediate revocation of usage occurs in two instances: – firstly, if the user ignores the stipulation of the `Ongoing Obligations`, and secondly, if the user does not deploy the `Break the Glass` facility when the `Ongoing Conditions` are no longer satisfied.

In terms of the enforcement of security policies, it is imperative that it is centrally located and enforced uniformly. Accordingly, the same notion would apply to the implementation of such policies in terms of application logic (Verhanneman et al., 2005). This type of deployment may be achieved through the use of aspect-oriented methodologies. The premise of the model is to create an aspect that will intercept calls when a subject requests access to an object and enforce deterrence control where access control is not imposed. A significant amount of work has been conducted on aspect-oriented security in respect of access control. It has been shown that aspect-oriented programming eases the implementation of security type concerns such as access control (De Win et al., 2001). It results in an implementation that is easier to maintain and port to different environments. Many recent systems are based on a three-tiered architecture – access is via the web, the application programs reside within an application server, and the data is stored within a database system (Li et al., 2005). However, only the application layer is considered in the next section.

4.3. *An aspect-oriented approach to deterrence control*

An aspect is a modular unit of a crosscutting implementation that is provided in terms of pointcuts and advices. It specifies what (advice) and when (pointcut) its code is going to be executed (Ortin and Cueva, 2004). In terms of codification, aspects are similar to objects. However, aspects observe objects and react to their behavior (Viega and Voas, 2000). An aspect is a piece of code that describes a recurring property of a program and can span multiple classes, interfaces or aspects (Choi, 2000). Unlike a class though, aspects are injected into other types. Aspects improve the separation of concerns by making it possible to cleanly localize crosscutting design concerns. They also allow programmers to write, view and edit a crosscutting concern as a separate entity. During program execution, there will be certain well-defined points where calls to aspect code would be inserted (Ortin and Cueva, 2004). These are known as join points. Aspects introduce their supplemental functionality at these join points (Viega and Voas, 2000). A pointcut is a set of join points described by a pointcut expression. An advice declaration is used to specify code that should run when the join points specified by the pointcut expression are reached (Mahrenholz et al., 2002). The advice code will be executed when a join point is reached, either before or after the execution proceeds. For example, AspectJ supports *before*, *after* and *around* advices, depending on the time the code is executed (De Win et al., 2002). A *before* (*after*) advice on a method execution defines code to be run before (after) the particular method is actually executed. An *around* advice defines code that is executed when the join point is reached and has control over whether the computation at the join point (i.e. an application method) is allowed to be executed or not (Kiczales et al., 2001). Combining the application functional code and its specific aspects generates the final application. These two entities will be combined at compile time by invoking a special tool called a 'weaver' (Choi, 2000).

Consider the following class that controls user accesses to features. A feature could be "View File c:\candidate.txt" or "Edit Report c:\AnnualReport.txt".

```
public class WebSecurityManager
{
    public WebSecurityManager(){}
public static void init(String sApplicationName)
throws
WebSecurityManagerException{
    //initial WebSecurityManager
    }
public static final Boolean checkPermission(User user,
Feature feature){
    //determine if user is allowed access
}
public void request(User user, Feature feature){
    //allow user to request feature
}
}
```

With aspect-oriented programming, we can augment `WebSecurityManager`, which has methods for traditional access control with deterrence control without modifying the class. Further, all details relating to deterrence control can be confined in a singular modular structure, namely an aspect, without it being mixed in with the `WebSecurityManager` class. To accomplish this, a generic aspect `DeterrenceControlInjector` was defined. This aspect delineates three pointcuts. The first `pointcut` **`InterceptCheckPermission`** intercepts those calls where a user requests access to a feature, i.e. during program execution calls to `checkPermission` will be intercepted. The **`after`** advice defines code that is executed after the `checkPermission` method is called. The advice initially determines whether this information is in the public domain (i.e. not protected by the traditional access control facility), otherwise it allows traditional access control to proceed as usual. This advice contains operations to test the `pre-Obligations` and `pre-Conditions`. If the `pre-Obligations` are not satisfied, the user is not allowed to access the feature. If the `pre-Conditions` are invalid, the user has the opportunity to use the `Break the Glass` facility to access the feature. The aspect invokes threads to maintain the ongoing conditions and ongoing obligations to control the request to use the feature.

```
public aspect DeterrenceControlInjector {

private Thread conditionsThread;
private Thread obligationsThread;
private Thread AspectWebSecurityThread;
private WebSecurityManager SecurityManager;
private Boolean authorized;
private User user;
private Feature feature;
pointcut    InterceptCheckPermission(User    user,
Feature feature):
call(* *.checkPermission(..)) &&!within(Deterrence
ControlInjector)
     && args(user,feature)
after (User user, Feature feature)
returning (Boolean authorized):
InterceptCheckPermission(user,feature,WM){
if (isPublicDomain() && authorized){
  if (pre_Obligations(user,feature)){
     if (pre_Conditions(user,feature)
     || BreakTheGlass(user,feature)){
         Conditions  conditions  =  new  Conditions
(user,feature);
         conditionsThread = new Thread(conditions);
         conditionsThread.start();
         Obligations obligations = new Obligations
(user,feature);
         obligationsThread = new Thread(obligations);
         obligationsThread.start();
         AspectWebSecurityThread    =    new    Thread
(SecurityManager);
         AspectWebSecurityThread.start();
         SecurityManager.request(user,feature);
     }
  }
}}
pointcut OngoingConditions(Conditions conditions):
call(* *.warning()) && target(conditions);
after(Conditions conditions): OngoingConditions
(conditions){
if (!BreakTheGlass(conditions.getUser(), conditions.
getFeature())){
  conditions.stop();
  System.exit(0);
}}
pointcut OngoingObligations(Obligations obliga-
tions): call(* *.stop()) && target(obligations);
after(Obligations obligations): OngoingObligations
(obligations){
    SecurityManager.endrequest();
    PostObligations(obligations.
getUser(),obligations.getFeature());
    System.exit(0);
}
Boolean pre_Obligations(User user, Feature feature){
    //determine pre-Obligations for user to access
feature
}
Boolean pre_Conditions(User user, Feature feature){
    //determine pre-conditions for user to access
feature
}
Boolean PostObligations(User user,Feature feature){
    //determine postObligations for the user to access
feature
}
Boolean BreakTheGlass(User user, Feature feature){
    //present the break the glass option to user
}
void LogAccess(User user,Feature feature,String
s,String a){
    //log accesses
}
Boolean isPublicDomain(){
    //inspect usage control policies
}
}
```

The next two pointcuts, namely pointcut **OngoingConditions** and pointcut **OngoingObligations**, intercept execution points that indicate that the ongoing conditions and ongoing obligations are no longer satisfied. The *after* advices of each pointcut define code that is executed after such an irregularity has been detected. In this case, if some action results in the warning or stop method being called on either the conditions object or the obligations object, then this call will be intercepted by these pointcuts. If the ongoing obligations are no longer satisfied, then the access is terminated immediately. If the ongoing conditions are no longer satisfied, the user has the opportunity to use the Break the Glass facility to continue with the access.

In order for the immediate revocation of access to occur, the object that is responsible for allowing access needs to now terminate the access. If we consider the WebSecurityManager class, we need to extend it, to control users' requests to a feature and to end requests. Fortunately, aspect-orientation permits a seamless integration of this additional functionality by facilitating the creation of a special aspect known as an *intertype declaration* without modifying the WebSecurityManager class. The *intertype declaration* construct is supported by aspect-oriented programming languages such as AspectJ. An *intertype declaration* is generally used to add on information such as methods or fields to an object without modifying the existing class. Here, methods to control requests and to end requests to information were added. Furthermore, as this process needed to be controlled within thread, in Java this implies that this class must implement the java.lang.Runnable thread interface. With AspectJ, this can be done using the declare parents syntax so that WebSecurityThread can be the active object.

```
public aspect AspectWebSecurityThread {
Feature feature;
User user;
private Thread WebSecurityManager.myThread;
declare parents: WebSecurityManager implements
Runnable;
public void WebSecurityManager.run() {
//implement run for WebSecurityManager
}
public void WebSecurityManager.stop() {
    myThread = null;
}
public void WebSecurityManager.request(User user,
Feature feature){
    //control user's request to feature
}
public void WebSecurityManager.endrequest(){
    stop();
}
}
```

Augmenting traditional access control with usage control features will slow down program execution, as it involves the inclusion of additional code to the functional system. In terms of usability, controls such pre-obligations and ongoing obligations may be distracting and impact negatively on the productivity of users. Perhaps as the user becomes more "trustworthy", some obligations or conditions may be relaxed or negotiated. The cost of implementing usage control as a deterrent may have to be weighed against the cost of information misuse. South Africa's draft bill on the protection of personal information is viewed as a means of ensuring South Africa's future participation in the information market by providing 'adequate' information protection of an international standard (see Discussion Paper 109). If individuals are ensured that their privacy is taken into account in a software system, it is understandable that these individuals will trust the system with their private information. The survival of e-business will probably depend on its ability to ensure the privacy of its clients.

5. Conclusions and future work

The element of trust within deterrence control warrants an investigation into human behaviors and responses to its application. It would be pragmatic to investigate whether the model presented here does in fact dissuade individuals from accessing and misusing information in the public domain. To ease the development and maintenance of optimistic usage control measures, it is posited that they be completely separated from the application logic by using the aspect-oriented paradigm. The aspect designed has not been tested within a real-world context. However, confining all the operations pertaining to deterrent control to a single modular structure will ease both development and maintenance costs as it can be integrated seamlessly into a system based on traditional access control. Future studies would involve conducting a case study to test the usability and performance issues of the approach presented here.

The proposed solution may well ease the burden of system administrators significantly. It is rather difficult for administrators to predict all of the possible usage scenarios and thus all of the necessary permissions. With deterrence control, it is ultimately left to users to make that judgment. Consequently, the complexity of implementing and maintaining preconfigured access control policies is shifted to the way the user interacts with the system. Adapting usage control with respect to deterrence control provides a proactive mechanism in addition to the retroactive methods of auditing and accountability. Hence, with a proactive means of deterrent control, a larger subset of information may be relegated into the public domain.

Acknowledgements

The code for WebSecurityManager class was derived from the "Secure a Web application" article that was created by M Cymerman, [http://www.javaworld.com/javaworld/jw-04-2000/jw-0428-websecurity.html?page=1] however it has been customized to fit the current scenario. We also like to acknowledge the use of the Eclipse platform and AspectJ (http://www.eclipse.org/aspectj/) during the development process.

REFERENCES

Choi JP. Aspect-oriented programming with enterprise JavaBeans. In: Fourth international enterprise distributed object computing conference (EDOC'00). Makuhari, Japan; 2000. p. 252–61.

D'Arcy D, Hovav A. Deterring internal information systems misuse. Communications of the ACM 2007;50(20):113–7.

De Win B, Joosen W, Piessens F. Developing secure applications through aspect-oriented programming. In: Aksit M, editor. Aspect-oriented software development. Boston: Addison-Wesley; 2002. p. 633–50.

De Win B, Vanhaute B, Decker B. Security through aspect-oriented programming. In: Advances in network and distributed systems security, IFIP TC11 WG11.4 first working conference on network security, 2001, Leuven, Belgium. Boston: Kluwer Academic Publishers; 2001. p. 125–38.

Discussion paper 109 (Project 124), http://www.ispa.org.za/regcom/privacyfiles/chapter-9-draft-bill-protection-personal-info.pdf; 2006.

Etalle S, Winsborough WH. A posteriori compliance control. In: SACMAT '07: proceedings of the 12th ACM symposium on access control models and technologies. Sophia Antipolis, France: ACM; 2007. p. 11–20.

Ferreira A, Cruz-Correia R, Antunes L, Farinha P, Oliveira-Palhares E, Chadwick DW, et al. How to break access control in a controlled manner. In: Proceedings of the 19th IEEE international symposium on computer-based medical systems. Salt Lake City, UT; 2006. p. 847–51.

Georgiev IK, Georgiev II. A security model for distributed computing. Journal of Computing Sciences in Colleges 2001; 17(1):178–86.

Hong JI, Landay JA. An architecture for privacy sensitive ubiquitous computing. In: MobiSys '04. Boston, MA, USA; 2004. p. 177–89.

Jones RL, Rastogi A. Secure code: building security into the software development life cycle. Information Security Journal: A Global Perspective 2004;15(5):29–39.

Kiczales G, Hilsdale E, Hugunin J, Kersten M, Palm J. Getting started with AspectJ. Communications of the ACM 2001;44(10): 59–65.

Kim S, Leem CS. An information engineering methodology for the security strategy planning. In: Computational science and its applications – ICCSA 2004, Assisi, Italy. Berlin/Heidelberg: Springer; 2004. p. 597–607.

Lee G, Kim W, Kim D-K. Novel method to support user's consent in usage control for stable trust in E-business. Lecture Notes in Computer Science 2004;3045:906–14.

Li X, Naeem NA, Kemme B. Fine-granularity access control in 3-tier laboratory information systems. In: Database engineering and application symposium, IDEAS. Montreal, Canada; 2005. p. 391–7.

Mahrenholz D, Spinczyk O, Schröder-Preikschat W. Program instrumentation for debugging and monitoring with AspectC++. In: Proceedings of the 5th IEEE international symposium on object-oriented real-time distributed computing. Washington, DC, USA; 2002. p. 249–56.

Ortin F, Cueva JM. Dynamic adaptation of application aspects. Journal of Systems and Software 2004;71(3):229–43.

Padayachee K, Eloff JHP. Enhancing optimistic access controls with usage control. In: Lambrinoudakis C, Pernul G, Tjoa AM, editors. Trust, privacy and security in digital business. Regensburg, Germany: Springer; 2007. p. 75–82.

Park J, Sandhu R. The UCON_ABC usage control model. ACM Transactions on Information and System Security 2004;7(1): 128–74.

Pfleeger CP. Security in computing. 2nd ed. United States of America: Prentice Hall; 1997.

Povey D. Optimistic security: a new access control paradigm. In: Proceedings of the 1999 workshop on new security paradigms. Caledon Hills, Ontario, Canada; 1999.

Pretschner A, Walter T. Negotiation of usage control policies – simply the best? In: ARES '08: proceedings of the 2008 third international conference on availability, reliability and security. Washington, DC, USA: IEEE Computer Society; 2008. p. 1135–6.

Ramachandran R, Pearce DJ, Welch I. AspectJ for multilevel security. In: The 5th AOSD workshop on aspects, components, and patterns for infrastructure software (ACP4IS). Bonn, Germany; 2006. p. 1–5.

Russell D, Gangemi GT. Computer security basics. Sebastopol, CA: O'Reilly and Associates; 1991.

Sandhu R, Park J. Usage control: a vision for next generation access control. In: The second international workshop on mathematical methods, models and architectures for computer networks security. St Petersburg, Russia; 2003. p. 17–31.

Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. IEEE Computer 1996;29(2):38–47.

Shin W, Yoo SB. Secured web services based on extended usage control. In: PAKDD 2007 workshops. Lecture notes in artificial intelligence, vol. 4819; 2007. p. 656–63.
Stevens G, Wulf V. A new dimension in access control: studying maintenance engineering across organizational boundaries. In: Proceedings of the ACM conference on computer supported cooperative work (CSCW). New Orleans, LA, USA; 2002.
Verhanneman T, Piessens F, De Win B, Joosen W. Uniform application-level access control enforcement of organizationwide policies. In: Proceedings of the 21st annual computer security applications conference (ACSAC 2005); 2005.
Viega J, Voas J. Can aspect-oriented programming lead to more reliable software. IEEE Software 2000;17(6):19–21.
Wang H, Zhang Y, Cao J. Ubiquitous computing environments and its usage access control. In: InfoScale '06: proceedings of the 1st international conference on scalable information systems. Hong Kong: ACM; 2006. p. 6–16.
Xu Z, Feng D, Li L, Chen H. UC-RBAC: a usage constrained role-based access control model. Lecture Notes in Computer Science 2003; 2836/2003:337–47.
Zhang X, Nakae M. A usage-based authorization framework for collaborative computing systems. In: Symposium on access control models and technologies (SACMAT'06). Lake Tahoe, CA, USA; 2006. p. 180–9.
Zurko ME. User-centered security: stepping up to the grand challenge. In: Annual computer security applications conference 2005. Tucson, AZ, USA; 2005.

Keshnee Padayachee is a Lecturer in the School of Computing at the University of South Africa. She holds a masters degree in Computer Science from the University of KwaZulu Natal in South Africa.

Ms Padayachee's main areas of interest are Aspect-Oriented Programming and its relevance to information security.

Jan Eloff received a PhD (Computer Science) from the Rand Afrikaans University, South Africa. He gained practical experience by working as management consultant specialising in the field of information security. Since October 2002 he is Head of Department and full professor in Computer Science at the Department of Computer Science, University of Pretoria. Prior to that he has was a full professor in Computer Science at the Rand Afrikaans University.

He is an expert representative from South Africa on Technical Committee 11 (Information Security) of the International Federation for Information Processing (IFIP). In 2001 he received the IFIP Silver Core and Outstanding Services Award for his long-term services to IFIP. He also serves as the South African representative on ISO (International Standards Organization) contributing to the development of computer and information security standards. In October 2004 he was elected as the President of the South African Institute of Computer Scientists and Information Technologists (SAICSIT).

He has published extensively in a wide spectrum of accredited international subject journals. Many acclaimed international and national conferences were organised and conducted under his leadership. He delivered papers at leading information security conferences on an international level.

He is an evaluated researcher from The National Research Foundation (NRF), South Africa. He is a member of the Council for Natural Scientists of South Africa.

He is an annually invited guest professor at the International Institute of Management in Telecommunications, University of Fribourg, Switzerland.

He advises to industry on various information security projects.