

Preventing Information Leakage in Mobile Applications with Object-Oriented Access Control Lists and Security Monitor Encapsulation

SHIOW-YANG WU AND SHIH-CHIEN CHOU

*Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien, 974 Taiwan*

We propose a model and associated algorithms for information flow control to prevent information leakage in mobile computing environments. The model employs access control lists and encapsulated security monitors under a fully object-oriented framework. We show that our model prevents unauthorized direct access to sensitive information from a mobile user to the server, as well as any attempt on indirect access through intermediate entities. To understand the feasibility of our model, we suggest an event-driven approach and efficient implementation for the realization of the model. A Java-based preliminary implementation and performance evaluation results demonstrate that our model can successfully prevent information leakage with very low overhead.

Keywords: mobile data access, information flow control, access control lists, encapsulated security monitor, information leakage

1. INTRODUCTION

With the proliferation of handheld devices and wireless communication technologies, mobile information services are becoming more and more important in recent years. In a mobile environment, a user holding a *client device* (we use the term PDA as a representative device henceforth) in an area covered by a cell is served by the server (*i.e.*, base station) of the cell. In offering information services, there are security issues to resolve. Consider the following scenario. A man travels through the east coast of Taiwan. Whenever necessary, he uses a PDA to download information regarding hotels, rest areas, mountains, beaches, *etc.* during the trip. Due to the geographical characteristics of the area, the user can encounter three types of situations. Most parts of the coast are opened to the public so anyone can enjoy the beach. Around the coast near the airport, however, is restricted to authorized ground personnel only. The area near the military airbase, on the other hand, is strictly prohibitive against civilian access. In general, the trip may cover areas of different security levels such as public areas, loosely military-controlled areas, strictly military-controlled areas, and so on. Users of different roles can access information of different security levels in different areas. That is, what a PDA can download is determined by the *security role* of the PDA holder (which is termed “*PDA role*”). For example, the PDA role “officer” may download all information in every area. A PDA role “ground crew” may download all information of the airport but not that of

Received November 26, 2007; revised March 25 & May 14, 2008; accepted June 12, 2008.
Communicated by Tzong-Chen Wu.

the military airbase. All “civilian” PDA roles can only download recreation related information in all areas. In the example above, the following security issues should be properly addressed:

- Information leakage from a server to a PDA should be prevented. Information should only be offered to PDAs according to the PDA roles.
- Information leakage among PDA roles should be prevented. This corresponds to preventing Trojan horses [23, 24]. For example, an officer intentionally leaking information of an air base to a civilian is regarded as a Trojan horse.

Fig. 1 demonstrates the desired access patterns that a secure information repository for mobile applications should be capable of handling. We designed an object-oriented model with integrated access control lists (ACLs) and encapsulated security monitors to solve the issues. The basic approach of the model can be described as follows:

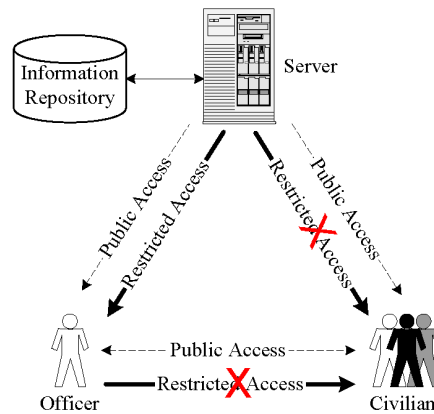


Fig. 1. Access model.

- An ACL is attached to each object encapsulating sensitive information that can only be accessed by a restricted set of PDA roles. If a PDA role is allowed to access the information, the object is allowed to be downloaded to the PDA. Otherwise the request is rejected. It is important to note that we always download the entire object encapsulating the sensitive information instead of downloading just the information.
- An ACL is attached to each object method that accesses sensitive information. If a PDA role is allowed to access the information, the object method can be executed by the PDA holder. During the execution, the method presents the sensitive information to the PDA holder.
- An encapsulated security monitor is associated with each object possessing sensitive information. The monitor is used to guard against any malicious process that may lead to information leakage among PDA roles.

By providing formal characterization of our model and detailed account of the security monitoring algorithm, we show that the proposed model can successfully prevent information leakage in mobile applications.

In addition to solving the security issues mentioned above, we also address the performance issue regarding the implementation of the proposed model. The performance may be hindered because: (a) the entire object encapsulating the required information (which is of larger size than the information itself) is downloaded and (b) the security checking (*i.e.*, checking the ACLs) consumes extra time. We devise an event-driven architecture and a prototype implementation for the evaluation of our model. The evaluation results demonstrate that the desired level of security can be achieved with very low overhead. In the following sections, we first describe research work closely related to our approach, then describe the model and present the evaluation results.

2. RELATED WORK

Our work is along the line of research on access control, which is an important issue in information security [12, 14, 30, 33]. The issue has long been recognized as essential in operating system protection [35] and database security [36]. Recently, access control is also recognized as important in controlling the security of information flows within an application [3, 4, 11, 23, 24], ensuring workflow [17, 26, 37] and web application security [29]. Our research provides object-based access control to mobile applications.

Many access control methods have been proposed in the past including discretionary access control (DAC) [20, 31], mandatory access control (MAC) [2, 5, 28], label based access control [3, 4, 22], role-based access control (RBAC) [1, 5, 9, 10, 25, 28, 32], and so on. In the access control matrix (ACM) approach [15, 34], a subject can access an object if the required access right is recorded in the ACL. MAC is a kind of multiple-level access control approach that uses a lattice to categorize security levels [2, 5]. Access control in MAC generally follows the “no read up” and “no write down” rules [2]. MAC has been criticized as too restricted [21]. Therefore, other approaches were proposed to loosen the restriction of MAC. For example, the label based approach [22-24] attaches labels to sensitive information that should be protected, in which the purpose of a label is similar to that of an ACL. As another example, RBAC has recently been recognized as an important approach to access control because it is a super set of DAC and MAC [25, 27, 28] (remember that DAC and MAC are all useful in access control). Kabra *et al.* tackled the problem of fine-grained access control of SQL queries by proposing redundancy reduction techniques for access control predicates and methods for generating safe query plan to prevent information leakage [16]. In general, the problem of secure information flow should be formally characterized using a model or any formalism such as the logic TIFL proposed by Lanotte *et al.* [19]. Proper access control methods can then be designed with theoretical guarantee against information leakage.

Features offered by an access control model for different usage are different. For example, access control within object-oriented systems should offer the features of avoiding Trojan horses [23, 24], declassification [23, 24], purpose-oriented method invocation [40], precisely controlling information flows among objects [5], and so on. As another example, access control in workflow should offer the features of protecting resources, roles, and sensitive information [6, 17]. Security in mobile computing environment is highly challenging in general [8]. An access control model for mobile applications, in particular, should offer the features mentioned in section 1. Moreover, perform-

ance in mobile applications is essential because PDAs may move from cell to cell quickly. Therefore, access control models applied in mobile applications should not downgrade the overall performance too much. Several models have been proposed to provide a secure environment for mobile applications. Kwok and Lau proposed a novel approach for up-link access control in mobile environments [18]. Wang *et al.* presented ticket-based methods for secure M-services [39]. Villate, Illarramendi and Pitoura proposed a new service named *Data Lockers* to provide a safe and highly available ubiquitous persistent storage space for mobile users [38].

Our work differs from previous methods in several respects. We propose a model for characterizing the information flow problem in mobile applications. Our model does not rely on any security infrastructure to work properly. The ACLs, security monitors and operating patterns adhere closely to the principle of object orientation and successfully prevent information leakage from server to clients as well as among clients of different security levels. Finally, from the prototype implementation and performance evaluation results presented in section 4, our approach can be effectively realized with high scalability and very low overhead.

3. SYSTEM MODEL

In this section, we first describe the assumptions about the environment and system behavior then discuss the threat model to characterize the adversary, followed by a formal description of our model and algorithm, and end with analysis of behavior to show that the model offers the desired level of security mentioned in section 1.

3.1 Assumptions

In designing our approach, we make the following assumptions with the purpose of delimiting the scope of our model.

- We do not consider the security issues on low level wireless transmission. Our model is an application level model that handles information flow security among mobile applications. Therefore, we assume that the wireless communication subsystem takes care of transmission issues such that all objects transmitted are either received intact or dropped entirely in case of any problem.
- The server of a cell can automatically detect the PDA role of its owner whenever the user enters the area covered by the cell. This can be achieved quite easily with the current cellular network technology.
- A server will not broadcast sensitive information in a restricted area. In other words, sensitive information can only be accessed through downloading the object within which the information is encapsulated.
- We do not consider the security issues caused by virus or hacker programs embedded in either the servers or PDAs. More specifically, we assume that the object-oriented execution of all application objects is not tempered in any way. This is primarily a separation of concerns. It is supposed to be the responsibility of virus detection tools.
- Information leakage among PDA holders (through, for example, verbal communication or hand writing on papers) is out of the scope of the paper. Our model prevents the

leakage of information transferred through communication.

- A server does not need to know the applications and objects running on a PDA which may be manufactured by different vendors. A good model must be able to tolerate the heterogeneity of client devices and platforms.
- The interfaces between servers and PDAs are well known by both parties.

We note that these assumptions are to make our argument rigorous. They represent security domains that can either be readily handled by other mechanisms or out of the scope of the paper. They do not constitute any obstacles on the application of our model in the open environment.

3.2 The Threat Model

Information security is complicated and therefore solving all security issues in a model is impossible. In designing our approach, we make the following assumptions with the purpose of delimiting the scope of our model.

Before the design of any security techniques, it is necessary to recognize the vulnerabilities of the system being designed. An effective way to start with is to clearly identify the adversaries and possible attacks. In other words, we need to define a threat model as the target of our security measure. In this section, we characterize various types of potential threats and attacks to the system. We also define the trusted computing base (TCB) upon which the development of our security model can depend. Based on the analysis, we can easily identify the focus of our security effort.

For a mobile information system, the major threats come from unauthorized users who try to access sensitive information. The possible attacks can be a direct attempt to download sensitive information from the server, or an indirect attempt to steal the information from an authorized user who has access to the information. Even an authorized user may (on purpose or unintentionally) try to pass the sensitive information to an unauthorized user. To prevent information leakage from a direct attack is relatively easier since the server is in charge. The key vulnerability of the system is when the sensitive information is not under server control. Traditionally, the countermeasures only exist on the server and therefore only the first type of attacks can be successfully prevented. It is not clear how to maintain information flow security without server intervention.

As a trusted computing base, we assume that both the servers and PDAs are executing in a fully object-oriented environment such that all possible information access must go through object references and method invocations. Under this assumption, the key innovation of our model is the integration of object-oriented access control lists and security monitor encapsulation techniques to block both unauthorized direct accesses and indirect attacks and thereby keep the sensitive information intact. In other words, we encapsulate security measures within each object such that objects can protect themselves even if not under server control.

3.3 Design Philosophy

Under the assumptions and threat model discussed in the previous sections, we design our model based on the following principles:

- (1) An object should not encapsulate too much information because the entire object, instead of just the information, is sent to PDAs when a legal request is executed. If an object contains too much information, a download operation can be both bandwidth and time consuming. Moreover, unused information may be sent to a PDA.
- (2) A method in an object should manipulate only one type of information. For example, a method should be designed to manipulate the information of hotels and another manipulates the information of lakes. The purpose of designing methods in this manner is twofold. First, if a method manipulates too much information, the method may offer unused information to the PDA holder when the method is executed. Second, manipulating information of different security levels causes trouble when the method is required to execute. For example, if a method manipulates the information of both hotels and lakes, then neither allowing nor disallowing the execution of the method by a civilian role is proper in a strictly military controlled area. The rationale is that a civilian can retrieve the information of hotels but not that of lakes in such area.
- (3) Every object encapsulating sensitive information is associated with an ACL, which is composed of PDA roles that can download the object. The purpose of this ACL is to prevent unauthorized direct access to sensitive objects and save communication bandwidth at the same time. If this type of ACLs were not used, some download operations may be wasted because the methods in the downloaded objects cannot be executed by the PDAs receiving the objects. The access control to sensitive objects is handled by an *object monitor* on the server. All requests to sensitive objects must go through the object monitor to make sure that the PDA roles are eligible for accessing the desired objects.
- (4) Every method manipulating sensitive information is associated with an ACL, which is composed of PDA roles that can access the information. This type of ACLs prevents unauthorized access as well as information leakage among PDAs (*i.e.*, preventing Trojan horses). For example, suppose a PDA transfers a sensitive object to another PDA. Then, the latter PDA cannot execute a method if the role is not within the ACL of the method. The access control as described above is carried out by a *method monitor* downloaded along with the object encapsulating the sensitive information. All invocations to sensitive methods must go through the method monitor to make sure that the requesting PDA roles are eligible for executing the desired methods. Fig. 2 demonstrates the use of object monitor and the downloading of a restricted object with the associated method monitor.

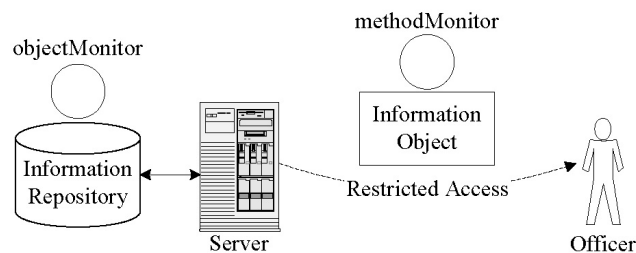


Fig. 2. Associating security monitors with objects.

- (5) So far, we have emphasized that a server sends objects instead of information required by a PDA without explaining the reason. In fact, the purpose is to prevent information leakage among PDAs. Imagine that when an officer requests the information of beaches, the server directly sends the information to the officer's PDA. The officer can then transfer the information to other PDA even if the PDA role is not allowed to receive the information. To prevent this type of information leakage, when a PDA requests for a piece of information and the PDA role is allowed to access the information, the server sends the objects encapsulating the information to the PDA. To obtain the information, the PDA holder must execute the corresponding method. If a sensitive object is transferred to a PDA whose role is not allowed to access the information, the method cannot be executed because the PDA role is not within the ACL of the method.
- (6) When a method is executed, the following actions should be taken to prevent information leakage among PDAs. First, the information offered by a downloaded method can only be shown on the PDA's display. It cannot be stored in the PDA's storage (either main memory or secondary storage). The rationale is that information stored in the PDA's storage cannot be controlled by an ACL. The PDA holder can freely transfer the information in the PDA's storage to other PDAs, which may result in information leakage. Second, before executing a sensitive method, every application in the PDA should be stopped. The rationale is that information in the display buffer may be caught by an application and then saved in the PDA's storage. Since applications in a PDA are unknown to the server, no application is allowed to be executed in parallel with a sensitive method. Third, during the execution of a sensitive method, if another application of the PDA is executed, the method is instantly stopped and the display buffer is instantly flushed. Flushing the buffer prevents the contents of the buffer from being caught. Fourth, if the PDA holder stops the sensitive method, we ensure that the method stops and the display buffer is flushed at the same time. Although we do not allow other applications to execute in parallel with a sensitive method, we do allow operating system (OS) processes to execute in parallel with the methods because we assume that no OS process will leak information.
- (7) If the information requested by a PDA is not sensitive, the server sends the information directly instead of the object encapsulating the information. This consideration is to save bandwidth when transferring nonsensitive information.

We note that our model does not require an object to be removed once a session is terminated. Once an object is allowed to be downloaded to a client device, it is up to the user to keep the object or remove it after the session is over. As long as the object state remains unchanged, the user does not need to re-download the object. In any case, the object is always under security control by the method monitor.

3.4 Security Model

Formally, we propose a two-level access control model consisting of object level ACLs/monitor and method level ACLs/monitors as depicted in Fig. 2. ACLs limit the access to objects and information while the monitors ensure that the requirements of the

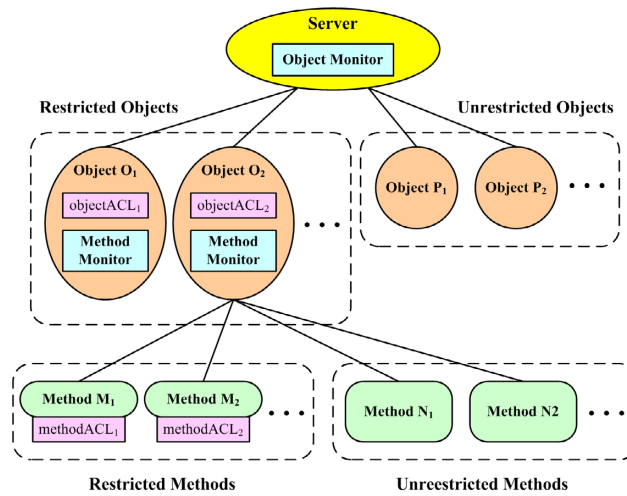


Fig. 2. Two-level access control model for mobile applications.

ACLs are fulfilled. In general, an object-oriented server application is composed of objects and messages, in which messages are passed among objects (message passing corresponds to method invocation). Objects are categorized into two types, namely restricted and unrestricted objects, in which the former encapsulate sensitive information while the latter do not. Only a restricted object is associated with an objectACL. When a request from a PDA is received to download information, the server invokes the object monitor to check whether the information is encapsulated in a restricted object. If the answer is yes, the monitor checks the ACL of the restricted object. If the PDA role is within the ACL, the restricted object is sent to the PDA. Otherwise, the request is denied. On the other hand, if the information is encapsulated in an unrestricted object, the server directly sends the information to the PDA.

According to the description above, we formally define our model as follows.

Definition 1 A server S_i is defined as

$$S_i = (R, U, M, objMonitor)$$

where R is the set of restricted objects,

U is the set of unrestricted objects,

M is the set of messages passed among objects,

$objMonitor$ is the monitor on the server controlling the downloading of restricted objects and unrestricted information.

An object has a name, and encapsulates a set of attributes and methods, in which the attributes contains the information a PDA intended to download. For a restricted object, the following extra information should be included: the ACL of the objects and the indication of whether a method is restricted or not. The latter information is needed because not every method in a restricted object is restricted. A restricted method is associated

with an ACL but an unrestricted one is not. When a restricted object is downloaded to a PDA, every method in the objects becomes an application of the PDA. An unrestricted method can be executed freely. On the other hand, when the PDA holder executes a restricted method, the method monitor checks whether the PDA role is within the ACL of the method. If the PDA role is within the ACL, the monitor checks whether another application is executing. If the answer is yes, the monitor asks the PDA holder to stop the executing applications. After the monitor ensures that no application is executing, the desired method is executed, during which the sensitive information is displayed on the PDA's monitor (*i.e.*, sent to the display buffer of the PDA). During the execution of the method, if the monitor identifies that another application is initiated, the monitor instantly stops the method and flushes the display buffer. Moreover, when the PDA holder stops the execution of the method, the monitor stops the method and flushes the display buffer. Based on the description above, a restricted object is defined as follows.

Definition 2 A restricted object r_i is defined as

$$r_i = (\textit{name}, A, \textit{objACL}, R\textit{Methods}, U\textit{Methods}, \textit{methodMonitor})$$

where *Name* is the object name,

A is the set of attributes of the object representing the object state,

objACL is the ACL attached to the object, which is composed of PDA roles that can access the object,

RMethods is the set of restricted methods,

UMethods is the set of unrestricted methods, and

methodMonitor is the monitor on the server controlling the downloading of restricted objects and unrestricted information.

A method has a name and code that implements the method. If a method is restricted, an ACL is attached to the method. The ACLs of restricted methods are used by the method monitor to control the execution of the methods. A restricted method can therefore be defined as follows.

Definition 3 A restricted method rMd_i is defined as

$$rMd_i = (\textit{name}, \textit{Code}, \textit{methodACL})$$

where *name* is the name of the method,

Code is the code of the method, and

methodACL is the ACL attached to the method, which is composed of PDA roles that can execute the method.

Our model uses two types of monitors namely the object monitors and the method monitors. Every server is associated with an object monitor to check object ACLs and decides whether an object can be downloaded to a PDA. Sensitive information is protected by method ACLs. The purpose of object ACLs and object monitor is to prevent unnecessary download operations, which saves bandwidth. The method ACLs and method

monitors prevent information leakage. Although Definition 2 associates a method monitor to every restricted object, the method monitor of every restricted object is the same. Therefore, when a PDA downloads a restricted object the first time, the method monitor will be downloaded as well. Moreover, when a PDA retrieves a restricted object from another PDA and the former PDA does not possess the method monitor, the method monitor will be sent to the latter PDA together with the restricted object. The method monitor will not be downloaded or retrieved the second time. After a method monitor is downloaded or sent to a PDA, it is executed instantly. The detail operations of the multi-threaded event-driven method monitor is described in Algorithm 1. The first thread checks whether the execution of a restricted method is allowed and ensures that no other applications execute in parallel with the restricted method. The second thread preemptively stops a restricted method when an application is initiated during the method's execution. The interruption is necessary because the newly initiated application may catch the contents of the display buffer. The third thread terminates a restricted method when the PDA holder stops the method.

Algorithm 1 The method monitor

Data: ACLs of restricted methods, display buffer, and PDA role.

Thread 1:

```

while TRUE do
  if a restricted method is required to execute then
    if the PDA role is within the ACL of the restricted method then
      Inquire application status from the PDA's operating system.
      while there are applications executing do
        Inform the PDA holder to stop the applications.
      end
      Execute the restricted method and display the sensitive information.
    end
  end
end

```

Thread 2:

```

while TRUE do
  if an application is initiated when a restricted method is executing then
    Stop the restricted method and flush the memory and display buffer.
  end
end

```

Thread 3:

```

while TRUE do
  if the PDA holder stops a restricted method then
    Stop the restricted method and flush the memory and display buffer.
  end
end

```

3.5 Behavior Analysis

In section 1, we claim that our model solves two important types of information leakage in mobile environments. Moreover, we hope that the performance will not be downgraded too much. The performance will be shown in section 4 using a prototype implementation and simulation results. In this section, we show that the claimed issues are indeed correctly addressed by our model.

Property 1 The model prevents information leakage among servers and PDAs.

Analysis: Our model downloads sensitive objects to PDAs. Within a sensitive object, every sensitive method manipulates a piece of sensitive information. Moreover, every sensitive method is associated with an ACL composed of PDA roles that can execute the sensitive method (*i.e.*, a PDA can execute a sensitive method if the PDA role is within the method's ACL). When a PDA tries to execute a sensitive method, Thread 1 of Algorithm 1 will block the execution if the PDA role is not within the method's ACL. This prevents information leakage among servers and PDAs. \square

Property 2 The model prevents information leakage among PDAs.

Analysis: If the model fails to offer this prevention, a PDA can send sensitive information obtained from a server to a PDA that is not allowed to access the information. Since our model downloads sensitive objects to servers and does not allow sensitive information to be stored in the storage of a PDA, it is impossible for a PDA to send sensitive information directly to another PDA. In fact, our model only allows a PDA to send sensitive objects to other PDAs.

To prove that the model prevent information leakage among PDAs, we make the following assumptions:

- the PDA $PDA1$ with PDA role $PDAR1$ is not allowed to access the sensitive information $sInf$,
- the sensitive information $sInf$ is encapsulated in the object $sObj$ and handled by the method sMd ,
- the ACL attached to sMd is ACL_{sMd} , and
- the PDA $PDA2$ possesses the sensitive object $sObj$ downloaded from a server.

As described above, it is impossible for $PDA2$ to send $sInf$ directly to $PDA1$. However, $PDA2$ can send $sObj$ to $PDA1$. Suppose $PDA2$ does send $sObj$ to $PDA1$ and $PDA1$ tries to obtain $sInf$ from $sObj$ by executing the method sMd . According to assumption above, $PDA1$ is not within the ACL ACL_{sMd} . From Thread 1 in Algorithm 1, $PDA1$ cannot execute sMd and therefore $PDA1$ cannot obtain $sInf$. \square

A legitimate concern with encapsulating ACLs within objects is that whenever there is any change in ACL policies, the new policies must be incorporated into every object (through re-compilation, for example). This situation does not necessarily cause a problem since, on a policy change, only the objectACLs of those objects that are affected by the new policy need to be modified. Since an objectACL is treated the same as a data

member of an object, no re-compilation is required. It does constitute a problem for an object that has been downloaded to a client device before the change since the object-ACL is no longer the same. However, it is more a consistency maintenance problem than a security problem. Any consistency maintenance protocol can be used in here. The simplest way is to invalidate the object.

For applications with complex access control, embedding all ACL policies within each object can be a potential source of excessive overhead. However, some implementation optimization techniques can be applied to alleviate the problem. For example, if all instances of the same object class share the same set of ACL policies which are relatively static, then we can keep the policies along with the class definition rather than duplicating it on each object. Another way is to embed only the method monitor with the object while keeping all objectACLs with the information server.

4. A PROTOTYPE IMPLEMENTATION AND PERFORMANCE EVALUATION

To demonstrate the feasibility of our model, we have built a prototype of the security monitor under the Java environment. The monitor guards against three types of access to the object under protection.

- Direct access to the object. In this case, the monitor checks the ACL to ensure that the access is legal.
- Indirect access through system clipboard. This can happen when a user tries to copy the information in the object and potentially may paste it to another application. In this case, the monitor will clear the clipboard immediately such that no such operation is allowed.
- Indirect access through the manipulation of display buffer. This is similar to the second situation. On detection of such an attempt, the monitor will clear the display buffer immediately such that no information is leaked.

In our prototype, we have constructed the monitor to successfully prevent information leakage of the first and second types which is enough for feasibility demonstration and performance evaluation purposes. The third type of information leakage can be prohibited in a similar way as the second type. The monitor was built based on the object and event model depicted in Fig. 4. The four boxes at the top are the prototype objects implementing the corresponding concepts. The vertical lines are time lines while the horizontal lines are messages (method invocations) sent between objects. The Client Object class represents mobile users. The Object Monitor class simulates the server with an object monitor. The Information Object class implements the restricted objects. RMI is the Remote Method Invocation of the Java language which is used for message communication and remote method invocation between objects. On the construction of an information object, the object monitor on the server is informed (with `monitorInit` and `monitorReady` message pair) and the encapsulated method monitor on the object is initialized and started to monitor all method invocations based on method ACLs. All legal operations (*i.e.* method invocations that satisfy the ACLs) are forwarded to the information object for actual method calls. The result is returned directly to the client without

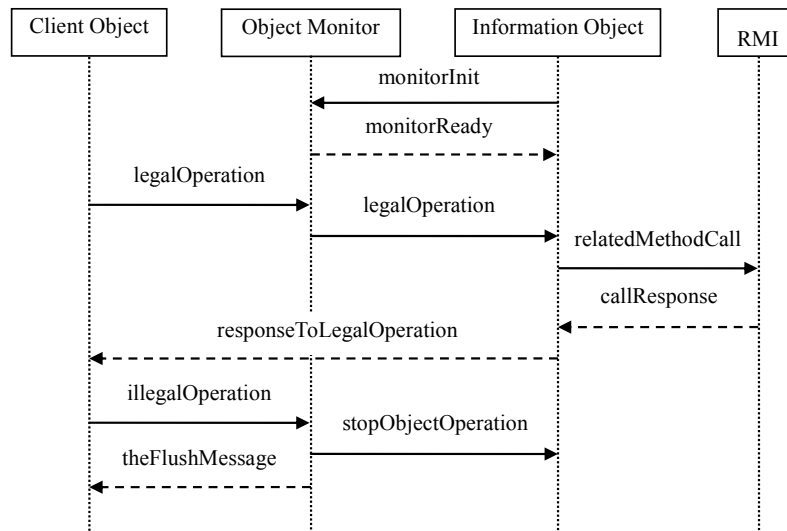


Fig. 3. The object and event model of the prototype.

server intervention. On the occurrence of any illegal operations, the restricted information object is informed and stopped immediately while a flush message is sent to clear up the display buffer. All programs of the prototype are Java J2SE programs running on Windows XP PCs and notebooks under IEEE 802.11 wireless LAN. Therefore, all the object downloading and monitoring operations are actually performed over the network. Thus the performance figures are true measure of the execution time even without implementing the prototype on mobile devices.

Based on the prototype, we conducted two sets of experimentations on simulated requests. The first set was designed to test the feasibility and correctness of our model while the second set was for performance evaluation, especially on the overhead of the security monitor. The object ACLs and the method ACLs are realized following standard ACL implementation. A synthetic scenario for an insecure clipboard operation is illustrated in Fig. 5. In the scenario, a user tries to copy part of the sensitive information to the clipboard. Our monitor successfully detects the illegal operation through clipboard monitoring and flushes the system clipboard and display immediately.

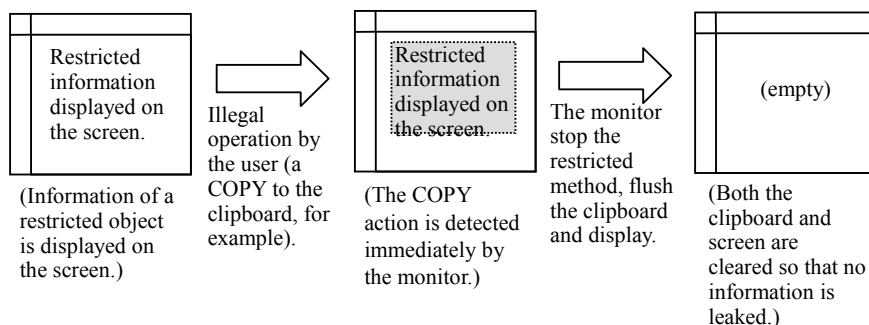


Fig. 5. Monitoring the system clipboard against information leakage.

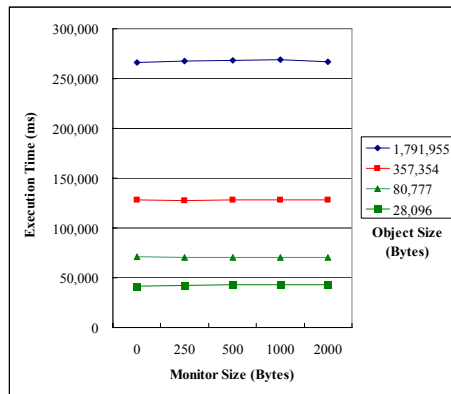


Fig. 6. Average execution time of download operations on objects and encapsulated monitors of various sizes.

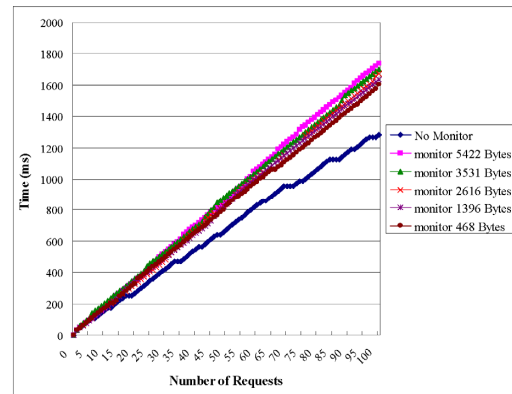


Fig. 7. The accumulated execution time of object downloading operations with encapsulated monitors of various sizes.

For overhead estimation, we measure the average response time of 100 download operations on objects and monitors of various sizes and present the result in Fig. 6. We can see that the overhead of downloading a monitor with an object is almost negligible.

Fig. 7, on the other hand, demonstrates the accumulated execution cost of the download operations. It can be observed quite clearly again that the downloading of the security monitor with an information object incurs only a very small amount of overhead which is relatively independent of the monitor size.

From the experimentations above, we conclude that our model successfully prevents information leakage of the types discussed earlier and achieves the desired access behavior described in section 1. The performance study also shows that the model can be efficiently implemented with negligible overhead.

5. CONCLUSIONS AND FUTURE WORK

We propose an object-oriented model and security monitor encapsulation technique to prevent information leakage in mobile applications. The model solves the issues of preventing information leakage due to: (a) unauthorized direct accesses issued by mobile clients to the servers, and (b) indirect accesses among clients (corresponds to avoiding Trojan horses). The proposed model attached access control lists to sensitive objects and methods. Both are enforced by encapsulating security monitors within servers and sensitive objects such that any illegal access is detected and responded immediately. We have demonstrated the feasibility and performance of our model by using a prototype implementation and simulation.

Our model can be extended in many ways:

- The model can be extended to allow finer grain of access control such that information can be selectively released based on flexible and customizable security policies.
- To reduce the size of an object and to increase the efficiency of access, the encapsulated security monitor can also be differentially instantiated according to the security policies.

- Based on the principles of object-orientation (such as inheritance and polymorphism), we would like to develop type inference and similar techniques for the automatic generation of proper monitor instances for different types of objects.
- Finally, our model has been designed based on the assumption that the underlying object-oriented execution environments on both the servers and PDAs are working properly. It is both very interesting and challenging to extend our model, or even to build new models such that this assumption can be relaxed.

The actual implementation of our model on mobile devices is also on the top of our list of future work. We intend to develop several optimization techniques to cope with the potential overhead of encapsulating ACLs within every object, especially for applications with complex access control or ACL policies that change rapidly.

REFERENCES

1. J. Bacon, K. Moody, and W. Yao, "A model of oasis role-based access control and its support for active security," *ACM Transactions on Information and System Security*, Vol. 5, 2002, pp. 492-540.
2. D. E. Bell and L. J. LaPadula, "Secure computer systems: Unified exposition and multics interpretation," Technical Report, MTR-2997, Mitre Corp., 1976.
3. E. Bertino, S. de C. di Vimercati, E. Ferrari, and P. Samarati, "Exception based information flow control in object-oriented systems," *ACM Transactions on Information System Security*, Vol. 1, 1998, pp. 26-65.
4. D. F. C. Brewer and M. J. Nash, "The Chinese wall access control policy," in *Proceedings of the 5th IEEE Symposium on Security and Privacy*, 1989, pp. 206-214.
5. S. C. Chou, "Embedding role-based access control model in object oriented systems to protect privacy," *Journal of Systems and Software*, Vol. 71, 2004, pp. 143-161.
6. S. C. Chou, A. F. Liu, and C. J. Wu, "Preventing information leakage within workflows that execute among competing organizations," *Journal of Systems and Software*, Vol. 75, 2005, pp. 109-123.
7. D. E. Denning, "A lattice model of secure information flow," *Communications of the ACM*, Vol. 19, 1976, pp. 236-243.
8. D. Djenouri, L. Khelladi, and A. Badache, "A survey of security issues in mobile ad hoc and sensor networks," *IEEE Communications Surveys and Tutorials*, Vol. 7, 2005, pp. 2-28.
9. W. Essmayr, S. Probst, and E. Weippl, "Role-based access controls: Status, dissemination, and prospects for generic security mechanisms," *Electronic Commerce Research*, Vol. 4, 2004, pp. 127-156.
10. D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Transactions on Information and System Security*, Vol. 4, 2001, pp. 224-274.
11. E. Ferrari, P. Samarati, E. Bertino, and S. Jajodia, "Providing flexibility in information flow control for object-oriented systems," in *Proceedings of the 13th IEEE Symposium on Security and Privacy*, 1997, pp. 130-140.
12. W. Ford and M. S. Baum, *Secure Electronic Commerce*, 2nd ed., Prantice-Hall, New York, 2001.

13. G. Saunders and V. V. M. Hitchens, "Role-based access control and the access control matrix," *ACM SIGOPS Operating Systems Review*, Vol. 35, 2001, pp. 6-20.
14. H. M. Gladney, "Access control for large collections," *ACM Transactions on Information Systems*, Vol. 15, 1997, pp. 154-194.
15. M. H. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Communications of the ACM*, Vol. 19, 1976, pp. 461-471.
16. G. Kabra, R. Ramamurthy, and S. Sudarshan, "Redundancy and information leakage in fine-grained access control," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2006, pp. 133-144.
17. M. H. Kang, J. S. Park, and J. N. Froscher, "Access control mechanisms for inter-organizational workflow," in *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies*, 2001, pp. 66-74.
18. Y. K. Kwok and V. Lau, "A novel channel-adaptive uplink access control protocol for nomadic computing," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, 2002, pp. 1150-1165.
19. R. Lanotte, A. Maggiolo-Schettini, and S. Tini, "Information flow in hybrid systems," *ACM Transactions on Embedded Computing Systems*, Vol. 3, 2004, pp. 760-799.
20. A. Maamir and A. Fellah, "Adding flexibility in information flow control for object-oriented systems using versions," *International Journal of Software Engineering and Knowledge Engineering*, Vol. 13, 2003, pp. 313-326.
21. C. J. McCollum, J. R. Messing, and L. Notargiacomo, "Beyond the pale of mac and dac-defining new forms of access control," in *Proceedings of the 6th IEEE Symposium on Security and Privacy*, 2000, pp. 190-200.
22. M. D. McIlroy and J. A. Reeds, "Multilevel security in the UNIX tradition," *Software – Practice and Experience*, Vol. 22, 1992, pp. 673-694.
23. A. C. Myers, "Jflow: Practical mostly-static information flow control," in *Proceedings of the 26th ACM Symposium on Principles of Programming Language*, 1999, pp. 228-241.
24. A. C. Myers and B. Liskov, "Protecting privacy using the decentralized label model," *ACM Transactions on Software Engineering and Methodology*, Vol. 9, 2000, pp. 410-442.
25. M. Nyanchama and S. Osborn, "Modeling mandatory access control in role based security systems," *Database Security IX: Status and Prospects*, 1995, pp. 129-144.
26. M. S. Olivier, R. P. van de Riet, and E. Gudes, "Specifying application-level security in workflow systems," in *Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, 1998, pp. 346-351.
27. S. Osborn, "Mandatory access control and role-based access control revisited," in *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, 1997, pp. 31-40.
28. S. Osborn, R. Sandhu, and Q. Munawer, "Configuring role-based access control to enforce mandatory and discretionary access control policies," *ACM Transactions on Information and System Security*, Vol. 3, 2000, pp. 85-106.
29. J. S. Park, R. Sandhu, and G. J. Ahn, "Role-based access control on the web," *ACM Transactions on Information and System Security*, Vol. 4, 2001, pp. 37-71.
30. T. Romuald and C. Stephane, "Integration of access control in information systems: From role engineering to implementation," *Informatica*, Vol. 30, 2006, pp. 87-95.
31. P. Samarati, E. Bertino, A. Ciampichetti, and S. Jajodia, "Information flow control in

- object-oriented systems,” *IEEE Transactions on Knowledge and Data Engineering* Vol. 9, 1997, pp. 524-538.
32. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” *IEEE Computer*, Vol. 29, 1996, pp. 38-47.
 33. R. S. Sandhu and P. Samarati, “Access control: Principles and practice,” *IEEE Communication Magazine*, Vol. 32, 1994, pp. 40-48.
 34. G. Saunders, M. Hitchens, and V. Varadharajan, “Role-based access control and the access control matrix,” *ACM SIGOPS Operating Systems Review*, Vol. 35, 2001, pp. 6-20.
 35. A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 8th ed., John Wiley and Sons, Inc., New York, 2009.
 36. A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 5th ed., McGraw-Hill, New York, 2005.
 37. R. K. Thomas and R. S. Sandhu, “Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management,” in *Proceedings of IFIP WG11.3 Workshop on Database Security*, 1997, pp. 166-181.
 38. Y. Villate, A. Illarramendi, and E. Pitoura, “Keep your data safe and available while roaming,” *ACM/Kluwer Journal on Mobile Networks and Applications*, Vol. 7, 2002, pp. 315-328.
 39. H. Wang, Y. Zhang, J. Cao, and V. Varadharajan, “Achieving secure and flexible m-services through tickets,” *IEEE Transactions on Systems, Man and Cybernetics, Part A*, Vol. 33, 2003, pp. 697-708.
 40. M. Yasuda, T. Tachikawa, and M. Takizawa, “A purpose-oriented access control model,” in *Proceedings of the 12th International Conference on Information Networking*, 1998, pp. 168-173.



Shiow-Yang Wu (吳秀陽) is an associate professor of the Department of Computer Science and Information Engineering at National Dong Hwa University, Hualien, Taiwan, R.O.C. He received the B.S. and M.S. degrees in Computer Engineering from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., and the Ph.D. degree in Computer Science from the University of Texas at Austin in 1984, 1986, and 1995, respectively. His research interests include data/knowledge bases, mobile and pervasive computing, distributed processing, intelligence information systems, and electronic commerce.



Shih-Chien Chou (周世杰) received a Ph.D. degree from the Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan. He is currently a professor in the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan, R.O.C. His research interests include software engineering, process environment, software reuse, and information flow control.