Optimization of Access Policy among Multiple Objects in Java Card

Longlong Jiang
School of Computer Science and Technology
Guangdong University of Technology
Guangzhou, China
long001898@163.com

Daiping Li
School of Computer Science and Technology
Guangdong University of Technology
Guangzhou, China
l-dp@163.com

Abstract—Applets which are in different contexts in java card use shareable interface object to pass information and provide method services. This kind of access applications method is only suitable for the situation between two applets in different contexts. When pass information and services among three or more applications, it appears the security issues that it is unable to control the data flow. To find the reason of the problem, analyze the whole access process and point to the key of the judgment method. In order to solve this problem, propose a new security access policy among multiple objects, use the feature set of the applet, and calculate the feature set by JCRE logical judgment module. According to the result of calculation, decide whether to allow application to use the shareable interface object. The new policy implements data security access and adapts to a variety of data-access requirements.

Keywords- context; applet isolation; security access control; shareable interface objec; feature set

I. INTRODUCTION

Java smart card is more and more used in all aspects of society. The advantage of the java card include: convenient program, safe, especially multiple applications on one card.

Java card application firewall provides isolation between different applications [1]. Isolation means that the applet does not readily accessible to other applications and services, or communication to obtain information [2]. If the applet need communicate with other applications, system resource or service programs that in a different context, should follow certain rules in order to complete the communication, access to information or transmitting information [3].

Applet can access static fields and methods, static fields and methods belong to the global object [4]. Applets in the same context environment can reciprocally communicate with each other which are not subject to restrictions. Between different contexts access rules are as follows:

(1) Access between JCRE and applet

i. Java Card runtime environment (JCRE) belongs to a particular context, that context is a system context, with special privilege, unrestricted access to the context of the application of space. So access is not restricted from JCRE to applet.

ii. Data access from the applet to JCRE systems require JCRE Entry object. Applet cannot access readily JCRE system resources, when needs necessary data access, JCRE should provide specific JCRE Enter and the applet can only use the JCRE Enter of its own.

(2) The data access between applets in different contexts in applet space requires shareable interface object (SIO) supported by Java smart card technology to carry out visits or calls. Shareable interface defines a set of interface methods, these methods can be called by context [5].

II. SHAREABLE INTERFACE OBJECT

Shareable interface object provides interfaces for communication between applets in different contexts [6], specific access steps are described as follows:

First, applet A that provides the shareable interface object registers the reference of the object to the JCRE.

Second, applet B that will access to the applet A requests the reference of the shareable interface toward A via the JCRE. Applet A returns the reference of the shareable interface object via the JCRE. Applet B receives the referenced and can access the contents in the shareable interface object. Then, it implements the communication between applets in two different contexts.

Third, when applet B calls a method in the shareable interface object, accordingly JCVM does context switch.

The whole process as shown in the Fig.1:

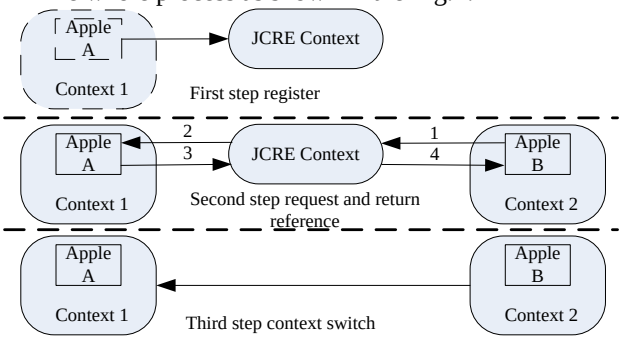

Figure 1. Access steps between different context applet via shatreable interface object.

Applet B requests the shareable interface object reference through the JCRE by calling the API method JCSystem.getAppletShareableInterfaeeobjeet [7]. After receipt the requisition applet A will validate whether the applicant has permission to use the shareable interface.

A. Summary of Shareable Interface Object

Application firewall mechanisms specified in Java smart card Technical specification provide logical application essentially multiple application collections labeled by the object context [8]. Objects access in the same collection is legitimate, and objects access of between different contexts is conditional. The access rules crossing different contexts is the shareable interface object access mechanism.

Analyze the Java Card API that involved in shareable interface object access mechanism, can get the information that the shareable interface object reference security checks would be done in the Applet that provides the shareable interface object.

Shareable interface object mechanism essentially puts the data that will be provided to other application that require these data and outside the firewall and methods into one special object.

Java card application firewall allows such objects to be accessed by applications that are in other contexts. Firewall application does not make any verifications, it is allowed access through the firewall. For the judgment of applicant, applications full that receives applicant make their own decisions. If the judge allows the application to access, send the shareable interface reference to JCRE, and JCRE will return the reference to the applicant, otherwise access will be denied.

III. ANALYSIS OF SHAREABLE INTERFACE OBJECT

A. Problem in the Process among Three Objects

This communication policy between applications is suitable for the two objects. If it is more than two objects that communicate using this policy, there will be big problems.

There are three applet: applet A, applet B and applet C in Fig.2. Applet A provides data and method through shareable interfaces, applet B request to use the data and methods. After receive the application, applet A will return the shareable interface reference to the applet B. Applet B obtain the right to use this object. If at the same time applet B provides data and methods for applet C also using shareable interface, there may be such problem: applet B may put reference of object of applet A into shareable interface object of its own which is requested by applet C, then after getting the object reference of applet B, applet C also gets the data of the applet A. Actually, these data belonged to applet A is only available to applet B, and only applet B can use it, but applet B put the shareable interface reference into its own shareable interface object, and unexpected applet C access to these data.

B. Insufficient of the Access Policy

One of insufficient point is that once the applet implements a shareable interface object and put the data and methods into the shareable interface object, these data and methods within the object will no longer be under the control of its own applet.

This problem is due to when applet A accept the requisition of applet B and provide the shareable interface reference, it only uses the AID of applet B and judge whether applet B's AID is in its AID list or not. This judgment does not adapt data access across multiple applications.

Another insufficient point is that applet needs to maintain a list of AID. If there are frequent changes, there need a constantly update the AID list.

C. Analysis of Other Method

Girand and his copartners proposed a level object sharing mechanism based on partial sequence grid [9]. In order to control the flow of data, you can specify a level for all objects generated by the card application, the data information can only flow from a low level applet object to a high level of object in java card. This approach has three problems:

First, if there is a malicious program having obtained the highest level, all data and method in the related applet will no longer be safe. Malicious programs applet can freely access the data information in the other application that it wants to get [10].

The second is how to specify the level of Java objects in a smart card , especially if the new downloadable application to access the shareable interface object of the original application in the card, the virtual machine how to specify the level of shareable interface object newly created by applet objects;

Third, this sharing object mechanism is not suitable for a variety of sharing access requirements. If an applet program requires both low-level information and high-level information and itself must be set to the middle level, it will not be able to suitable for this mechanism. Another situation, if there is need to change the applet level and will need to re-set level, then all associated applet needs to reconsider its level setting. This situation is quite complicated.

So this approach is not suitable for many application scenarios of java smart card. Access mechanism based on existing shareable interface objects, as well as other access methods, introduce a new shared object access mechanism and give its formal definition.

IV. SOLUTION OF MULTIPLE APPLET SHAREABLE INTERFACE OBJECT

Each applet sets a feature set of its own. When applet B requests shareable interface of applet A, it will transmit its feature set to JCRE. When applet A receives the requisition, it is not need to do any of judgment and transmit its feature set and shareable interface object reference to JCRE. Then JCRE calculate both feature set to decide whether applet B has the permission to access the object of applet A. As shown in Fig.2:

When the third applet C asks for access to SIO of applet B, it sends its own feature set to JCRE, applet B upon receipt of the request at the same time transmits its SIO and feature set to JCRE. Then JCRE is to judge the requisition, but due to the SIO of applet B contain the SIO reference of applet A ,JCRE will once again send a notification to applet A. Applet A receive the notice and sends its own feature set to JCRE.

The result whether applet C can obtain the SIO reference of applet B is decided by the two judgment that the judgment against both feature sets of applet B and applet C and the judgment against the both feature sets applet A and applet C. This is an "and" operation. Only

two judgment success, can applet C obtain the SIO reference of applet B.

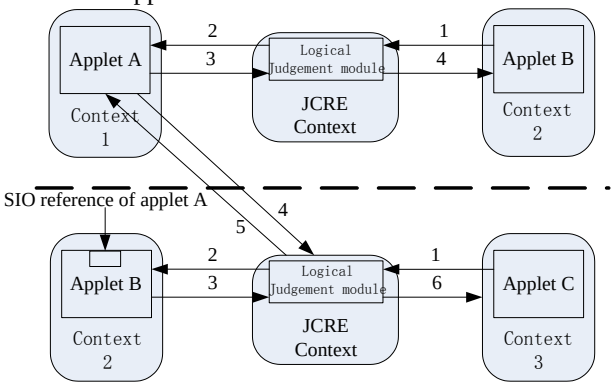

Figure 2. Security access control among multiple applet shareable interface object

V. FORMAL DESCRIPTION

The specification through formal descritption of new proposed shareable interface object access mechanism is as follows:

Define:

SIO represents a set of all shareable interface objects;

APPLET represents a set of all applets [11];

F_{applet} represents a set of all applet feature sets, where applet $\in$ APPLET;

$SIO_A \in SIO$ represents the shareable interface object of applet A, where $A \in APPLET$;

Trust($applet_s$, $applet_a$) represents the trust relationship, that is $applet_s$ which represents the provider of the SIO_s trust the $applet_a$ which represents the applicant for the SIO_s; If and only if the operation results of the feature set is true, the logical is true.

Judge(F_s, F_a, SIO_s) represents logical decision of the JCRE through calculating the both feature sets, F_s represents the feature set of the SIO_s provider and F_a represents the feature set of applicant for the SIO_s.

By above definition, standardization logic strategy can be expressed as follows:

$$Trust(applets , appleta) \leftrightarrow Judge(Fs , Fa , SIOs) \quad (1)$$

Only when the logic judgment condition is true, the trust relationship is true.

For the Fig.2, formula (2) that represents the applicants apply to the provider's SIO is as follows:

$$Trust(A_s , B_a) \leftrightarrow Judge(F_A , F_B , SIO_A) \quad (2)$$

where $A \in APPLET$, $B \in APPLET$, $SIO_A \in SIO$

If the judgment of F_A and F_B is true, applet A will accept the requisition, otherwise refuse the requisition.

This is the situation of two applet that one accesses the other one's shareable interface object.

For the latter situation in the Fig.2, among three applet:

$$Trust(B_s , C_a) \leftrightarrow Judge(F_B , F_C , SIO_B) \quad (3)$$

where $B \in APPLET$, $C \in APPLET$, $SIO_B \in SIO$

Because $SIO_A \subset SIO_B$, SO :

$$\begin{aligned} &Judge(F_B , F_C , SIO_B) \\ &\quad = Judge(F_B , F_C , SIO_b) \wedge Judge(F_A , F_C , SIO_B) \\ &\quad = Judge(F_B , F_C , SIO_b) \wedge Judge(F_A , F_C , SIO_A) \\ &\quad = Trust(B_s , C_a) \wedge Trust(A_s , C_a) \end{aligned} \quad (4)$$

where $A \in APPLET$, $B \in APPLET$, $C \in APPLET$, $SIO_A \in SIO$, $SIO_B \in SIO$, $SIO_b \in SIO$.

SIO_b represents the SIO of applet B except the reference of applet A's SIO. Here, use the SIO_b to distinguish the SIO_B that represents the SIO of applet B contains the reference of applet A's SIO.

After conversion, Standardization formula is as follows:

$$\begin{aligned} &Trust(B_s , C_a) \\ &\quad \leftrightarrow Judge(F_B , F_C , SIO_b) \wedge Judge(F_A , F_C , SIO_A) \end{aligned} \quad (5)$$

This formula represents the judgment condition for the situation that is descripted in the lower part of the Fig.2.

From the new judgment condition, can get the new trust relationship as follows:

$$Trust(B_s , C_a) \leftrightarrow Trust(B_s , C_a) \wedge Trust(A_s , C_a) \quad (6)$$

This formula represents the new trust relationship that is contained in the Fig.2.

A. *Specification of Formal Description*

If the requested SIO contains other SIO reference, there must be an additional judgment that judge the feature set of the applicant and the feature set of the applet that contain the other SIO. So as shown in Fig.2, the security control rule is:

1) determine whether the applet C satisfy access conditions of applet B;

2) determine whether the applet C satisfy access conditions of applet A;

Only two conditions are satisfied, applet C can access the SIO of applet B.

Here Hidden another condition, it is B satisfy the access conditions for applet A. In general, if there is not the reference of applet A shareable interface object in the shareable interface object of applet B, the problem of three applets access requisition does not exist. The access requisition among three applets can be broken down into access requisition between two applets. Only when the requisition of applet C to shareable interface object of applet B involves the third party, it will be the safety problem that applet A can't control the data in its own shareable interface object. So, the hidden condition is:

There is the shareable interface object reference of applet A in the shareable interface object of applet B.

That is, applet B must satisfy the access condition of applet A.

VI. FEATURE SET DESCRIPTION

Feature set is include applet's identity, rights, and some characteristics of need, and it judge whether satisfy the access conditions between different objects. Feature set should have at least one element, elements' status is equal. Feature set contains elements in general are:

1) its own identity: applet AID, use the applet AID can quickly judge whether the applicant applet satisfy the conditions of access control.

2) access regular: permission and role [12]. This is the basic access regular, when each applet package was installed into the card system, the JCRE system defined its permission and role [13].

3) the other feature: include the access condition that defined by developer or issuer. This feature should be installed into the java card system when application was installed on the card.

A. *Advantage of Feature Set*

Using the feature set to describe each apple and to judge whether the requisition can be allowed has great flexibility and convenience. There are three reasons:

First, using feature set to describe the applet can increase the safety of the applet. Illegal applet wants to fake other legal application is bound to become more difficult. Feature set can make different security measures together data such as security access mechanism, identity recognition, permissions and role etc. Even a malicious applet steals the identity information from a legal application such as AID [14]. But if there is no other security features, it will not be able to access other applications shareable interface object.

Second, provide flexible judgment method and decision module can use a single element to make the decision, can also use the combination of elements to make the decision. It depends on the use of smart card specific environment and security access level [15].

Third, simplify the setting of the security access. Different context can inherit the same set of feature set. When develop applet in a certain context group, the feature set of the applet can be directly inherit the group's feature set. Do not need to set a specific security access every time, also do not need to update AID list of related applets every time.

VII. CONCLUSION

The advantage that all applets at the same time provide the feature set used to judge whether the requisition satisfy the security requirements is that it is good adaptability and flexibility to be suitable for access request diversity. In the simplest case of application between two objects, it just need to verify their AID to decide whether to accept the application. If there not set up an AID list, then you can determine through the high level of access control regular, so that it reduces the frequency of setting applet AID list when develop the applet program, and does not require frequent updates AID list. And through JCRE systems level of judgment using the feature set, the new access policy may reject some cases that malicious applications access applet shareable interface object by illegally obtained access right.

Through the actual Java card test environment and the access interaction among three applets, has verified the effectiveness and reliability of the new security access strategy. From the perspective of the overall analysis, the new security access mechanism, not only solves the problem of access to each other among more than two application objects, but also improves the overall security of the smart card. And the design of the new solution does not need to take up additional memory or calculation resource, only need to integrate some existing security mechanism. It improves the availability of this method, and makes the Java card to well adapt to development and utilization of multiple applications.

REFERENCES

[1] Oracle Corporation. "Virtual Machine Specification Java Card Platform. Version 3.0.1, Classic Edition." 2009. http://www.oracle.com/

[2] Oracle Corporation. "Runtime Environment Specification Java Card Platform. Version 3.0.1 Classic Edition." 2012. http://www.oracle.com/

[3] Xu Yixin and Zhang Qishan. "Analysis and Implementation of Security Object Sharing Policy in Java Card." Journal of Computer Applications. Vol 29, No 6, 2009. pp: 1615-1617.

[4] Kyongho Han, Yongsang Song, and Jongmoo Choi. "Multimodal Security Enhancement Scheme for Java Card." 2011 International Conference on Network-Based Information Systems. 2011.IEEE Computer Society , doi: 10.1109/NBiS.2011.105

[5] Sun Yaqin and Wu Yuchuan. "Analysis and Research to Security Testing of Smart Card." 2009 International Conference on Electronic Commerce and Business Intelligence. 2009. IEEE Computer Society. pp. 99-101, doi: 10.1109/ECBI.2009.66.

[6] Liu Hui. "Java card Security Analyses and Research." Shandong University. 2008.

[7] Jiangpei Xu, Liji Wu, Xiangjun Yang, Yuzhong Wang and Xiangmin Zhang. "A Security Vulnerability of Java Card on Array Access in Financial System." Beijing Smart Card Research Institute of Zhongchao Credit CARD co.,Ltd. pp: 707-710.

[8] Marco Avvenuti, Cinzia Bernardeschi, Nicoletta De Francesco, and Paolo Masci. "JCSI: A Tool for Checking Secure Infromation Flow in Java Card Applications." The journal of Systems and Software, Vol. 85, 2012, pp. 2479-2493.

[9] Michael Lackner, Reinhard Berlach, Michael Hraschan, Reinhoid Weiss and Christian Steger. "A Defensive Java Card Virtual Machine to Thwart Fault Attacks by Microarchitectural Support." 2013. International Conference on Risks and Security of Internet and Systems.

[10] Jason Howarth, Irfan Altas, and Barney Dalgarno. "Information Flow Control Using the Java Virtual Machine Tool Interface." 2010. International Conference on Availability, Reliability and Security. IEEE Computer Society. pp.689-695, doi:10.1109/ARES.2010.75.

[11] Won-Ho Choi, Se-Won Oh, Gwang Jung, and min-Soo Jung. "A Novel Scheme for Efficient Installation of Applets for Advanced Java Card System." 2009 World Congress on Computer Science and Information Engineering. IEEE Computer Society. pp. 60-66, doi: 10.1109/CSIE.2009.867.

[12] Ahmadou Al Khary Sere, Julien Iguchi Cartigny, and Jean Louis Lanet. "Checking the Paths to Identify Mutant Application on Embedded Systems." Lecture Notes in Computer Science 6485, FGIT 2010. pp. 459-468.

[13] Julien Lguchi-Cartigny and Jean Louis Lanet. "Developing a Trojan Applets in a Smart Card." Journal of Computer Virologu and Hacking Techniques, Vol, 6. 2010. pp. 343-351. doi: 10.1007/s11416-009-0135-3.

[14] Ewout Keuleers and Jean Marc Dinant. "Data Protection and Multi-application Smart Cards – the Use of Intelligent Servers to Ensure Interoperability and Data Flow Requirements." Computer Law and Security Report, Data Protection Implications of Smart Card Schemes. Vol, 21, 2005, pp. 146-153. doi : 10.1013/j.clsr.2005.02.001.

[15] WeiNgan Chin, Tuan Huang Pham, and Anh Hoang Truong. "A Fast Algorithm to Compute Heap Memory Bounds of Java Card Applet." 2008 Sisth IEEE International Conference on Software Engineering and Formal Methods. 2008. IEEE Computer Society. doi: 10.1109/SEFM.2008.30.