Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 21 (2013) 18 – 24

Procedia Computer Science

The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2013)

Personalized Security Approaches in E-Banking Employing Flask Architecture over Cloud Environment

Nayer A. Hamidi[a,*], Mahdi Rahimi G. K.[b], Alireza Nafarieh[c], Ali Hamidi[c], Bill Robertson[c]

[a]*Department of Computer Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*
[b]*Department of ComputerEngineering& Information Technology, Amirkabir University of Technology, Tehran, Iran*
[c]*Faculty of Engineering, Dalhousie University, Halifax, Canada*

Abstract

Personalized security in E-banking is an important issue for many individuals and companies that are looking for achieving the proper level of security. The cloud environment is a suitable infrastructure to implement personalized security mechanisms for many big companies such as banks. Employing mandatory access controls boosts the security of E-banking to a high level. Flask architecture is the security architecture which enforces mandatory access control that provides a clean separation of security policies and enforcements. In this paper, it a model for implementing personalized security in E-banking over a cloud environment is introduced. Using role-based access control models, the security system can be configured to provide expected level of security in e-payment systems employing user-defined policies for individuals, companies and governmental organizations. The paper focuses on defining such policies and how they improve the level of security.

*Keywords:*Personalized security; E-Banking; E-Payment; Flask architecture; Cloud computing; Security

1. Introduction

Since financial systems are one of the main paramount goals for hackers, security has been an eminent issue in these systems. The increase in the number of financial institutes that use computational internet-based services requires them to apply more complicated and advanced security schemes to cope with threats and frauds vulnerabilities. Meanwhile, any failure to meet the security requirements in such

* Corresponding author. Tel.: +98-21-29982604.
E-mail address: n.hamidi@qiau.ac.ir.

doi:10.1016/j.procs.2013.09.005

systems may lead to a critical disaster for these institutes. The effective security requirements are an abstract idea that would not be easily achievable. Many solutions have been proposed to reduce the risk of fraud and to provide the better immunity over online transactions such as e-banking operations. However, none of those has had an effective means of addressing issues thoroughly. In this paper, some of the well-known existing approaches together with their advantages and shortcomings are discussed, and a novel personalized security system is proposed to provide a safe environment for E-banking transactions over cloud environments. The safe environment is provided by enforcing some policy related to individuals, companies, and governmental institutions.

The Flask architecture introduced by National Institute of Standards and Technology (NIST) has reinforced mandatory access and concurrency control to meet the high-speed e-banking transactions processing requirements. Flask introduces essential requirements for personal safety. The paper is organized as follows: Section 2 presents the related work on implementing Flask architecture and cloud applications in E-banking. The proposed personalized security mechanism is discussed in Section 3. In this Section, a model and all related policies to implement the personalized security in E-Banking over cloud computing is presented. Section 4 investigates the performance of the proposed mechanism, and Section 5 draws the conclusion.

2. Related Work

Design and implementation of Flask architecture in distributed systems such as cloud has been discussed in [1] and [2]. The geographical dispersal of information, the improved reliability of multiple computer systems, and the possibility of concurrent execution of applications are important reasons of Flask implementation. The implemented methods in [1] that are usually used to improve the security performance makes it difficult to access or even compare the models and often requires expensive reconstruction costs. Concurrent execution of applications is a very important factor in E-banking systems, since the goal is to reduce transaction processing and report generating times. The solution proposed in this paper adopts the distributed security infrastructure presented in [2] with a selective mechanism for remote authentication based on process types to limit the overhead of some specific processes.

There are some special features that Flask benefits from, such as building and comparing of distributed and concurrent models, having a generic layered architecture, and having sufficient flexibility to support multiple models. Flask architecture runs different concurrent control schemes over persistent data [1]. Moreover, it has the flexibility to support different models of concurrency, such as atomic [3], and nested transactions [4] over the same data. In addition, it consists of a recovery mechanism that can be set up to suit persistent applications [3]. Flask architecture includes primary elements for object managers [5] while allowing users to customize the level of security by enabling or disabling modules during transaction runtimes [6]. In this paper, not only the implementation of Flask architecture in a cloud environment is discussed, but also, to decrease credit card frauds, the implementation of the personalized security in E-banking is introduced.

Banking systems play an eminent role in developing of global knowledge-based economies these days. Numerous advantages of applying information technology on banking solutions are to enable financial institutes to easily adapt to the market constraints and remain on top in providing favourite services for clients. Cloud environment is one of the newest informational technologies to provide such infrastructures and capabilities [7]. Although employing cloud has enormous benefits [8], security issues such as security in general and security in database should be taken into account. Cloud has four models of deployments. One of the best one that fits better in e-banking for sensitive, non-shared data is private cloud [9]. Cloud has the capacity to change completely the financial service prospects. Enterprise banking systems using related technologies give accessibility anyone, anywhere to benefit from modern core banking solutions

without high cost and other difficulties of the newly established knowledge [10]. Service level agreements (SLAs) define security and customer's information constraints compliances over cloud. In addition, cloud-based solutions fully comply with the minimum compliance standards and are acceptable to the risk and security officers [11].

3. Personalized Security Mechanism

3.1 Proposed Solution

This paper concentrates on the user security. That is, security matters considered in a private financial institute is totally different than an individual with a small amount of bank deposit. These two must be evaluated from different aspects and separate ways. As the rules are to be applied to different types of accounts, security policies can also be different. For instance, a large private company with large revenue should provide maximum security for the accounts by using group signatures on cheques. Security can be defined for an account or a card. To enhance the security of cards, the following means of card security should be considered: i) a card should work with certain devices under any circumstances. For instance, cards only should be active through internet banking and transactions should be via the IP range of the company; ii) some accounts require having a phone number to call and a text message confirmation. If the confirmation is not received within a certain period of time, the transaction will be cancelled; iii) transactions must be acknowledged by a fax or an e-mail. Otherwise, after a certain period of time, the reversal transaction is generated by the system and the first transaction will be cancelled; iv) transaction tracking codes will be sent through a channel. One or more phone numbers can be used to confirm the transactions. Based on these considerations, in the next section a security solution for card transactions is proposed in which Flask architecture is adopted as the existing architecture to control user access and enhance the security over cloud environment.

3.2 Personalized security Policy Definitions

This section defines registering, employing, modifying and deleting personalized security policies. For this purpose, a server (or servers in order to load balance and fault tolerance) is allocated to apply the conditions on personalized security policies. The server is also responsible for sending and receiving SMS confirmations and sending reversal transactions to credit cards. Individuals and company agents that use this method are authenticated after filling up related forms at bank branches. An application can be used to translate hard-copy requests to the policies, which is issued by personal security servers. For instance, if confirmation is done via SMS or Email by a single person, the policy is created as follows:

Allow {Cell_number | Email} {PAN | Account_no} expired_time(min) [CardPresent | CardNotPresent]
Allow +989xxxxxxxx 603799xxxxxxxxx 5

Where *Allow* refers to the listed phone numbers or emails valid to send confirmation messages. The policy identifies that for a certain card or account number, an SMS confirmation from phone or an Email should come until a timer times out. *CardPresent | CardNotPresent* is optional and refers to transactions in which cards are physically present, such as using a point of sales (POS) to do a transaction compared to transactions that does not require physical cards such as Internet e-payments. All transactions will be considered if this option is not used. SMS can contain one of the following options: *y* as yes and *n* as no to switch to Email solution in which user can click the sent link to confirm the transaction.

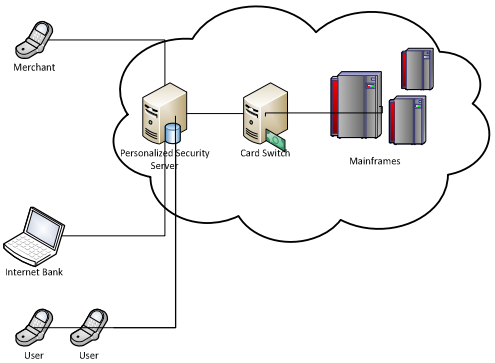


Figure 1: Overview of proposed schema

The following policy is a group signature policy by which a group of people acknowledge together, and if one of them refuses to confirm, the transaction must be failed. This policy is so useful for companies to be operated by several accountants in an accounting department. In this case, confirmation of director of accounting department and one of the accountants is enough to approve the transaction. In the following policies, SMS confirmations are sent to the entire list (i.e. {or {Cell_number | Email}}*) and are approved if one is confirmed.

Allow {Cell_number | Email} {{or | and} {Cell_number | Email}} {PAN | Account_no} expired_time(min) [CardPresent | CardNotPresent]*
Cell number of director of accounting department AND
(Cell numbers of accountant OR Cell numbers of accountant OR …)
Allow {Cell_number | Email} and {{Cell_number | Email} {or {Cell_number | Email}} {PAN | Account_no} expired_time(min) [CardPresent | CardNotPresent]*

Another type of policy defined below identifies that if the source of transactions made by certain primary account numbers (PANs) is in the range of local IP addresses, transactions are valid. This policy is well defined for internet banking and online shopping applications, and is a suitable solution when a static IP address is used.

Allow IP_Range {PAN | Account_no}

The following policy applies restrictions on devices to let certain cards make transactions. For instance, a card may only make transactions via point of sales, information kiosks (InfoKiosks), and automatic teller machines (ATMs). In General form we can restrict the use of card only in devices that need to card presents and vice versa. In order to restrict one of the devices, *Disallow* command may be used.

Allow {CardPresent | CardNotPresent} {PAN | Account_no}
Allow{POS [and ATM [and InfoKiosk [and internet [and Telephone [and mobile]]]]]} {PAN | Account_no}
Disallow {POS | ATM | InfoKiosk| internet | Telephone | mobile}{PAN | Account_no}

The following policy is the most commonly used policy by those who are not willing to use teller machines or mobile banking transactions. These policies are installed on authorized machines and check any kind of transactions.

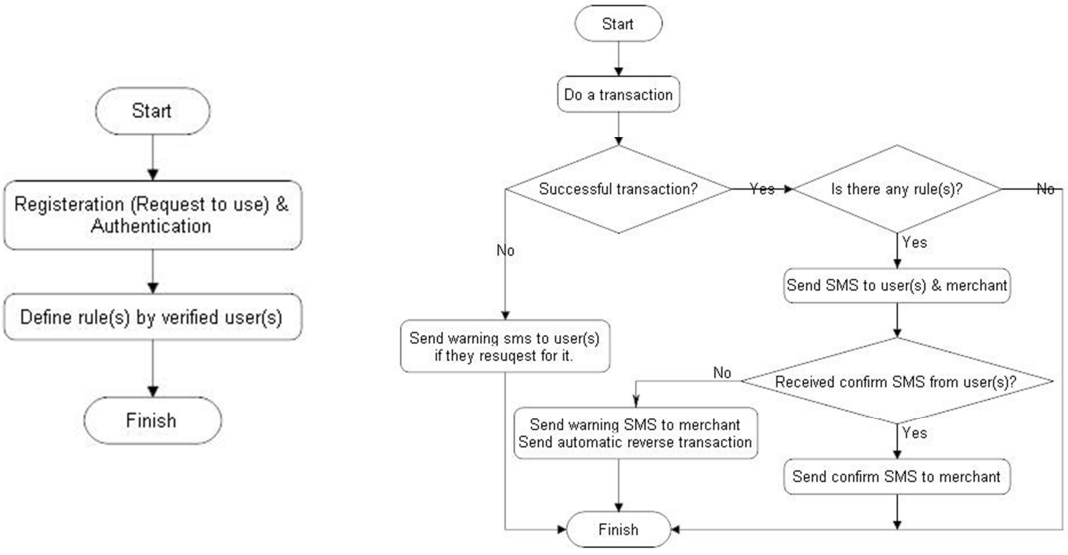
```
TransactionCheckingMethod ()
{
If (Transaction.Type = 200 and Tranaction.ResponceCode = 0) // a successful purchase
        {
                If (Defined any personalized security rule) Then
```

```
                {
                        SendSMS(Users, "If you do the TransactionNoxxxxxx., Please confirm it" );
                        SendSMS(Merchant, "TransactionNoxxxxxx might be cancel, please consider this
                issue" );
                        If (Receive SMS from users and satisfy the rule's conditions)
                        Then SendSMS(Merchant , "Successful purchase for TransactionNoxxxxxx");
                        Else SendSMS(Merchant, "The TransactionNoxxxxxx failed.");
                }
                Else return;
        }
Else
        {
                If (Defined warning rule)
                Then SendSMS(Users, "TransactionNoxxxxxx unsuccessful")
        }
}
```

The policy is implemented if a purchase process is done successfully or any kind of notifications such as "insufficient fund" is not received. If the purchase is unsuccessful and the client still requests the transaction, an SMS is sent regarding the transaction. The policy is defined as fraud detection.

Warning {Cell_number | Email} {{or | and} {Cell_number | Email}} {PAN | Account_no}*

In a successful transaction, if the personalized security policy is applied to a PAN, SMSs are sent to seller(s) and client(s). After receiving the SMS, the seller finds out that the transaction might not be confirmed, and it might be cancelled. In this case, the seller can refuse delivering the goods before receiving the transaction confirmation SMS. If the policy is successful, a transaction confirmation SMS is sent to the seller and the entire process is ended up successfully. If a transaction is cancelled, the system generates a reverse transaction notification and sends it to the switch banking. That is, the money is sent back to the client's account. The way the policy changes are applied is the same as the policy registration process. After the client and the representative of a company are authenticated by the bank, they can define, change, or delete a new policy. All policies are removed from the system after they are cancelled.



(a) (b)

Figure 2: Flowcharts for (a) registration in personalized security system (b) placing transactions

4. Performance Analysis

The proposed solution to improve security in banking transactions has the following considerations: i) although the *expired_time* improves the operation of the proposed mechanism, the merchant must wait to receive the SMS confirmation before delivering the goods to customers; ii) the method proposed in this paper decreases fraud by transaction confirmation; iii) it may increase the cost of a transaction, since for a transaction, the personalized security server sends at least three SMSs and in group policies, the number of SMSs may increase; iv) the cost may decrease by using Email rather than SMS; v) another solution to decrease the cost is to define a threshold for transaction limits. For example, if the amount is greater than 100$, the proposed mechanism sends SMS or Email to confirm the transaction.

Fraudulent transactions in e-commerce have a huge growth from 2.7 billion dollars in 2010 to approximately 3.5 billion dollars in 2012 [12]. Companies reported an average loss of 0.9% of total online revenue as fraud in 2012 [13]. According to the report, fraud rate has increased from 0.6% in 2011 to 0.8% in 2012 and average ticket value for a fraudulent order was $200 higher than a valid order ($149) [13].

If transactions are confirmed by SMS or email, the fraud rate can be reduced by the number of users that register and use the system. For simulation purposes, it is assumed that the statistical population to be 10^5, the number of purchase transactions per year for everyone 10^2, one fraudulent transaction per 10^4 transactions, the number of people using the proposed method 5000 and more, and the policy effectiveness ranging from 50% to 80%. The policy effectiveness is defined as the percentage of frauds that are prevented over total transactions. As Figure 3 (a) and (b) show, the number of frauds decreases when the number of policies that the proposed system employs increases.

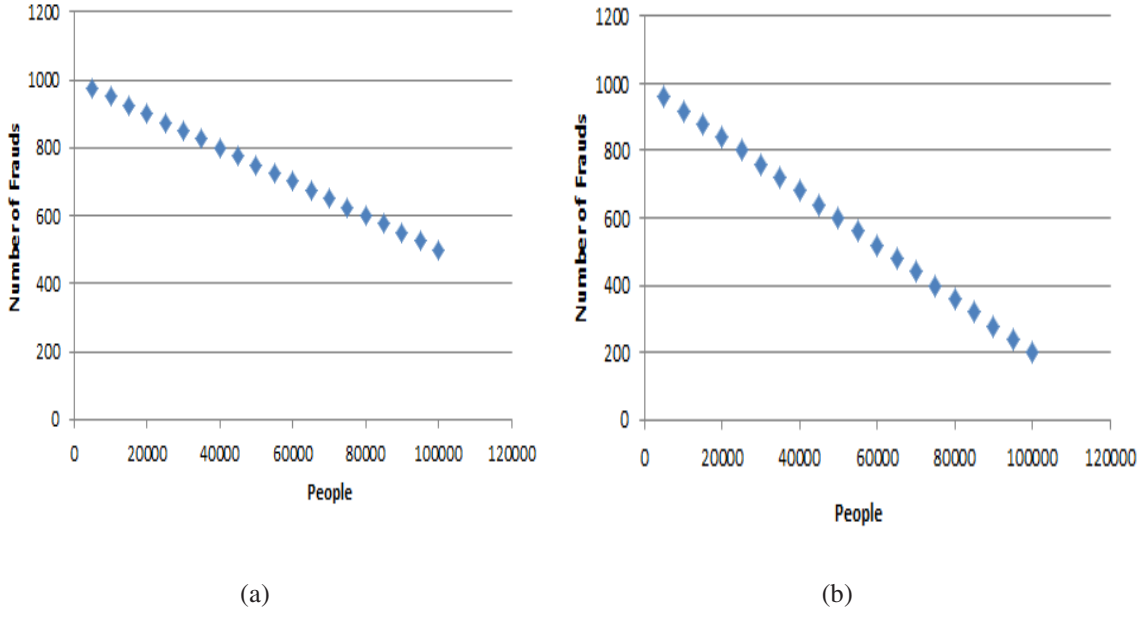


Figure 3: Personalized security system with : (a) 50% effectiveness; (b) 80% effectiveness

5. Conclusion

The paper has introduced a personalized security method in which E-banking transactions are secured against fraud. This paper has presented a method which employs Flask architecture in cloud environments to more effectively avoid fraudulent transactions. To achieve this goal, the policies and algorithms have been proposed to meet different organizations requirements for fraud detection. The performance analysis has shown that the risk of fraud has reduced.

References

[1] Kirby, G.N., Connor, R.C., Cutts, Q.I., Morrison, R., Munro, D.S., Scheuerl, S., Flask: an architecture supporting concurrent distributed persistent applications, Technical report no. CS/97/4, University of St Andrews, 1997.

[2] Leangsuksun, C., Tikotekar, A., Scott, S.L., Pourzandi, M., Haddad, I., Towards cluster survivability.

[3] Scheverl, S.J.G., Connor, R.C.H., Morrison, R., Munro, D.S., The data safe failure recovery mechanism in the Flask architecture. Australian Computer Science Communications 18, 1996, pp. 573–581.

[4] Moss, J.E.B., Eliot, B., Nested transactions: An approach to reliable distributed computing. 1985, Citeseer.

[5] Lepreau, J., Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D., The Flask Security Architecture: System Support for Diverse Security Policies, Secure Computing Corp Saint Paul MN, 2006

[6] Wu, Y., Shi, W., Liang, H., Shang, Q., Yuan, C., Bin, L., Security on-demand architecture with multiple modules support, in: Information Security Practice and Experience. Springer, 2005, pp. 121–131.

[7] Zhou, M., Zhang, R., Xie, W., Qian, W., Zhou, A., Security and privacy in cloud computing: A survey, in: Semantics Knowledge and Grid (SKG), Sixth International Conference On., 2010, pp. 105–112.

[8] Ghanbari M., Cloud and DBMS, International Conference on Information Theory and Application, Singapore, 2011.

[9] Kim, A., McDermott, J., Kang, M., Security and architectural issues for national security cloud computing, in: Distributed Computing Systems Workshops (ICDCSW), IEEE 30th International Conference On. pp. 21–25, 2010.

[10] Banking in the Cloud, www.temenos.com, (accessed June 5, 2013).

[11] Appandairaj, A., Murugappan, S., Service oriented architecture design for web based home banking systems with cloud based service, International Journal of Emerging Technology and Advanced Engineering, vol. 3, pp. 138-143, 2013.

[12] "2013 Online Fraud Report Online Payment Fraud Trends, Merchant Practices, and Benchmarks." CyberSource. http://cybersource.com (accessed June 5, 2013). Page 4.

[13] King, D., Chip-and-PIN: Success and Challenges in Reducing Fraud, in: Retail Payments Risk Forum, January 2012.