A Self-Protective Key Management Framework

Anne V.D.M. Kayem, Patrick Martin, Selim G. Akl and Wendy Powley
School of Computing, Queen's University
Kingston, Ontario, CANADA, K7L 3N6
{*kayem*, *martin*, *akl*, *wendy*}@cs.queensu.ca

Position Paper

Abstract

Self-protection is a key property of autonomic computing systems that researchers have only begun to study. The problem of designing adaptive key management schemes for access control is addressed. We explain why standard key management schemes are unable to efficiently adjust to dynamic scenarios and we present a possible solution to the problem by drawing on the paradigm of autonomic computing. The approach we propose uses a stochastic model supported by data replication to preemptively adjust the resources (in this case, cryptographic keys and encrypted data) to handle varying scenarios in a seamless manner.

1. Introduction

The paradigm of autonomic computing has inspired the creation of numerous computing models aimed at coping adaptively with varying complex scenarios [6]. Yet, these methods have not gained as much popularity in the domain of access control for reasons that include skepticism and reluctance [4]. Skepticism, because security mistakes (failures) can have far reaching consequences. We have only to recall security scandals like the one that occurred in January 2007, when hackers broke into Winners'[1] computers and stole customer credit card information [9]. The public outrage was obvious, how did hackers get into the system? Interestingly enough, failure in other domains is not treated in the same way probably because users are given some assurance or alternative solution. For instance, power failures/surges do not raise the same dust and yet can be equally, perhaps even more, costly. However, since electrical systems have additional backup generators and power protectors to name but a few, the fear of power failure does not generate as much paranoia. Furthermore, the reluctance is born out of the outrage resulting from security failures. Business owners want to trust the software they purchase to yield the results it promises. Costly law suits filed by discontented users are definitely not a part of the bargain. Thus, autonomic security schemes are questioned as to their reliability and dependability in handling complex scenarios.

A number of questions need to be addressed to motivate users to accept the idea of autonomic control in security schemes [6]. First, how can this be done without destroying the very shaky trust that business owners and users have in the power of security schemes? Second, is there a way of establishing when too much control has been handed over to the system? Third, how can we guarantee that the source of a breach will be easier to trace than in standard security schemes? The answers to these questions are not straight forward. However, the growing complexity of computing system management undeniably tips the balance in favor of autonomic computing. Breaches are currently difficult to trace and prevent, and the problem will only become worse with time [4].

We focus on the problem of cryptographic key management (KM) for shared data access control. In this context, the privileges associated with a role can change dynamically, making it difficult for a security administrator to envisage all the possible update scenarios and update the keys associated with a role. The crux of the problem here is that shared key usage requires that key updates be performed in real-time to meet service level agreements. Handling this manually creates delays that impact negatively on performance, making the benefits of autonomicity appealing. We advocate the creation of KM schemes that play the role of self-protectors for a system. Here, self-protection implies that the KM scheme possesses the ability not only to assign users cryptographic keys that prevent security breaches, but also to study user behavior patterns and to adjust the security characteristics of the scheme to

[1] Departmental store in the US and Canada specialized in clothing, shoes and accessories.

cope with the perceived changes. Consequently, the scheme transcends its current status of quasi-static security into the realm of dynamic security making for adaptable and better security.

Our approach does not change the inherent characteristics of standard *KM* schemes, instead it builds on the foundation they lay, using a stochastic model and data replication, to adjust *KM* to handle a perceived scenario. We propose to model the arrival of user join/leave requests and to use the model to predict subsequent arrival rates. Since an existing key is linked to a data object (file) that is encrypted with it, key replacement implies that the associated data needs to be re-encrypted with the new key. Predicted arrival rates are used to solve the problem by creating an optimal number of background replicas and keys that can take over when the change needs to be effected. The background keys are time-bounded to prevent data inconsistency.

This paper presents a self-protective key management (*SPKM*) framework centered on shared data access control. We begin in Section 2 with an example to support the argument in favor of *SPKM* and explain why autonomic computing has not been successful in the domain of security. In Section 3, we discuss our proposed *SPKM* framework and analyze the pros and cons. Concluding remarks are offered in Section 4.

2. Shared Data Access - Background

We focus on using cryptographic keys to control access to shared data. Applications based on shared data are popular because of the flexibility they afford both users and system administrators. Several applications attest to this popularity including web-based distributed collaborative systems (like Facebook [5]).

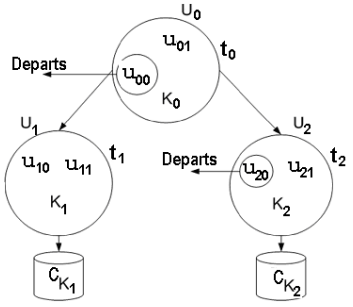

Figure 1. Example of a Dependent Key Graph

Security management is facilitated by classifying data into a number of classes C_{K_i}, such that $1 \leq i \leq n$ where n is the maximum number of user groups in the hierarchy and K_i is the cryptographic key used to encrypt the data. Possession of a correct key grants a user access to the data.

Cryptographic keys for the various user groups requiring access to part of the shared data are defined by classifying users into a number of disjoint security groups U_i, represented by a partially ordered set $(S, \preceq)$, where $S = \{U_0, U_1, ..., U_{n-1}\}$ [1]. In the partially ordered set $U_i \preceq U_j$ implies that users in group U_j can have access to information destined for users in U_i, but not the reverse.

The downside of this traditional method of *KM* is that whenever a group's membership changes, the group key needs to be replaced and the associated data re-encrypted with the new key. Furthermore, when the keys are modeled with a dependency key graph (i.e. each group is assigned a single key from which sibling keys can be derived) all the keys in the affected sub-tree need to be replaced. For example, in Figure 1, when a user u_{00} departs from U_0 both K_0 and the correlated keys K_1, and K_2 need to be changed to prevent the departed u_{00} from continuing to access C_{K_1}, and C_{K_2}. Likewise, when u_{20} departs from U_2 the keys K_0, and K_1 need to be changed in addition to K_2 so that K_0 can derive the new K_2. Additionally the change must guarantee that the new K_2 does not overlap with K_1 and unknowingly grant U_1 access to C_{K_2} or vice versa. This approach to key assignment is not scalable for environments with frequent group membership changes where meeting the goals of service level agreements is an additional constraint [10]. To address these limitations, we advocate using a method suggested by the paradigm of autonomic computing [2, 3].

Autonomicity endows the *KM* scheme with the capability of self-management allowing it to overcome its difficulties in coping with dynamic scenarios [6]. The *KM* scheme can predict future situations based on current observations of user behavior. In essence, the job of the security administrator (SA) is made easier since the SA no longer needs to take care of every key update case that arises but rather, the SA presets specific parameters and allows the scheme to run on its own. Cases directly requiring the SA would henceforth be limited to situations that require the consent/advice of the SA to proceed. In response to the question as to how to design an autonomic *KM* scheme to achieve this objective, we present a *SPKM* framework that automatically protects a system based on observed user behavior patterns.

3. *SPKM* Framework

We argue that *SPKM* for shared data access control can provide performance gains and a reduced management cost. Adaptive *KM* is modeled with a feedback control loop from autonomic management systems [6]. As shown in Figure 2, the model is composed of seven principal components: sensor, monitor, analyzer, plan-

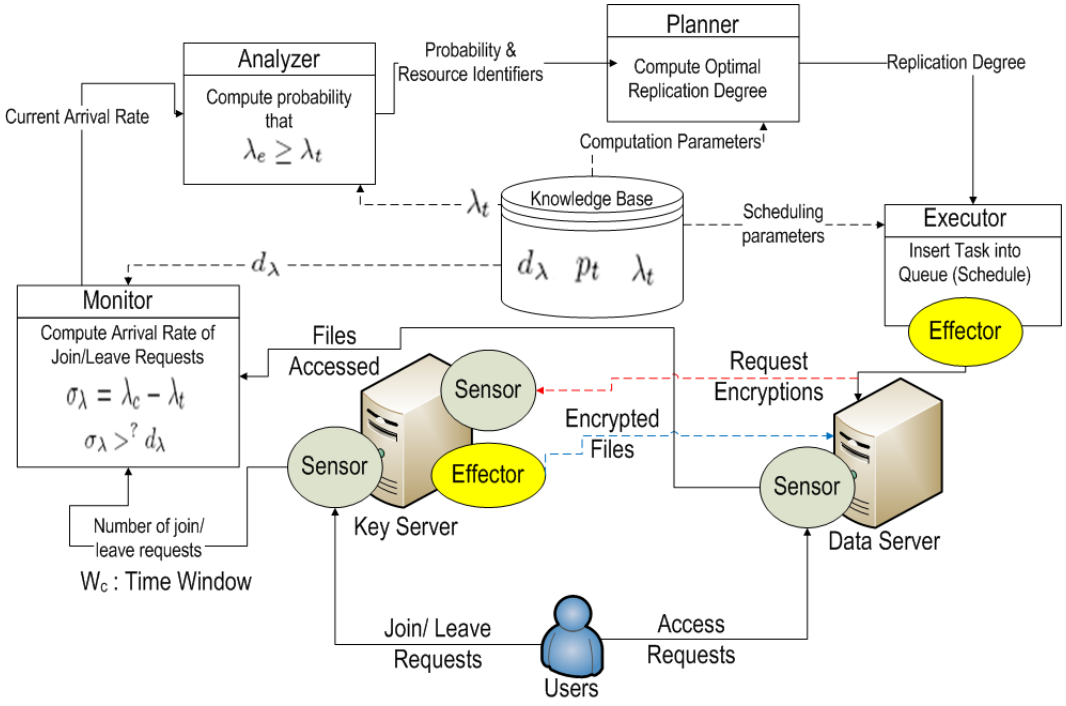

Figure 2. Self-Protective Key Management (*SPKM*) Framework.

ner, executor, effector, and knowledge base. The components react and adjust system parameters according to observations in behavior patterns exhibited by the managed component, namely the data and key servers. For simplicity, the scheme is explained as though we are dealing with a single node in the hierarchy. However, in practice, all the nodes are involved and parameters are set on a node by node basis.

3.1 Sensor and Monitor

The sensor captures requests for key updates emitted by users to the key server during a preset time window W_c and transmits the information to the monitor. The monitoring function computes the mean arrival rate λ_c, of join/leave requests during the current window W_c and compares this value to a preset threshold value of arrival rates, λ_t. If the deviation $\sigma_\lambda = \lambda_c - \lambda_t$ is greater than a preset threshold value d_λ, the event is tagged and transmitted to the analyzer.

3.2 Analyzer

Based on the assumption that the arrival rate follows a Poisson process, the analyzer computes an expected time window W_e, and an expected arrival rate λ_e. Here, "expected" refers to a future or predicted rate. The probability p_0 that λ_e will surpass λ_t is computed and compared to the probability threshold value p_t. If $p_0 > p_t$, the analyzer tags the event and transmits p_0 as well as the resource identifiers (keys and associated file names) to the Planner.

3.3 Planner

The overhead generated by check pointing can be alleviated by equipping the Planner with a method of determining an acceptable degree of replication to maintain in the system [7,8]. Our proposed approach uses a Markov process to design the optimal replication degree algorithm. With this model, the Planner evaluates the expected overhead for a given number of replicas with respect to the degree of availability α (probability that the system can satisfy a given request before a set deadline) and uses the information to determine an optimum number of replicas N to generate in response to λ_e. The optimum values for N, W_e, and α as well as the identifiers of the keys and files concerned are transmitted to the executor.

3.4 Executor and Effector

The executor proceeds to create the required replicas and instruct the key server to create time-bounded keys for the replicas and encrypt the copies. A time-bounded key K_i^t is defined by a combination of the key K_i associated with a security class U_i and the duration of its validity t, such that $T_s \leq t \leq T_f$. Here T_s and T_f refer to the start and finish times of the bound.

For instance in Figure 3, the replicas under U_i are encrypted with keys $K_i^{t_0}, ..., K_i^{t_{n-1}}$. Updates on backup copies are only accepted if they are encrypted with the key that is associated with the primary copy. When the key $K_i^{t_0}$, associated with the primary copy, expires (i.e. the time-bound $T_s \leq t \leq T_f$ is no longer valid) or a join/leave request requires replacing the key $K_i^{t_0}$, the replica $R_{K_i^{t_0}}$ is destroyed. The key $K_i^{t_1}$ is then distributed to the users in group U_i and $R_{K_i^{t_1}}$ becomes the new primary copy.

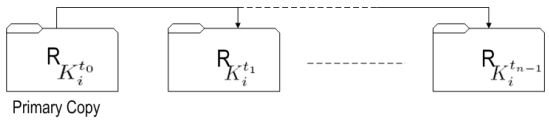

Figure 3. Replication based time-bound KM

4. Conclusions

In the preceding sections we outlined some of the reasons behind the hesitancy to adopt autonomicity into security frameworks. We noted that for reasons pertaining to cost and credibility, business owners prefer to have full control over their security mechanisms. Our aim therefore, was to argue that an autonomic approach can enhance the underlying scheme making the job of the SA easier. In this way the SA can focus on more intellectually demanding tasks.

Specifically, we considered the problems that arise in shared data access scenarios drawing attention to the fact that dynamic key updates are beyond the scope of quasi-static schemes and that the increasing complexity of access scenarios implies by default that effective security would be impossible even with full time manual dedication to the problem. We then proceeded to present a simple but effective *SPKM* framework that does not detract from the basic qualities of the standard security scheme but rather enhances its capabilities with a combination of a stochastic model and replication. The stochastic model determines an acceptable degree of replication to maintain based on an observed arrival rate and the potential impact of checkpointing on the overall performance of the system. In this way backup replicas and keys are generated to preemptively handle situations of high demand. The advantage of this approach is that user transactions no longer need to be delayed while the system adjusts security parameters to cope with requests for changes. Additionally, the SA now only has to preset required parameters and let the system run, without having to be present to manually handle every change.

In a nutshell, if "trust" is defined as the ability of a system to inspire confidence in its users by yielding the results expected of it, then we can affirm that our proposed idea of a *SPKM* scheme addresses this concern in the sense that the design of the *KM* scheme remains the responsibility of the SA while the stochastic processes help to meet performance goals. Can we guarantee that these theories will work well in practice? An implementation and experimentation aimed at testing and validating the hypothesis will probably give a more accurate view. One might also ask what is to stop a lower level user from maliciously attempting to join a group to which they should not belong? The assumption is that the SA would specify what groups a user can join at the authentication level and implement these policies so that they are incorporated autonomatically into the *SPKM* scheme.

In closing, we mention some challenges that need to be addressed in implementing the framework. The assumption that users arrive according to a Poisson process may not always be true. Are there other distributions that may be more effective? Is there a good way to define adequate monitoring thresholds?

References

[1] S. G. Akl and P. D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems*, 1(3):239–248, August 1983.

[2] R. Bastos, F. de Oliveira, and de Oliveira J.P.M. Autonomc computing approach for resource allocation. *Elsevier: Expert Systems with Applications*, 28:9–19, 2005.

[3] K. Birman, R. van Renesse, and V. Werner. Adding high availability and autonomic behaviour to web services. *Proc. of* 26^{th} *International Conf. on Software Engineering (ICSE'04)*, pages 17–26, 2004.

[4] D. Chess, C. Palmer, and S. White. Security in an autonomic computing environment. *IBM Systems Journal*, 42(1):107–118, 2003.

[5] Facebook. Facebook. *http://www.facebook.com/*, 2007.

[6] J. Kephart. Research challenges of autonomic computing. *In Proceedings of 27th International Conference on Software engineering, St. Louis, MO, USA*, pages 15–22, 2005.

[7] M. Mat Deris, J. Abawaly, and A. Mamat. An efficient replicated data access approach for large-scale distributed systems. *Future Generation Computer Systems*, (In Press.) 2007.

[8] M. Mat Deris, J. Abawaly, and A. Mamat. Rwar: A resilient window-consistent asynchronous replication protocol. *In Proc.* 2^{nd} *Int'l Conf. on Availability, Reliability and Security*, pages 499–505, 10-13 April 2007.

[9] N. Post. Credit card information stolen from winners. *http://www.canada.com/nationalpost/story.html*, Jan. 2007.

[10] S. Zhu, S. Setia, and S. Jajodia. Performance optimizations for group key management schemes. *Proc. of* 23^{rd} *International Conference on Distributed Computing Systems (ICDCS '03)*, pages 163–171, 2003.