Toward Unified and Flexible Security Policies Enforceable within the Cloud

David Eyers[1] and Giovanni Russello[2]

[1] University of Otago, NZ
[2] University of Auckland, NZ

Abstract. Security engineering for any given application can usually be done in many different ways. There is often a tradeoff between usability (including efficiency) and the level of protection offered. Typically the risks are assessed by developers, and a particular approach is chosen, with the assumption that the design can stay fixed for some time.

Adoption of Cloud computing will challenge the viability of this approach. Beyond the extra difficulties faced when doing security engineering within distributed systems, Cloud providers require a different threat model from self-hosted resources. They are best considered "trusted but curious" even if the curiosity is accidental on the Cloud provider's part. Some threats from such Cloud providers can be confounded through the use of cryptography, but doing so is overkill in terms of the performance penalty for many applications.

To acquire the benefits of Cloud computing while minimising security risks, we believe that application developers should be provided with the ability to dynamically change the security enforcement technology in use by their software, balancing performance and security as they see fit. Recent cryptography research will significantly increase our ability to offer a runtime choice of contrasting security enforcement approaches *without needing to modify the security policy*. We present our initial research into this area, and outline our vision for the future.

1 Introduction

Application developers need to carefully consider the security engineering of their software. This is particularly true today, as so many devices are network accessible—and indeed need network functionality in order to operate at all—their exposed security surface area is larger than in the past days of isolated microcomputers.

Even software on single machines needs to interact with a number of local software components (broadly defined), simply due to the need to build on top of existing operating system and language runtime codebases.

Two broad types of data security model are Discretionary Access Control (DAC) and Mandatory Access Control (MAC)—see [1] for details. In DAC systems, the owners of protected resources are empowered to change the permissions of those objects, e.g., in Access control lists, capabilities and role-based access control (RBAC).

In MAC models, access control policy is enforced regardless of the desires of the owners of a protected resource. A common illustration is multi-level security models used in the military and other government organisations: data and principals are given

J. Dowling and F. Taïani (Eds.): DAIS 2013, LNCS 7891, pp. 181–186, 2013.

security labels, and policy determines for any data access whether that access will be permitted as a function of the type of access and the labels of the principal and the data, e.g., policy may prevent a principal writing 'top secret' data into a 'classified' resource.

DAC schemes are often enforced by ensuring that code execution paths that access protected resources do permission checking before allowing such access. A risk of this approach is that it relates more to software code than the data it is trying to protect: in some cases access control checks may be accidentally omitted. In other cases this is due to control flow paths allowing implicit access to protected data.

MAC schemes are frequently implemented in a more directly data-centric manner. Since no code path should be able to circumvent the label-based protection, it is appropriate to use techniques to protect the data that may have higher run-time overheads, such as hardware-assisted memory protection. It is impractical to avoid having some trusted computing base (TCB) even in the strictest MAC schemes, but the TCB is ideally isolated entirely from the applications and data it is protecting (often it would be part of the operating system).

One potential approach to ensuring mandatory, data-linked protection against information leakage is to maintain the data in encrypted form, if the TCB is the only part of the code that possesses the decryption keys. Implementing MAC this way may allow for less effort being required to protect the target data: some application-level functionality, such as writing data into a database, might be able to be permitted, despite the data storage functionality not actually being contained within the TCB. This may lead to a more practical system than approaches that require expensive runtime interception of all protected data access.

Regardless of the mechanism used to enforce policy, generally accepted best practice is to abstract security policy away from application-specific implementation code. Widely available implementations of access control technologies such as XACML have helped make it easier to achieve this practice. Maintaining this sort of policy abstraction simplifies making access control policy modifications after software has been deployed.

Security engineering in distributed systems is significantly more complex than for software running on single hosts. However, widespread adoption of Cloud computing will cause distributed security engineering to be required within a growing proportion of applications. The terminology of IETF RFC 2904 [2] includes separate notions of a policy decision point (PDP) and a policy enforcement point (PEP), which highlights the possibility to perform expensive policy checking in a different part of the distributed system than the software component that mediates access to a protected resource.

1.1 Enter Cloud Computing

Cloud computing provides a challenging type of distributed system in which to deploy security-sensitive software. The challenge stems from the Cloud provider being a different organisation from the Cloud tenant, combined with the requirement that Cloud resources need to be accessed across a network. Beyond the need to run above a hypervisor in the first place, Infrastructure as a Service (IaaS) Cloud offerings need to manage the data travelling in a Cloud provider's network between instances of virtual machines, e.g., that host parts of a typical three-tier web application. Then, as Cloud deployment increasingly uses Platform as a Service (PaaS) offerings, a growing number

of application components will be distributed in a way that exposes previously internal security considerations.

Cloud computing providers are most safely treated as "trusted but curious". The curiosity might well be accidental—a provider may leak data from the processes that they use to achieve data backups, for example. However the Cloud provider is clearly "trusted" in that outsourcing to an overtly malicious organisation makes no sense at all.

Put another way, if the primary objectives of computer security are to effect confidentiality, integrity, availability, then we are targeting systems that can provide integrity and availability, but through software or working practice errors, may fail to maintain complete confidentiality.

In this respect, Fully Homomorphic Encryption (FHE) represents the holy grail in Cloud security: FHE allows computers to perform arbitrary computation on encrypted data. FHE protects users' privacy from curious Cloud providers since they are unable to view the users' data. However, the first solution to FHE presented by Gentry [3] is very inefficient. Partially Homomorphic Encryption (PHE) schemes exist supporting limited computations on the encrypted data but in a more efficient way. PHE schemes have been used extensively for supporting search operations over encrypted databases in outsourced settings [4,5,6,7,8,9,10,11,12,13,14,15]. The main issue of these approaches is that they enforce a very basic access control policy: if a user has a decryption key then she is allowed to access the whole database. We argue that more flexible security policies can be enforced if the mechanism for protecting the data from the Cloud provider is separated from the mechanism for controlling access to the data.

1.2 Contribution

This position paper describes our work to increase the flexibility of security in outsourced systems, by proposing a distributed architecture where data protection mechanisms are orthogonal to the security policy enforcement. We aim to allow developers to flexibly tradeoff between speed, simplicity and security for different parts of their applications without needing to rewrite their access control policy.

In particular, application developers can choose between enforcing security using access control barriers, or through the use of encryption. Using barriers will process data more quickly, but the policy enforcement infrastructure must be trusted to see the data that it is releasing to the principals. In the latter case, we extend how cryptographic protection is implemented, by utilising PHE algorithms to allow the policy enforcement point to remain oblivious to the content of the data being released, avoiding an undesired, additional TCB point within the overall distributed system.

2 Encrypted Policy Enforcement

There are several PHE-based solutions that offer encrypted storage of data while allowing basic search capabilities to be performed on the server side without the server learning anything about the plaintext data [4,5,6,7,8,9,10,11,12,13,14,15]. However, these solutions assume all users have the same right to access data. Basically, these solutions lack access control mechanisms to enforce access policies for regulating the access of a

user (or a group of users) to a particular subset of the stored data. To partially alleviate this problem, solutions such as the one described in [16] have been proposed where access policies are encoded as a set of encryption keys. Only users possessing a key are authorised to access the data. The main drawback of this approach is that security policies are tightly coupled with the security mechanism. Therefore, any changes in the security policies require to generate new keys and to redistribute them to the users.

In our research, we argue that the security model used for controlling access to the data should be made disjoint from the mechanism used for protecting the data from the Cloud provider. For instance, most companies use as a security model for protecting their IT infrastructure variations of Role-Based Access Control (RBAC) [17]. In the RBAC model, access decisions are based on the role of the user making a access request. However, typical implementations of this model cannot be deployed on an infrastructure that is not fully trusted, such as Cloud environments because the information they deal with may leak information on the data they are protecting and the internal structure of the organisation. Similar considerations can be made for other MAC and DAC access control models—but for lack of space we focus on the RBAC model here.

We consider an Cloud scenario with outsourced security management involving two service providers: the **Data Service Provider** is responsible for storing the data; the **AC Service Provider** is responsible for the enforcement of access control policies. For reliability, the two services might be provided by different Cloud providers (although this is not a strict requirement for our approach). It is assumed that the Cloud providers are honest-but-curious, that is, they allow the components to follow the protocol to perform the required actions but curious to deduce information about the exchanged and stored data. Figure 1 provides an overview of our distributed architecture for separating policy enforcement from data protection mechanisms.

The responsibility of specifying policies and updating them is that of the **Admin User**. The **Requester** requests access to the data residing in the outsourced environment. The **Company RBAC Manager** is responsible for assigning a role to the requested user (that is, an Admin User or a Requester). The **Server RBAC Manager** is responsible for managing the encrypted role hierarchy graph. The **Trusted Key Management Authority (KMA)** generates and securely transmits the secret keys to the key store. An Admin User (i) gets a role from the Company RBAC Manager and then (ii) deploys the access policies to the Administration Point that (iii) stores the policies in the Policy Store. Meanwhile, the Company RBAC Manager sends the role hierarchy graph that is stored by the Server RBAC Manager (iv).

To request data, a query needs to be encrypted first (0). The Requester then (1) gets a role assigned by the Company RBAC Manager. Next, the Requester (2) sends the request to the Policy Enforcement Point (PEP) which (3) forwards the request to the Policy Decision Point (PDP). The PDP (4) fetches matching policies against the request, (5) collects the contextual information from the Policy Information Point (PIP), and (6) the role information. The PDP evaluates the access request and the contextual information against policies and sends the policy decision to the PEP (7). If the decision is a *deny*, then PEP usually drops communication or replies to the Requester with an error.If the decision is *permit*, the PEP (8) forwards the encrypted query to the Data Store. Finally, the PEP (9) forwards the query results to the Requester.

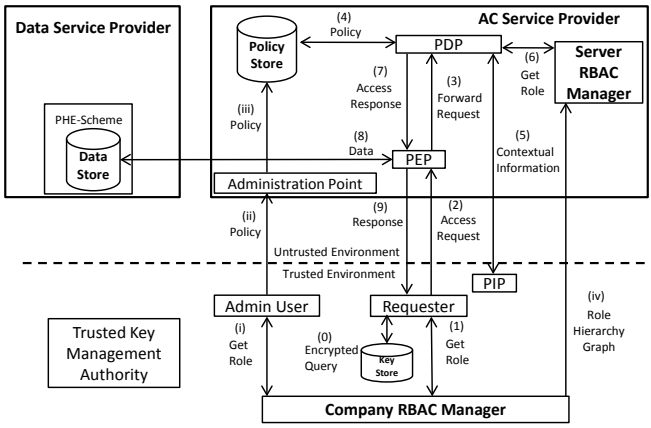

Fig. 1. Our distributed architecture for enforcing RBAC in outsourced environments

The enforcement of encrypted RBAC policies is based on the ESPOON$_{ERBAC}$ system [18]. However, the idea presented in this position paper is to extend the functionality of the PEP to support communication with external service providers where the data is stored. The storage service can use several different types of data protection, independently of the access control model implemented in the AC Service Provider.

The degree of decoupling can be selected by the system designer, depending on their view of the security risk posed by the various different hosting organisations in use within the distributed system. If the trusted/untrusted environment division shown in figure 1 is too conservative for some parts of the application, 'untrusted' components can be moved to the 'trusted' side. Crucially, for the RBAC policies discussed here, this change in trust *does not require the access control policy to be rewritten*.

3 Future Work and Conclusion

In this paper we have discussed how Cloud computing environments have increased the number of participants in distributed access control applications, and how this can negatively impact security. Nonetheless, highest-grade encryption may be overkill for some parts of an application, where the security risks do not justify its use.

Our goal is to develop security technologies for Cloud computing that can flexibly change between software-based protection (i.e., access control monitors) and stronger protection encoded using encryption, without requiring the policy to be rewritten. We have presented an implementation of RBAC that uses the ESPOON$_{ERBAC}$ system, and illustrated how PHE solutions can provide a cryptography-based distributed RBAC enforcement environment. The same RBAC can be checked more cheaply using conventional access control monitors, working from an equivalent policy implementation. Our future work will increase the coverage of policy features that can be implemented using encryption, and to improve the performance of our techniques. We believe our notion of decoupling access control enforcement from the policy specification will be crucial to effect comprehensive security within Cloud computing.

References

1. Department of Defense: Department of Defense Trusted Computer System Evaluation Criteria. (December 1985) DOD 5200.28-STD (supersedes CSC-STD-001-83).
2. Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., Spence, D.: IETF RFC 2904: AAA authorization framework (2000)
3. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 169–178. ACM, New York (2009)
4. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy, SP 2000, pp. 44–55. IEEE Computer Society, Washington, DC (2000)
5. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 216–227. ACM (2002)
6. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
7. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
8. Wang, H., Lakshmanan, L.: Efficient secure query evaluation over encrypted xml databases. In: Proceedings of the 32nd International Conference on Very large Data Bases, pp. 127–138. VLDB Endowment (2006)
9. Bösch, C., Brinkman, R., Hartel, P., Jonker, W.: Conjunctive wildcard search over encrypted data. In: 8th VLDB Workshop on Secure Data Management, Seattle, WA, USA (2011)
10. Popa, R., Redfield, C., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, pp. 85–100. ACM (2011)
11. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: CCS 2006: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88. ACM, NY (2006)
12. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
13. Zhu, B., Zhu, B., Ren, K.: Peksrand: Providing predicate privacy in public-key encryption with keyword search. In: 2011 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2011)
14. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. Journal of Computer Security 19(3), 367–397 (2011)
15. Shao, J., Cao, Z., Liang, X., Lin, H.: Proxy re-encryption with keyword search. Inf. Sci. 180(13), 2576–2587 (2010)
16. Di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Over-encryption: management of access control evolution on outsourced data. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 123–134. VLDB endowment (2007)
17. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer 29, 38–47 (1996)
18. Asghar, M.R., Ion, M., Russello, G., Crispo, B.: Espoon: Enforcing encrypted security policies in outsourced environments. In: ARES, pp. 99–108. IEEE (2011)