Implementation of Cloud-based Access Control and Resource Management System

Ye Wu
Infrastructure Department of TEG, Tencent Inc.
Shen Zhen, P. R. China
yewu@tencent.com

Yifei Yang, Junpei Ouyang
Infrastructure Department of TEG, Tencent Inc.
Shen Zhen, P. R. China
stephyang@tencent.com, robbieouyang@tencent.com

***Abstract*—This article considers the central problems of access control security and resource management in an enterprise-based private cloud. To gain real solutions, we design and implement TAAS system that enforces authorized access to protected data and services as well as provides fine-grained resource management, in support of secure and effective cloud computing. The key components with involved novelties in this work include but not limited to, a light-weight hybrid authentication protocol, a flexible powerful authorization framework, and an efficient resource organization model.**

Keywords-Cloud computing, access control security.

I. INTRODUCTION

Access control security in large-scale distributed system is an increasingly important concern particularly considered in enterprise cloud-based infrastructure, as bulk of sensitive data and privileged services spread over the system, prone to be misused and exposed to internal malicious attacks. To circumvent these threats, a preliminary security platform that protects information confidentiality and integrity is definitely demanded. Moreover, applied in a general trusted enterprise environment, it is desirable to work out a light weight system that not only meets aforementioned security properties, but cooperates with infrastructure primely to present a good performance in processing data and services.

This motivates our investigation, further giving rise to the generation of TAAS project (Tencent Authentication and Authorization Services), a distributed access control system that aims at enforcing authenticated and authorized access to protected resource, guaranteeing information confidentiality and integrity in given context. TAAS authentication service is responsible for recognizing and validating requester's identity and role, as a result of authenticity to access our cloud infrastructure services, such as relative job control and file control APIs, provided by distributed scheduling system (named TORCA) and distributed file system (named XFS) respectively. More interestingly, applications of this service are not limited to the authentication merely for aforesaid products, but easily incorporate new entities to authenticate one another. Therefore, probably some authentication path across multiple access points comes into being, say arising from application work flow in cloud. On the other hand, TAAS authorization system implements an advanced R-ABAC model (Role and Attribute Based Access Control) as its core, to grant the access based on authentication result and specific policy. This is organized and developed as an authorization framework, for applicability and scalability.

Effective resource management can also benefit from access control security at least at its conceptual level, say the central notion on *role*. In our cloud, it is associated with the amount of resources like computation and storage quotas. Further, we propose a role-based delegation mechanism applied to resource management model, to leverage its flexibility and usability.

Our contributions to this work are that: Design and implement a highly efficient light-weight authentication protocol, mixed with single-sign on and trusted delegation features. Propose a novel hybrid R-ABAC model not only reducing permission administration but facilitating flexible access authorization, along with its industrial strength implementation. Develop a scalable and effective cloud resource management system in practice.

We introduce TAAS architecture and its interaction with other components in our cloud infrastructure in section II. Section III depicts authentication protocol for TAAS and discusses related issues. In section IV, we consider an authorization framework that intends to unify miscellaneous authorizing contexts and discipline, followed by an introduction to our resource management model in section V. Experimental results are presented in section VI. Lastly, we give the related work and references.

II. TAAS ARCHITECTURE

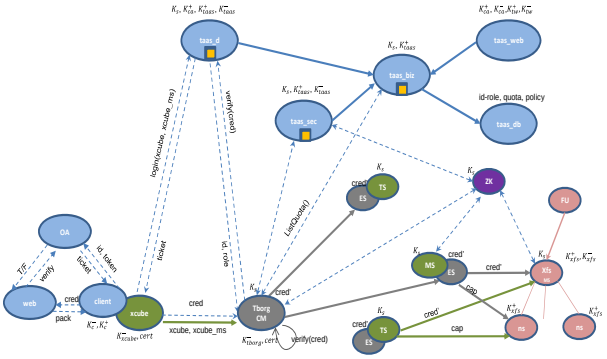

Figure 1. TAAS system and interoperation with other cloud components.

Briefly speaking, TAAS is primarily composed of two groups of sub-systems, represented by blue ovals in *Figure 1*.

- *Credential generation toolkit (left corner, three blue ovals)*: it is used to create user's private key and certificate for further access control purposes. Private key is to generate signature that demonstrates user authenticity, being testified by corresponding public key in subsequent usage, while the certificate obtained

here is the binding of public key and user identity, issued by our web-based authority that utilizes the authentication service offered by firm OA. With which, employees are supposed to show up their SecureID for taking advantage of this toolkit. Notice that, at this stage it is essentially a password-based authentication honored by trusted third-party (firm OA), resulted in a provable identity comes into being for the future uses.

- *Core system (upper right, five blue ovals)*: this accomplishes TAAS major functionalities, including identity authentication, privilege validation, credential release, access authorization, and resource management. We discuss these issues in details subsequently. By now, we primarily focus on its constructions. The most striking one is authentication center and capability certificate issuer (*taas_d*), responsible for verifying user identity and its claimed role so as to authorizing a role attribute-based certificate (ticket interchangeably) if succeed. Therefore, user possessing such a certificate is analogous to hold a *proof-carrying* exhibit in hand, with which to request accesses to targets. The second component is a security coordinator (*taas_sec*) for shared key distribution, also in charge of distributed access toward information-flow control, an ultimate end-to-end security property. The third one is a collection of DB oriented API services (*taas_biz*) that mediate the access to sensible data. Others like web front-end (*taas_web*) and databases (*taas_db*) are supplied as usual, particularly for supporting resource management. Here, we omit to present an important component in Figure 1, an authorization library which is the implementation of authorizing framework.

Considering interactions between TAAS and other components in our cloud infrastructure, this is exemplified in lower part of Figure 1 (different colored ovals represent different systems). For instance, imaging a distributed database system (XCUBE in green) launched as a map-reduce job, a client application, to be submitted to TORCA master (in grey). In turn, this job is dispatched as many tasks running on the workers (ES in grey), one of which is taken as XCUBE Master (MS), others as XCUBE workers (TS). They probably have to access XFS system (in pink) for pulling or storing data from which. For security concerns, access control is inevitable while these sorts of accessing requests occur across different component domains in the work-flow. This leads to the interactions with TAAS to perform authentications, represented by some dotted lines in figure.

III. TAAS AUTHENTICATION

A. Authentication Protocol

We design and implement such a novel protocol that combines the merits of PKI-based and Kerberos-based authentication protocols, say offline credential generation and single-sign on (SSO) properties, while harnessing system TCB for trusted delegation. These techniques reasonably mitigate the authentication workload, resulting in a light-weight authentication architecture. We intuitively outline this protocol by Figure 2 and the key points there, followed by its formalization in Figure 3.

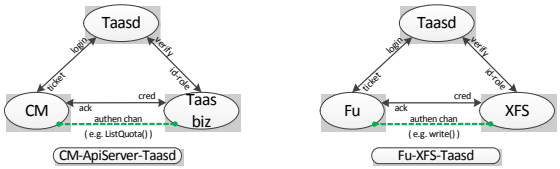

Figure 2. TAAS authentication core procedure.

The left and right diagrams in Figure 2 represent generic client-server authentication scenarios by using TAAS service. Say in the left one, (TORCA) CM intends to authenticate itself to Taas biz service in hope of establishing channel. Recall that our toolkit has already generated necessary credentials offline for client CM to prepare future authentications. This is the start point where CM takes *login* action. In this step, CM's credential is sent to Taasd for validating its identity and claimed privilege assumed by role. Were it succeeded, a ticket and session key will be sent back to CM, which is a PKI-based authentication scheme rolling up. Having received crypto, CM makes its authenticator then concatenates with ticket, assembled together as a new credential standing for its *proof-carrying* identity and capability. Thereafter, credential is delivered to Taas biz for verification, in turn, to be forwarded to Taasd that is the authority appropriate for this job. Later on, verification result echoes back to Taas biz. Based on the answer, it knows whether that requesting channel should be established or not. Observe that, a green dotted line in figure exactly indicates the success of authentication and the buildup of authenticated channel. To this end, it is not surprising to think of the main part of our protocol as Kerberos-based. The true story is that we did more than that, reflected in our proposed novel adaption and optimization for this protocol design, such as, consecutive authentication across domains, group key issues, information flow control and so forth, part of which are addressed below.

In contrast to intuitive illustration above, we also give a formal semantics of this authentication protocol in *Figure 3*, to help people understand its core procedure more precisely.

B. Some Advanced Topics in TAAS Authentication

- *Access control security vs. information-flow security*: the former enforces authorization merely at access point, however the latter pursues an end-to-end security goal as information propagated over the system. For instance, due to work-flow, user's credential must be exposed at different places where access control is required and conducted. Guaranteeing such data confidentiality and integrity is a notorious hard problem both in theory and practice, to which we develop a security coordinator (*taas_sec*), dynamically regulating credential usage in work-flow whenever extruding its original domain.
- *Proof-carrying authentication and authorization (PCA)*: this is a promising technique particularly appropriate for distributed and decentralized access control. A

typical scenario can be found in our work, where an anonymous user may access to XFS data node to execute read-write actions. Due to performance considerations in cloud computing, it is unrealistic for such data nodes to carry out regular access control, instead, a feather-weight verification imposed on user's certificate is expected for simply deriving its capability.

- *Attack models*: as applied in a private enterprise cloud, certain trustworthy is assumed so that we rather than too care about malicious behavior happened in network transition, but focus on replay attack instead, because in this context it is reported to be the most common attacking pattern. We resolve this problem inspired by the idea in Kerberos. Novelty relative to this part shows that sort of address-based attack is prevented without interfering application work-flow, thanks to our security coordinator that safely alters IP in credential.
- *Certificate revocation*: this is a serious problem in pure PKI-based authentication system. In our work, right after early PKI related *login* procedure, protocol steps into Kerberos-alike processing without such hassles intrinsically. In fact, certificate in TAAS is issued by centralized authority (*taas_d*) that stamps cert with short session time, effectively avoiding related problem.

$$client \rightarrow taas_d: \quad cred_1, \quad where\ cred_1 \triangleq cert_c \oplus auth_1$$
$$cert_c \triangleq \{id_c, K_c^+\}\ K_{ca}^-$$
$$auth_1 \triangleq \{idc, role, T_1, times\}\ K_c^-$$
$$T_1 \triangleq time\ stamp$$
$$times \triangleq T_b, T_e, T_r$$
$$taas_d: \quad isOk = \textbf{let}\ m_1 = \{cred_1|_{cert}\}K_{ca}^+, m_2 = \{cred_1|_{authen}\}m_1|_{key}\ \textbf{in}\ m_1|_{id} \equiv m_2|_{id}\ \&\&\ m_2|_{id} \in m_2|_{role}$$
$$keypack \triangleq \textbf{new}\ K_{ss}\ \textbf{in}\ \{Kss, t\}\ m_1|_{key}$$
$$ticket \triangleq \textbf{let}\ m = (id_c, role, times, IPc, Kss)\ \textbf{in}\ \{m, \{H(m)\}K_{taas}^-\}K_s$$
$$taas_d \rightarrow client: \quad \textbf{if}\ isOk\ \textbf{then}\ return\ (keypack \oplus ticket)\ \textbf{else}\ return\ err$$
$$client: \quad auth_2, \quad where\ auth_2 \triangleq \textbf{let}\ m = \{keypack\}K_c^-\ \textbf{in}\ \{id_c, t'\}m|_{key}$$
$$client \rightarrow server: \quad cred_2, \quad where\ cred_2 \triangleq ticket \oplus auth_2$$
$$server \rightarrow taasd: \quad cred_2$$
$$taas_d: \quad isAuth = \textbf{let}\ m_0 = \{cred_2|_{tk}\}K_s, m_1 = m_0|_{msg}, m_2 = \{cred_2|_{authen}\}m_1|_{key}\ \textbf{in}\ H(m_1) \equiv \{m_0|_{hval}\}K_{taas}^+\ \&\&\ m_1|_{id} \equiv m_2|_{id}\ \&\&\ m_1|_{ip} \equiv ReqIP$$
$$taas_d \rightarrow server: \quad \textbf{if}\ isAuth\ \textbf{then}\ return\ (idc, role)\ \textbf{else}\ return\ err$$

Figure 3. Formalization of TAAS authentication protocol.

IV. TAAS AUTHORIZATION

The core of this part consists of a policy model and an access decision procedure. To the former, we invent R_ABAC as an effort to reduce the complexity of policy administration and realize flexible fine-grained authorization. To the latter, we implement a generic framework aimed at usability and scalability, akin to Java security architecture.

A. R_ABAC Policy Model

We adapt $RBAC_0$ as the underpinning, combined with ABAC model for rule-based constraint. To do so is primarily arising from the real needs. For example, whether a user can be authorized to write to a specific XFS file depends on not only her role that represents permissions, but role-associated storage quota that must agree with quota threshold for write enable. Such threshold can be viewed as a role attribute, the constraint that prescribes minimum quota value for certain operations allowed. Thus, a need for R_ABAC emerged. One instance in practice is presented in Figure 4.

RBAC Policy		
Role	Cluster	Permissions
R_1	C_5	(AllAct, $Files_{R1}$)
R_2	C_8	(AllAct, $Files_{R2}$)
...	...	...

Role-attribute Constraint Policy				
Role	Cluster	Quota	#Files	#Dirs
R_1	C_5	20G	3000	200
R_2	C_8	40G	6000	400
...	...	...	...	...

Figure 4. R_ABAC policy example.

B. Implementations

The lower half of Figure 5 delineates main constructs of authorization lib, important interfaces as well as a UML-alike time chart to present interactions. As we have seen, these are implemented based on relative standard framework, following PAP, PEP, PDP, et al design scheme, if abstract.

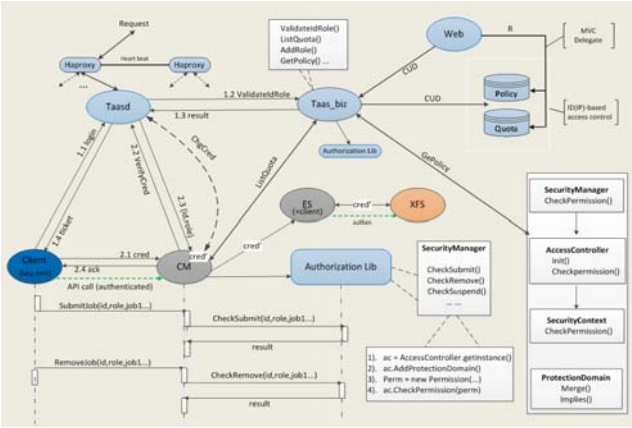

Figure 5. TAAS authorization framework.

V. TAAS RESOURCE MANAGEMENT

In our cloud platform, TAAS is in charging of the resource management not only for identity, role, and policy relativized to security, but for computation and storage quota that as aforesaid, are allocated on the basis of *role*. The central problems existing in such schema are that, one is about how to classify these resources and organize them in a logical hierarchy based on their affiliation; the other is about how to draft application workflows that influence the way of applying resource, processing approval, and auditing results. Obviously, the former concentrates on structuring a well-organized extensible architecture, while the latter emphasizes the flexible and accountable dynamic utilization of resources. Toward the specifications and goals above, we design and implement a management model in Figure 6 below.

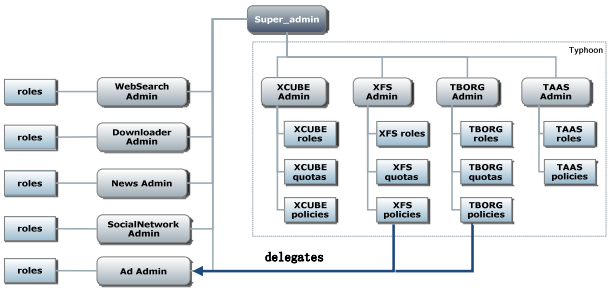

Figure 6. TAAS resource management model.

As seen, the system is logically layered as super-admin, cluster-admin and project-admin in hierarchy, each of which is responsible for managing and maintaining resources pertain to whom, say roles, as well as dynamic resource applying and partial approval at disposal. A worth mentioning point is that we introduce a *delegates* relation from cluster-admin to certain project-admin, for the designated cluster is solely possessed and used by some project. Doing so is to consider resource usability and management flexibility

VI. Experimental Results and System Deployment

Since TAAS authentication mechanisms, *login* and *verify*, are essentially implemented in network application layer toward secure communication, we are more concerned with the overhead for creating such an authenticated channel, which reflects in overall response time and throughput comparing with that of non-secure channel. Figure 7 shows the related testing outcome. An exciting statistical result manifests that such overhead is less than 3.1%.

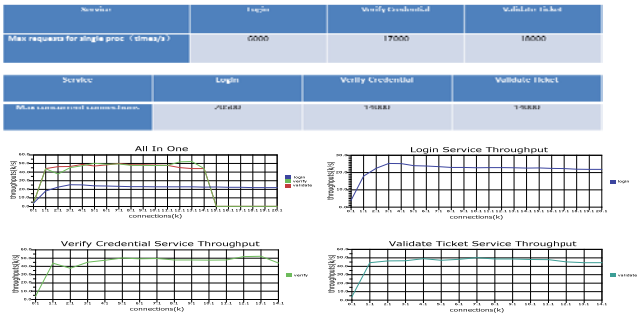

Figure 7. Experiment results of response time and throughput.

To achieve system reliability and localized service, Taas is deployed based on IDC described in Figure 8, where totally four suits of identical services are set up in cities: SZ, XA,TJ, and SH respectively. In case one site crashes, any of other three will take effects to continue serving. In fact, to meet reliability ends, one of our measures beside such redundancy is of an advanced fault tolerance mechanism, in support of regular business processing even if all TAAS services break down.

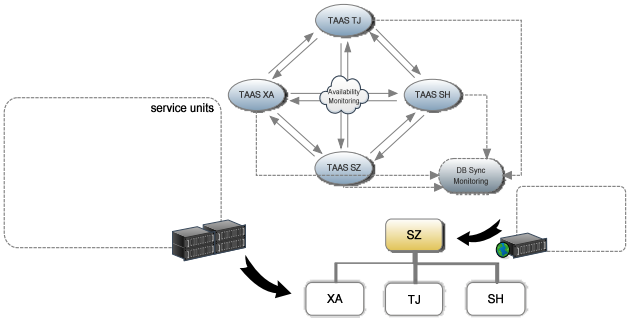

Figure 8. TAAS system deployment and monitoring.

Another diagram showed in Figure 8 depicts monitoring of TAAS service availability and data consistency, which is used to inform operation team with service running status, and notify relative faults in a real-time manner, if happened.

Acknowledgment

We are very grateful to our colleagues and leaders, for their insightful suggestions, continuous help and contributions to this work. They are Guang Yang, Allen Liu, Tao Wu, Bo Li, Hanmei Luo, Dafu Deng, Xin Yao, Jun Chen and Dr. Huican Zhu.

References

Lampson et al [1] consider authentication issues in distributed system and develop a logic to reason about its correctness. Appel and Felten [2] design and implement an authentication framework with a higher-order logic to model some existing related systems. Inspired by such work, we also figure out an application-specific logic to testify authentication soundness in TAAS. RBAC models are clearly explained in foundation work [3], while an ABAC model is proposed in [4]. We combine the merits of these two toward simplicity and flexibility in policy administration. Some source of information-flow security can be found in [5] [6], where Duggan and Wu propose some novel solutions for which. Common knowledge in access control security, particularly related to its system design is introduced in [7].

[1] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, “Authentication in distributed systems,” *ACM Trans. Computer Systems*, 1992.

[2] A. Appel, E. Felten, “Proof-Carrying Authentication,” ACM Conference on Computer and Communications Security, 1999.

[3] R. S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, "Role-Based Access Control Models," *IEEE Computer* 29(2): 38-47, IEEE Press.

[4] E. Yuan, J. Tong, “Attributed Based Access Control (ABAC) for Web Services,” International Conference on Web Services (ICWS'05).

[5] Dominic Duggan, Ye Wu. “Secure Nested Transactions,” IEEE Symposium on Security and Privacy, 2011

[6] Dominic Duggan, Ye Wu. "Causality and Accountability", Formal Aspects of Security and Trust, LNCS, 2009.

[7] B. Lampson, “Practical Principles for Computer Security,” Software System Reliability and Security, 2006.