# Multiple-Tiered Security Hierarchy for Web Applications Using Central Authentication and Authorization Service

Hisashi Naito
Graduate School of Mathematics
Nagoya University
Nagoya 464-8602 JAPAN
naito@math.nagoya-u.ac.jp

Shoji Kajita
Information Technology Center
Nagoya University
Nagoya 464-8601 JAPAN
kajita@nagoya-u.jp

Yasushi Hirano
Information Technology Center
Nagoya University
Nagoya 464-8601 JAPAN
hirano@itc.nagoya-u.ac.jp

Kenji Mase
Information Technology Center
Nagoya University
Nagoya 464-8601 JAPAN
mase@nagoya-u.jp

## Abstract

*The Central Authentication Service (CAS) is a middleware for constructing a Single Sign On infrastructure for Web applications and has been developed by JA-SIG. In this paper, we investigate a multiple-tiered security hierarchy infrastructure for Web applications, by extending CAS to the Central Authentication and Authorization Service (CAS$^2$). Since the new version of CAS (CAS3) supports the X.509 client certificate authentication, we use it as leverage to realize our multiple-tiered security hierarchy mechanism. As a result, CAS$^2$ uses X.509 client certification for not only authentication, but also authorization.*

## 1. Introduction

Most higher educational institutions are struggling to provide efficient and effective information services due to the hierarchical and silo structure of service units. For example, a section that serves students independently works from a section that serves faculties for research. Therefore, each section has a tendency to provide its own information system that has its own authentication and authorization functions. As a result, various authentication and authorization functions are introduced; however, the existence of many authentication functions gives rise to a lot of authentication data for each person. It is well-known that such a situation induces not only increases in management costs but also decreases the security of information systems. Therefore, institutions are trying to introduce "unified authentication databases" to decrease management costs and to strengthening security.

However, each information system having permission to access the unified authentication database increases risks of information leakage. This situation makes it difficult to realize a secure unified authentication database. To overcome it, we have previously introduced the Central Authentication Service (CAS) to realize a secure unified authentication database under a variety of hierarchical and silo structures in institutions [1].

CAS has enough functions as a Single Sign On infrastructure for Web applications, but it has no functions for authorization. Therefore, as an enhanced application of CAS, we developed Central Authentication and Authorization Service (CAS$^2$) to manage authorization of access for each Web application, and we achieved satisfactory operation results [2, 3, 4].

Recently, the use of X.509 client certification with Public Key Infrastructure (PKI) has been emerging to increase security level of information systems. CAS Version 3, released in 2006, supports both PIN authentication (Username/Password authentication) and X.509 client certification. However, it is not satisfactory for controlling access to information systems with various security levels.

In this paper, we introduce the notion of "security hierarchy" for each information system, and use CAS$^2$ to control authorization management based on a multiple-tiered security hierarchy.

Proceedings of the 2007 International Symposium
on Applications and the Internet Workshops (SAINTW'07)
0-7695-2757-4/07 $20.00 © 2007 IEEE

IEEE
COMPUTER
SOCIETY

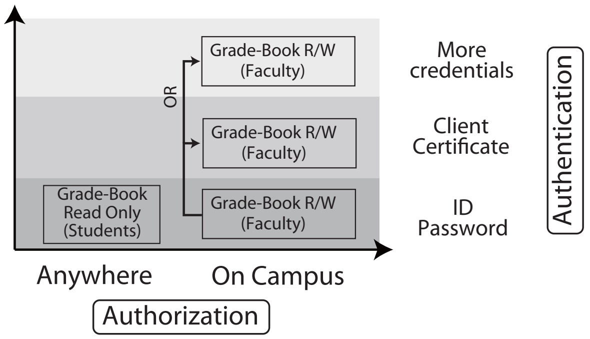

**Figure 1. Multiple-tiered security hierarchy**

## 2. Central Authentication and Authorization Service

The main features of CAS can be described as follows.

- CAS does not have own databases for authentication, and can handle many types of external database such as LDAP directory servers, SQL databases, and Radius.
- Web applications can obtain users' authentication results and information on attributes of authenticated users using CAS. However, they receive no data for authentication.
- CAS uses only such fundamental Web technologies as http redirection, Java scripts and cookies in its authentication processes.
- It is only necessary to fit a library (module) called a "CAS client" to Web applications in order to adopt CAS. Thus it is easy to adapt Web applications to CAS.

### 2.1. Authentication Mechanism of CAS

The authentication mechanism of CAS uses the following two tickets.

- "Ticket Granting Cookie" (TGC): The cookie that is issued by the CAS server to each authenticated browser.
- "Service Ticket" (ST): The "One Time Ticket", which shows that each access to a Web application is valid. An ST is sent to a Web application from the authenticated browser as a URL parameter, when the browser accesses it.

The mechanism CAS uses to apply these tickets is as follows.

1. When a browser (user) accesses a Web application without any ST, the CAS client module in the Web applications issues an http redirection to the CAS server.

   After the redirection, the CAS server attempts to authenticate the browser if the browser does not have a valid TGC (see Fig. 2).

2. If the CAS server successfully authenticates the browser using external user databases, CAS server then obtains the user's ID from the external databases. Furthermore, the CAS server issues a TGC, sends it to the browser, and performs a redirection to the Web application by using the parameter in the redirection of the first step. In this step, the ST is inserted in the redirective URL (see Fig. 3).

3. The Web application that accepts the access contains an ST as a URL parameter that can verify the ST by sending it to the CAS server (Fig. 4). The Web application continues its normal processing for the access granted by a valid ST.

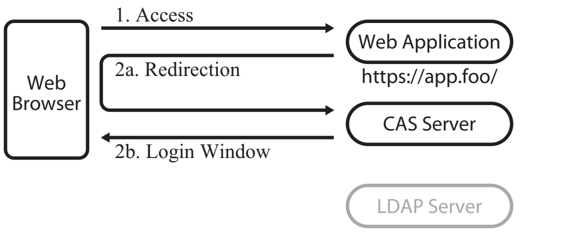

**Figure 2.**

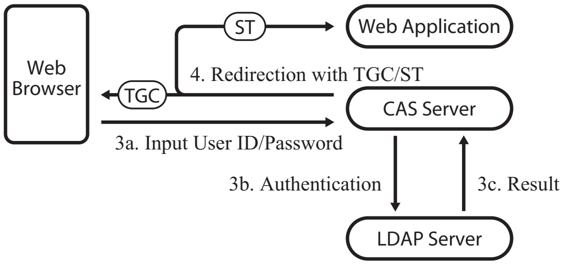

**Figure 3.**

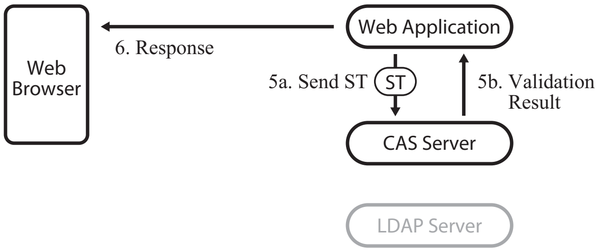

**Figure 4.**

Proceedings of the 2007 International Symposium
on Applications and the Internet Workshops (SAINTW'07)
0-7695-2757-4/07 $20.00 © 2007 IEEE

### 2.2. Authorization Mechanism of CAS$^2$

We introduce the following access authorization mechanisms into the processes of the CAS authentication mechanism in CAS$^2$.

- CAS$^2$ handles an external authority management database, which is independent from CAS$^2$. The database is called the “Access Control List” (CAS-ACL).
- CAS$^2$ verifies each ST issued during the authentication process. Access is granted only when CAS$^2$ judges the ST to be valid.
- CAS$^2$ collates the access with CAS-ACL based on “When,” “Who,” and “Where” using the ST.
- An access attempt to a URL not contained in CAS-ACL is judged to be invalid by CAS$^2$.

A typical CAS-ACL entry is as follows. (Since we first constructed CAS-ACL on an LDAP Directory Server, it is described using the LDAP DIT form. However, CAS-ACL can also be constructed on other kinds of databases.)

```
dn: cn=portal,ou=cas,o=NagoyaUniversity
cas-service: https://app\.foo/.*
cas-allow: (|(uid=naito)(uid=kajita))
cas-attributes: uid,username,mail
```

The actual collation method of CAS-ACL is processed in the CAS authentication mechanism as follows.

1. CAS$^2$ collates the URL with CAS-ACL when it issues an ST. (see 1 of “Authentication Mechanism of CAS” and Fig. 5). For an access that is judged as invalid, CAS$^2$ produces a page with the message “Access denied” on the browser. However, CAS$^2$ issues TGC if the authentication is valid.
2. CAS$^2$ collates the access with CAS-ACL upon verifying the ST. (See 3 of “Authentication Mechanism of CAS” and Fig. 6). If the ST is valid and granted, CAS$^2$ sends values of the user’s attributes shown by the attribute values of `cas-attributes` in the CAS-ACL entry.

We collate each access with CAS-ACL twice when an ST is issued and verified by the above-mentioned process. In the first stage, the CAS-ACL key is stored in the ST database, and CAS$^2$ compares the key obtained by the ST verification and the result from the previous stage. Consequently, we can prevent Man-in-Middle attachment due to falsification of the access destination.

The collation method with the access authorization using CAS-ACL is as follows.

1. The CAS server searches an entry of CAS-ACL whose `cas-service` attribute matches the access destination included as a URL parameter by regular expression.
2. If and only if the Boolean expression shown by the `cas-allow` attribute in the CAS-ACL entry returns “true,” the CAS server grants access.

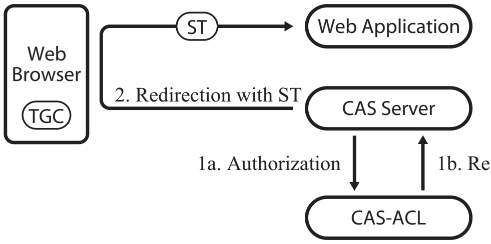

**Figure 5.**

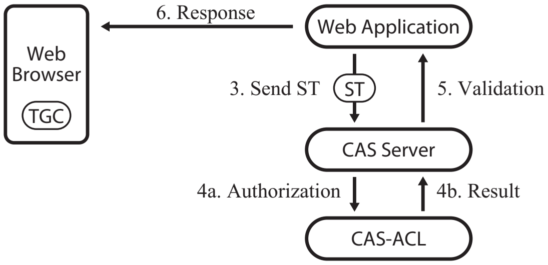

**Figure 6.**

## 3. X.509 client certification

We consider that access to important personal information (for example, a grade-book system) may not be restricted by conventional authentication methods and must instead be restricted by a strong user authentication method. One a strong user authentication method is to use X.509 client certification stored in the IC cards, which hold is each user’s ID card.

CAS Version 3 is equipped with an authentication function by an X.509 client certificate, and this is easy to use. However, Web applications using CAS do not always employ authentication as strong as an X.509 client certificate. For example, let us consider that a BBS system and the grade-book system use the same CAS server. We may require that users employ X.509 client certification to access the grade-book system. On the other hand, students may use the BBS system from a public terminal that is not equipped with any IC card reader. When two or more Web applications are operated on a unified authentication database like this, the case frequently arises that the required security level of each Web application is different.

To solve this problem, we introduce a mechanism of access control for Web applications that requires different security levels using CAS$^2$, and we control access based on the requested security level for Web applications.

### 3.1. Definition of security hierarchy and authorization mechanism using security hierarchy

We define the notion of **security hierarchy** using CAS-ACL to correspond to many Web applications that require different security levels. Actually, we add the `cas-security-hierarchy` attribute to entries of CAS-ACL, and defines for instance, that users authenticated by PIN can access Web applications with "class-1 security," and that users authenticated by X.509 client certification can access Web applications with less than "class-2 security" (see. Figs. 7 and 8).

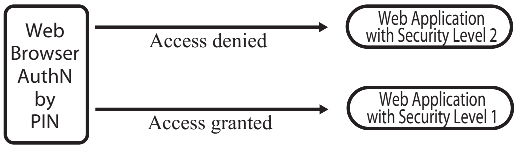
Web Browser AuthN by PIN
Access denied
Web Application with Security Level 2
Access granted
Web Application with Security Level 1

**Figure 7.**

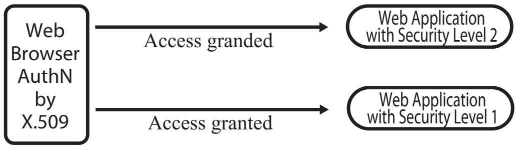
Web Browser AuthN by X.509
Access granded
Web Application with Security Level 2
Access granted
Web Application with Security Level 1

**Figure 8.**

According to that this definition, CAS$^2$ requires X.509 client certification when the users authenticated by only PIN access Web applications with class-2 security. The authorization procedure using CAS$^2$ is as follows.

1. If a user tries to log in by using a browser without X.509 client certification, CAS$^2$ shows the login window for PIN authentication and stores the fact that the user logged in at security level 1. If a user tries to log in by using a browser with a certification, CAS$^2$ verifies the certification and stores the user's log-in at security level 2 (see Fig. 9).

2. If a user logged in at security level 1 tries to access a Web application with class 2 security, CAS$^2$ issues a redirection to require X.509 client certification, since ST verification using CAS-ACL fails. (see Fig. 10).

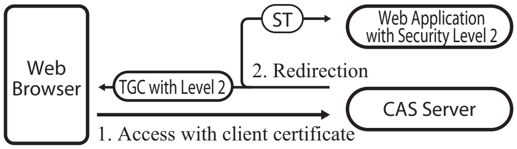
Web Browser
ST
Web Application with Security Level 2
2. Redirection
TGC with Level 2
CAS Server
1. Access with client certificate

**Figure 9.**

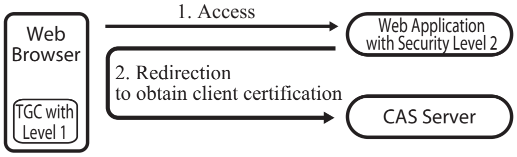
Web Browser
TGC with Level 1
1. Access
Web Application with Security Level 2
2. Redirection to obtain client certification
CAS Server

**Figure 10.**

## 4. Summary and Future works

In this paper, we discussed a multiple-tiered security hierarchy infrastructure for Web applications. This study focused not only on authentication infrastructure, but also on unified authorization management according to the security level that the Web application demands. As a result, we constructed a hierarchical authentication and authorization infrastructure using CAS$^2$.

Our future works include designing an inter-institutional authentication and authorization infrastructure. For example, it will be necessary for institutions to provide a wireless network service for visitors from another universities. In such a situation, we need a federated authentication and authorization infrastructures like Shibboleth [5].

## References

[1] JA-SIG. http://www.ja-sig.org/products/cas/.

[2] H. Naito and S. Kajita. CAS as an institutional-wide authentication and authorization infrastructure. http://www.ja-sig.org/sites/isapps/jasig/2005SummerBaltimore/. 2005 Summer uPortal Conference, Baltimore, USA.

[3] H. Naito and S. Kajita. Single Sign On and authorization infrastructure using CAS2. http://www.apan.net/meetings/tokyo2006/. 21st Asia Pacific Advanced Network (APAN) Meeting 2006 Tokyo, Middleware Workshop.

[4] H. Naito, S. Kajita, T. Kojiri, Y. Hirano, and K. Mase. CAS and its extension as an unified authentication authorization information infrastructure for institutions. *Journal of Information Processing Society of Japan*, 47(4):1127–1135, April 2006.

[5] Shibboleth Project. http://shibboleth.internet2.edu/.