CAS as an Institution-wide Authentication and Authorization Infrastructure

Hisashi NAITO

naito@math.nagoya-u.ac.jp

Graduate School of Mathematics, Nagoya University

Why do we need Authorization for CAS?

- Default Behaviour of CAS
- Security Problems

Default Behaviour of CAS

- **ANY** Web Application Server can use CAS **Ticket Validation Servlet**.
- **ANY** Web Application Client (Web Browser) can use CAS **Login Servlet**.
- CAS Server can not distinguish **User Class**.
- CAS Server always responses **uniform** User Information (eg. User's Fullname el. at.) for any Web Application Server, if Service Ticket is VALID.

Security Problems (Cross Site Scripting)

- Since CAS Server does not check **`service`** parameter, the following URL **Ticket Granting Cookie** is discovered:

```
https://mynu.jp/cas/index.jsp?
service=javascript
%3aalert%28document.cookie%29%3b//
```

The above URL arises the TGC

```
CASTGC=TGC-1-Fjlfdsjflsa789w3ljluoilsjl
```

What do we want to do?

- Want to make different ACCESS PERMISSION between Web Applications.
- Want to make different RESPONSES between Web Applications and/or URL.
- Want to make different ACCESS TIME between Web Applications.
- Want to easy to make to CASified for any Web Applications.

What do we want to do?

- Want to make different ACCESS PERMISSION between Web Applications.

Example

- A Web Application can be accessed by **ANY** Students from **ANY** client on Internet.
- Another Web Application can be accessed **ONLY** by Institution Faculty and **ONLY** from interior University.

What do we want to do?

- Want to make different RESPONSES between Web Applications and/or URL.

Example

- Reply “User’s Fullname”, “User ID” and so on for a Web Application.
- Reply **ONLY** “Authentication Result” (Validness of Service Ticket) for another Web Application.

What do we want to do?

- Want to make different ACCESS TIME between Web Applications.

Example

- Can access 24 hours on everyday for a Web Application.
- Can access **ONLY** on Working Time for another Web Application.

What do we want to do?

- Want to easy to make to CASified for any Web Applications.
  - Web Applications can equipped an authorization mechanism by itself, but it is not easy to equip the mechanism.
  - Concentrate authorization mechanism to CAS, then it is easy to maintain authorizations for Institution-wide Web Applications.

Solution – Service Based Authorization Mechanism

- Service Based Authorization
  - **Access Control Lists** (ACL)
    - The ACL is an exterior database of lists of Web Applications and datum of Access Controls.
    - The ACL is stored in LDAP Server or ....
  - **Access Control Class** (ACC)
    - A ACC is a class of URLs expressed by Regular Expressions.
  - Every Service Ticket belongs a class of ACL.
  - Authorization by Service Ticket.
    i.e., Authorization by ACL.

Example of ACL (LDAP DIT)

```
dn: cn=uPortal,ou=uPortal,ou=cas,o=NU
cas-auth-type: basic
cas-attributes: uid,MailAddress,IdNo,
 Fullname,username,dn
cas-service: https://nu\.jp/uPortal/.*
cas-allow: (dn=.+,ou=place.?,o=nu)
```

```
dn: cn=aApp,ou=uPortal,ou=cas,o=NU
cas-auth-type: basic
cas-attributes: uid
cas-service: https://nu\.jp/APP/.*
cas-allow: (&(dn=.+,ou=place.?,o=nu)
 (&(time>=0900)(time<1700)))
```

Example of ACC

```
https://nu\.jp/uPortal/.*
```

by **dn: cn=uPortal,ou=uPortal,ou=cas,o=NU**

```
https://nu\.jp/APP/.*
```

by **dn: cn=aApp,ou=uPortal,ou=cas,o=NU**

ACL/ACC

- Access for **`https://nu\.jp/uPortal/.*`**
  - restrict to USER matches "**`dn=.+,ou=place.?,o=nu`**".
  - Service Validation Servlet replies USER INFORMATION **`uid`**, **`MailAddress`**, **`IdNo`**, **`Fullname`**, **`username`**, **`dn`**.
- Access for **`https://nu\jp/APP/.*`**
  - restrict to USER matches "**`dn=.+,ou=place.?,o=nu`**", and from 09:00 to 17:00.
  - Service Validation Servlet replies USER INFORMATION **`uid`** ONLY.

ACL/ACC

- We can describe **`cas-allow`** entry in ACL by using
  - index of User Entry – **`dn`** in LDAP DIT.
  - any attributes in User Entry – any attributes in LDAP DIT.
  - Access date/time **`time`**, **`date`**, **`datetime`**, **`wday`**.
  - Combines the above elements by Polish Notation.
- We can describe **`cas-attribute`** entry in ACL by
  - any attributes in USER DATABASE. (any attributes in LDAP User entry.)
  - provides **`nextticket`** or not.

When do we authorize accesses?

- Check ACL when CAS server **issues new Service Ticket**.
  - CAS server verifies the user is permitted to access the service or not.
- Check ACL when **user accessed to a service**.
  - Service Validation Servlet does the followings:
    - compares actual **`service`** parameter and stored ACC in ST-database.
    - verifies the user is permitted to access the service or not.

Why do we need twice authorization?

First Authorization:

- To provide **`nextticket`** by CAS Server.

Last Authorization:

- Possibility **alteration** of the Service Parameter. (**Man in Middle Attack**)

Administration for ACL

- ACL is stored in an external database.
- Administrators can reload ACL in any time by using **Admin Servlet** of CAS server.
- Access Control for Admin Servlet is itself controlled by the **`trust`** entry in ACL.

Example of trust entry in ACL

```
dn: ou=cas,o=NU
cas-allow: (uid=kajita)
cn: trusted
```

“**kajita**” can reload the entire ACL.

```
dn: ou=AnotherDIT,ou=cas,o=NU
cas-allow: (uid=naito)
cn: trusted
```

“**naito**” can reload the ACL only subtree “**ou=AnotherDIT**”.