CAS^2 を利用した SSO と Authorization 環境

内藤久資 [1,3], 梶田将司 [2,3], 平野靖 [2], 間瀬健二 [2,3]

[1] 名古屋大学多元数理科学研究科
[2] 名古屋大学情報連携基盤センター
[3] 名古屋大学情報連携統括本部情報戦略室

名古屋大学では、Central Authentication and Authorization Service (CAS^2) を利用して、Web Application に対する統一認証基盤上の Single Sign On と Authorization 環境を構築した。本発表では、CAS^2 の SSO/AuthZ メカニズムの概要と２年間に渡る運用経過について述べる。また、よりセキュアな認証環境を実現するための最近の実験的な試みについても言及したい。

Plan of Talk

- **研究の動機と背景**
- Brief survey of Single Sign On using CAS
- Brief survey of Authorization Environment using CAS^2
- **名古屋大学での運用実績**
- **最近の実験的な試み**
- Summary

研究の動機と背景 > マジメな話

- 大学内の情報システムは複数の部局・部門が個別に管理している
- ユーザは複数の情報システムを利用しなくてはならない.（複数の情報システムの利用を強いられている）
- Example
  - 教務システム（成績入力システム）：学務部学務課
  - 研究者情報データベースシステム：研究協力課
  - IP アドレス登録システム：情報連携基盤センター
- どのような不便さ・リスクがあるか？

- 各システムごとの認証データベースの利用
  → UserID や Password を忘れる
    → 統一認証 DB と Single Sign On 環境の構築

- 各システムごとに管理形態・レベルが異なる
  → 統一認証 DB を通じた情報漏洩のリスク
  → ユーザの多様性に起因するアクセス権限管理が複雑

- 何が必要なのか？
  - 統一認証基盤への安全なアクセス
  - 「標準化」されたアクセス権限管理

  → 各情報システムは固有の機能に専念
  （認証・認可からの開放）

研究の動機と背景 > マジメじゃない話

- 成績入力・履修登録システムの更新（２００５年２月稼動）
  - 内藤・梶田は「学務情報システム推進委員」であった
  - 「おい！ だいじょうぶかよぉ」と思うことが連発
    - どうやって認証をやるの？
    - セッション管理は大丈夫？
    - 成績入力ページへの学生のアクセスをどうやって防ぐの？
    - （この他にも山ほど...）
  - 梶田：「CAS ってのがあるから使ってみようか？」
    某社：「そこのところは名大でやってもらえるんですね？」
    内藤：「しゃぁないなぁ...」
    と私は抜けられない道に足を踏み入れることに...
    （一部脚色あり）

研究の動機と背景 > マジメじゃない話

- 名古屋大学ポータルの実験的な運用（２００４年頃）
  - 学務システムの「入り口」として本格運用開始を予定
  - 「学務システム」と「名古屋大学ポータル」を CAS を利用して SSO できることを目標にする
  - CAS の実験を開始すると問題点に気が付く
    - 誰でもどこからでも「認証がパスしたかどうか」の情報を得ることができる

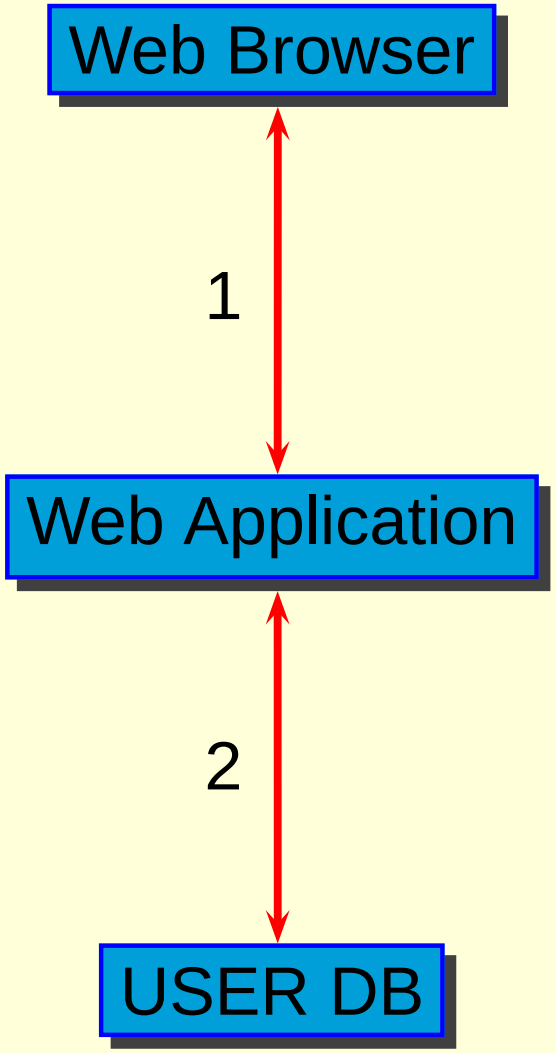
    → CAS を改造する必要が発生

    → もっと足抜けできないことになってしまい今に至る

Brief survey of SSO using CAS

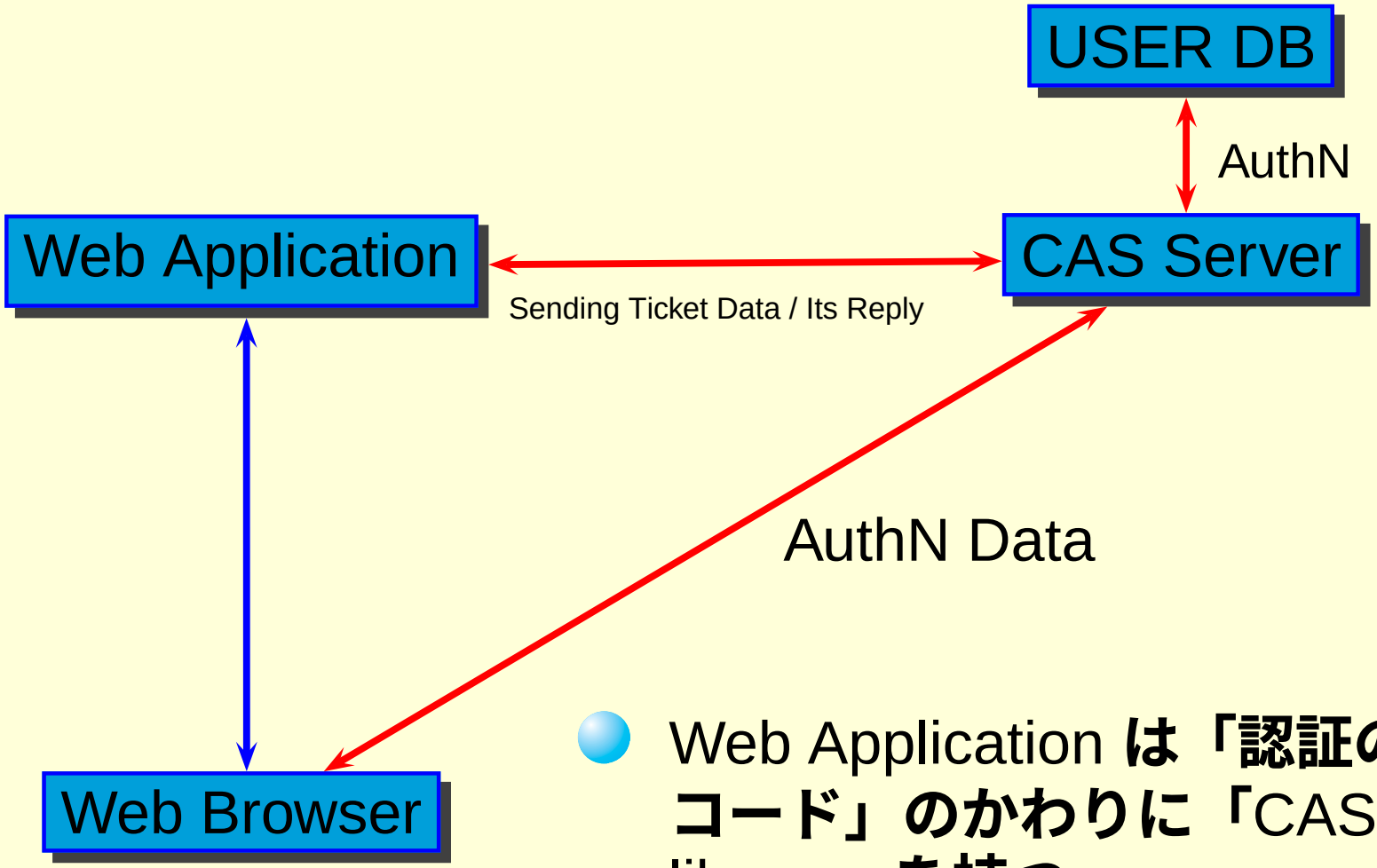
- CAS (Central Authentication Service)
  - Web Application **に対する** Single Sign On (SSO) **を構築**
  - Yale University, JA-SIG **によって** Open Source **として開発されている**
  - Cookie, http direction, JavaScript **などの標準的な機構だけで動作する**
  - **通信の暗号化には** SSL (https) **を利用**
  - **認証** DB **とは独立であり**, DB **の形式に依存しない**
  - **認証** DB **の安全性が飛躍的に向上**
  - Web Application **を** CAS **対応にすることが容易**

Brief ... using CAS > Usual Authentication



- Web Application **自身が認証のためのコードを持つ必要あり（認証** DB **の形式などに依存）**
- Web Application **が直接認証** DB **にアクセスする**
  - Web Application **はユーザのパスワードを受理する必要あり**

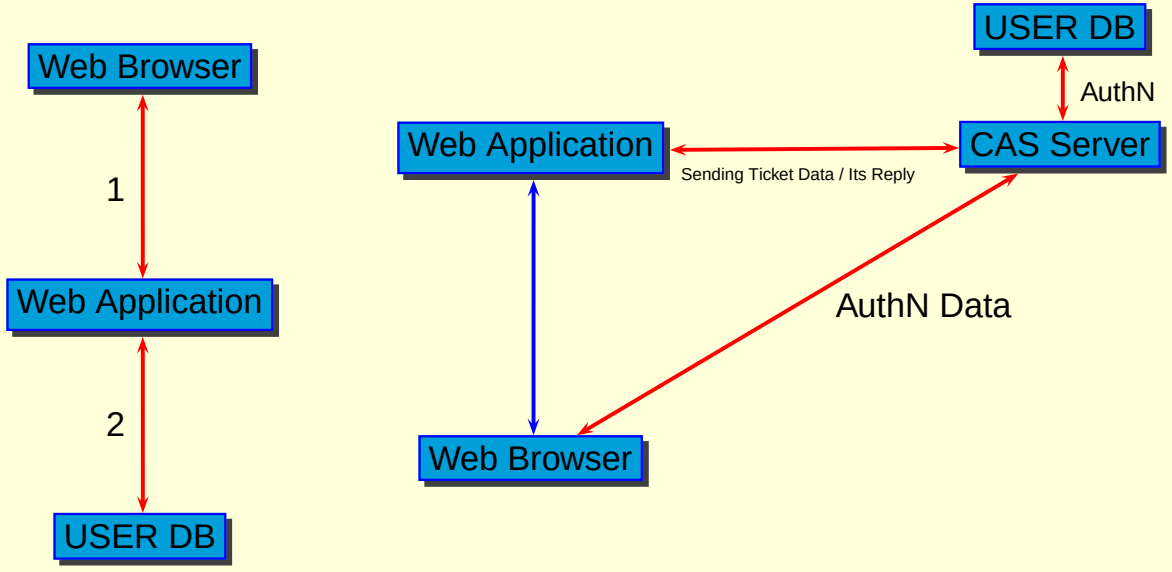
Brief ... using CAS > AuthN mechanisum of CAS



- Web Application **は「認証のためのコード」のかわりに「**CAS client library**」を持つ**
- Web Application **は直接は認証** DB **にアクセスしない**

Brief ... using CAS > 比較

- 「赤い矢印」で示した部分には「ユーザの情報」が流れる



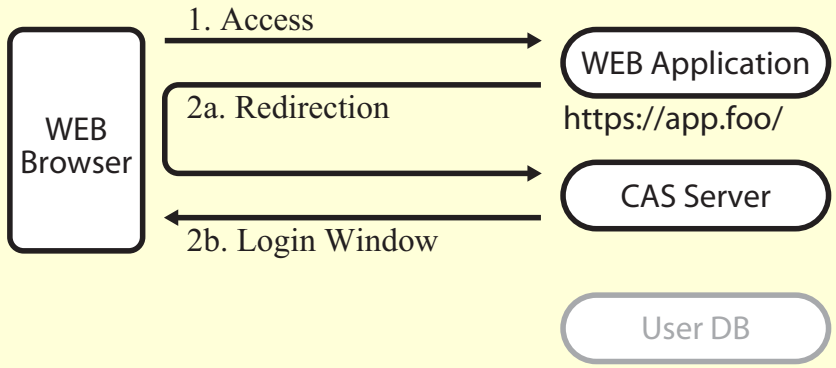
- App. の管理者が「SSL なんか必要ない！」とわめいたら...
  - 通常の認証形態では「そんなのダメ！」と言う以上のことはできない
  - CAS 認証なら App. 側が困るだけ

Brief ... using CAS > AuthN mechanisum of CAS

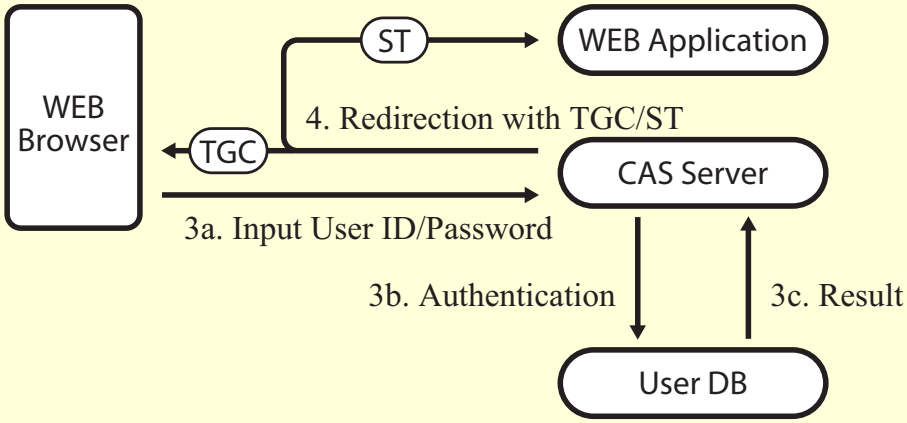
- Ticket Granting Cookie (TGC) – Cookie –
  - Browser **が有効な** TGC **を持つ** $\Longleftrightarrow$ **「認証されている」**
- Service Ticket (ST) – URL Parameter –
  - App. **にアクセスするための** One Time Ticket
  - App. **から** CAS Server **に有効な** ST **が提示される**
    $\Longrightarrow$ **「認証結果」（と「ユーザＩＤ」）が送信される**

Brief ... using CAS > AuthN mechanisum of CAS

- TGC を持たないブラウザが App. にアクセス
  → CAS Server が Login Window を提示

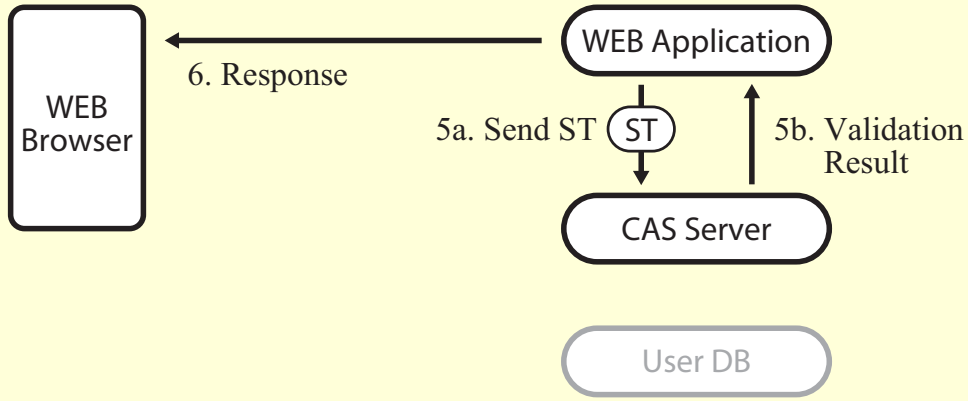


- 認証ＯＫ
  → CAS Server から TGC が送信＆ App. に ST を送信



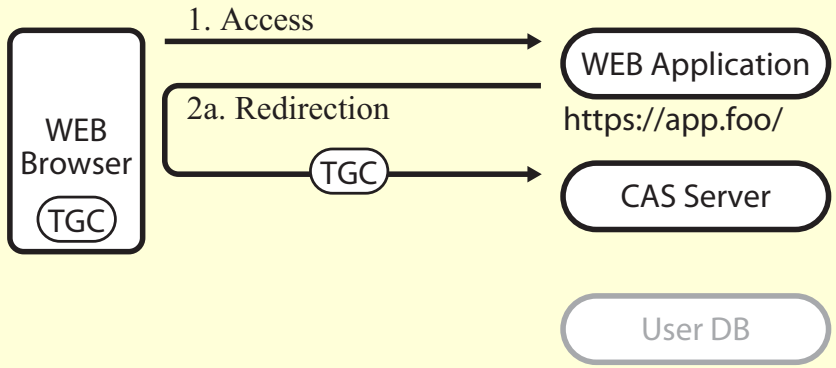
Brief ... using CAS > AuthN mechanisum of CAS

- App. は ST を CAS Server に送信
  → ST が有効ならばブラウザに情報を提示

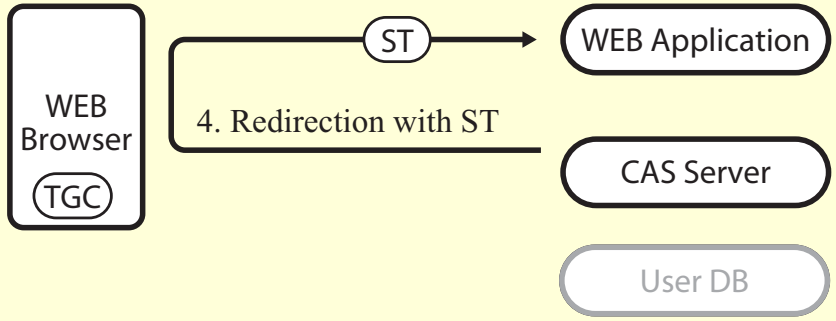


Brief ... using CAS > AuthN mechanisum of CAS

- TGC を持つブラウザが App. にアクセス
  → CAS Server への redirection が発生



- TGC が有効→ App. に ST を送信



- TGC が無効→ Login Window を提示

Brief ... using CAS > CAS の問題点

- 有効な ST さえ持っていれば, どの App. にもアクセス可能
  (current version では fix されている)
- CAS Server から App. に対して「User ID」のみが送信されていた
- POST method には対応できていなかった
- 国際化（日本語化）ができていなかった（あたりまえ？）

これらの問題を解決する（CAS^2 の開発）ことで
「Authorization 環境」が（自然に）出来上がった

Brief survey of Authorization Environment using CAS^2

- CAS^2 (Central Authentication and Authorization Service)
  - CAS **の** ST **を「アクセス権限管理」に利用**
  - App. **ごとに統一認証** DB **に基づく詳細なアクセス権限管理が可能**
  - CAS Server **から「**App. **に必要な個人情報」を送信可能**
  - CAS **対応の** Web Application **を** CAS^2 **対応にすることは** module **の入れ換えのみ**
  - **アクセス権限管理は**
    - FOR WHICH (URL of Web Application)
    - WHO (User)
    - WHEN (Access Time)
    - FROM WHERE (Client)

    に対して制御可能

Brief ... using CAS^2 > Access Control List

- CAS^2 **のアクセス権限管理は以下のような** CAS-ACL **に基づく**

```
dn: cn=entry1,ou=gakumu,ou=cas,o=nagoyaUniv
cas-allow: (&(uid=naito)(date>=20051010)
 (date<=20051110)(IP=133.6.130.0/24))
cas-service: https://app.*\.mynu\.jp/.+
cas-attributes: uid,mail
```

URL **が** `https://app.*\.mynu\.jp/.+` **にマッチしたとき**

- `uid` is `naito`
- Access time is between **2005/10/10** and **2005/11/10**
- Client IP: `133.6.130.0/24`

の時にのみアクセスが許可される

Brief ... using CAS^2 > Access Control List

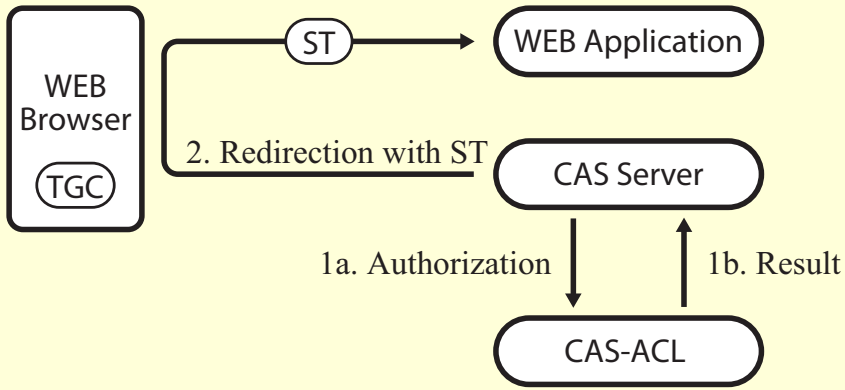
- CAS^2 のアクセス権限管理は以下のような CAS-ACL に基づく

```
dn: cn=entry1,ou=gakumu,ou=cas,o=nagoyaUniv
cas-allow: (&(uid=naito)(date>=20051010)
 (date<=20051110)(IP=133.6.130.0/24))
cas-service: https://app.*\.mynu\.jp/.+
cas-attributes: uid,mail
```

- アクセスが許可されたとき `cas-attributes` に示された属性情報のみが App. に送信される.
- 必要最小限の情報のみを App. に渡す設定が可能

Brief ... using CAS^2 > AuthZ mechanisum of CAS

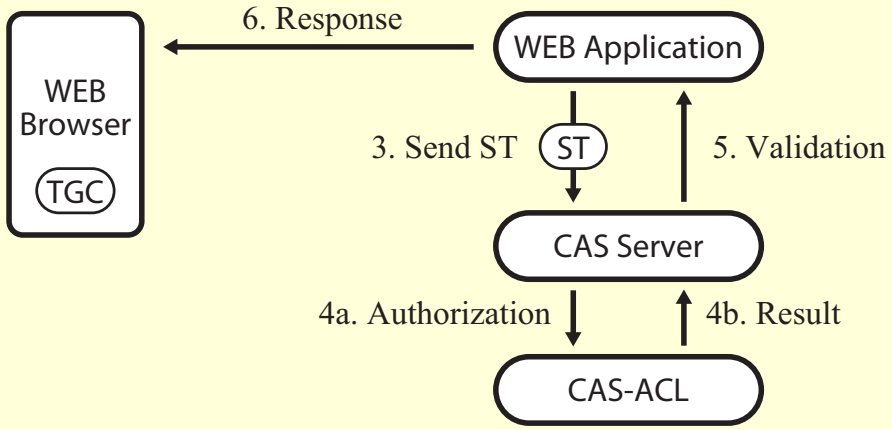
- ブラウザが App. にアクセス
  - App. の URL は redirection の パラメタとして渡される
  - App. の URL を CAS-ACL と照合
  - アクセスが許可されれば ST を発行



  - アクセスが許可されていないときには「Access Denied」のページを表示

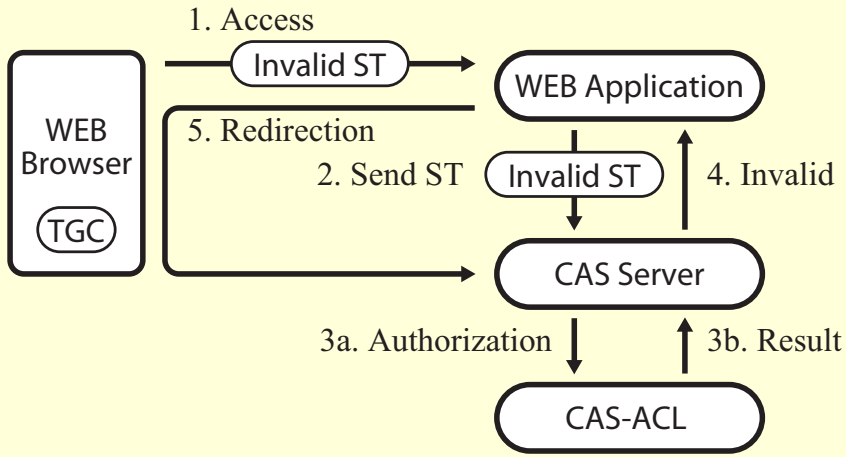
Brief ... using CAS^2 > AuthZ mechanisum of CAS

- App. は ST を CAS Server に送信
  - その際に App. の URL もパラメタとして渡される
  - 再度 ST を CAS-ACL と照合
    （ST の改竄による Man-in-Middle Attack を回避）
  - ST が有効ならば「アクセス許可」を App. に送信



Brief ... using CAS^2 > AuthZ mechanisum of CAS

- **アクセスが許可されていないとき, または** ST **が有効でないとき**
  →「Access Denied**」のページへの** redirection **を発生**

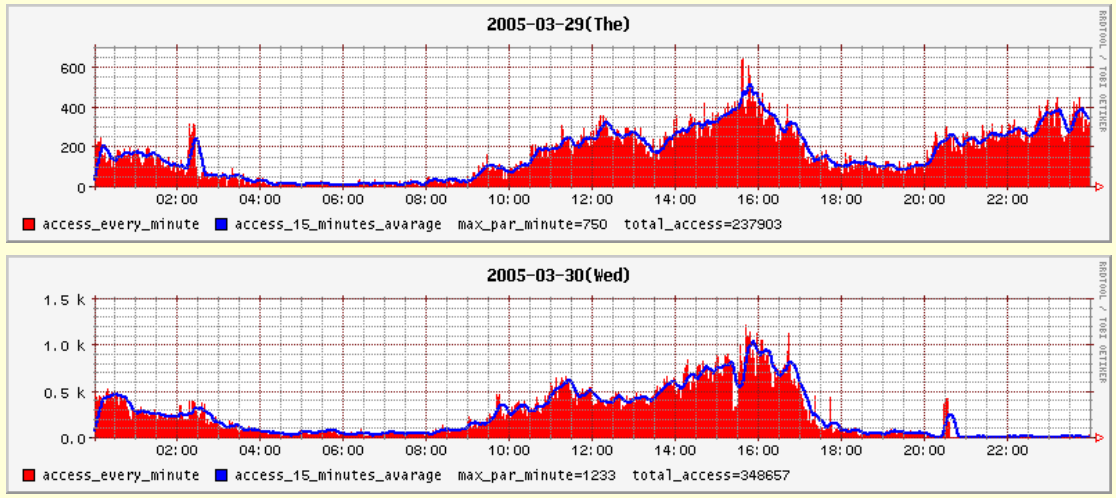


名古屋大学での運用実績

- ２００５年２月から運用開始
  - 「名古屋大学ポータル」＋「学務システム」＋「CAS^2」
  - 既存の情報システムの CAS^2 への移行
  - 新規情報システムは CAS^2 のみ
- CAS^2 を利用した情報システム
  - 名古屋大学ポータル
  - 学務システム
  - 研究者情報データベース
  - 法科大学院教育システム
  - 安否確認システム
  - （その他, 私の知らないもの... ）

名古屋大学での運用実績 > 実運用での負荷

- ２００５年３月履修登録時の CAS Server への負荷



- CAS Server の access log から解析
- 毎分 1000 回程度のアクセス
- 学務システムと組み合わせた負荷試験では, 毎分 3000 回程度のアクセス
  - この時は Oracle のアクセス限界に達して, 負荷試験を終了

名古屋大学での運用実績 > ID 切り替え

- 名古屋大学では，２００７年に「全学ＩＤ」から「名古屋大学ＩＤ」への切り替えを予定
  - 切り替えの理由はいろいろあります...
  - とりあえず，一定期間は両方のＩＤが共存
- このような場合でも CAS を使っていれば柔軟な対応が可能
- 今回の切り替えでは，認証 DB は依然として LDAP を使いますが....

名古屋大学での運用実績 > ID 切り替えの問題点

- もし認証 DB の形式を変更したら....
  - 各システムの「認証モジュール」の全面的な変更が発生
    → あまりに非現実的
  - CAS を使っていれば...
    CAS の認証 DB へのアクセスハンドラの置き換きかえ
- それでも「ＩＤが変わっちゃうんだから...」
  - CAS がアクセスする認証 DB を切り替える,
    または認証 DB へのアクセス方法を切り替える
  - App. 側の本質的な変更はない
  - CAS からの返却データのどの属性を見るかを切りかえる
- 当初我々が想定していなかった「予想外のオマケ」

名古屋大学での運用実績 > 問題点

- CAS-ACL を適切に記述できれば，「統一認証・認可基盤」が構築できる.
- CAS-ACL を適切に記述する方法は？
  - 情報システムの利用者の「Role Management」が必要
  - リソースへのアクセスポリシの明確化が必要
- 「だれ」が「どのリソース」に「いつ」「どこから」アクセスできるか？
  - 「だれ」が
    $\Leftarrow$ 「Identity Management」＋「Role Management」
  - 「どのリソース」に「いつ」「どこから」
    $\Leftarrow$ 「アクセスポリシ」

最近の実験的な試み

- 動機
  - よりセキュアな SSO 環境を構築したい
  - PKI (クライアント証明書) を有効に使えないか？
    （流行に乗り遅れたくない？）
  - いつでも「クライアント証明書が必要」なんて不便で使えないじゃん！
- 目標
  - 高いセキュリティが必要なアプリにはクライアント証明書を
  - そうでもないアプリは, セキュリティよりも利便性を

最近の実験的な試み > よくある話

- IC Card with PKI を導入すると....
  - → どんなシステムにアクセスするためにも IC Card を要求
  - → BBS みたいな light なシステムにアクセスするにも PKI ？
  - → IC Card Reader なんて, どこにでも転がっているわけじゃない
  - → だれも BBS にアクセスしなくなる
  - → 「２ちゃんねる」でボロクソ書かれる
- 各情報システムのレベルに沿った SSO/AuthZ 環境が構築できないか？

最近の実験的な試み > Example

次の３つの情報システムが CAS^2 で SSO 環境にあると仮定

- **成績入力システム**
  requirement : 可能な限り高いセキュリティ
- **履修登録システム**
  requirement : セキュリティは確保したいけど, 学生に取っても利便性も大事
- BBS
  requirement : まあ, セキュリティは気にしない. それよりも利便性を...

最近の実験的な試み > Example

次の “3-tiered security hierarchy” を定義

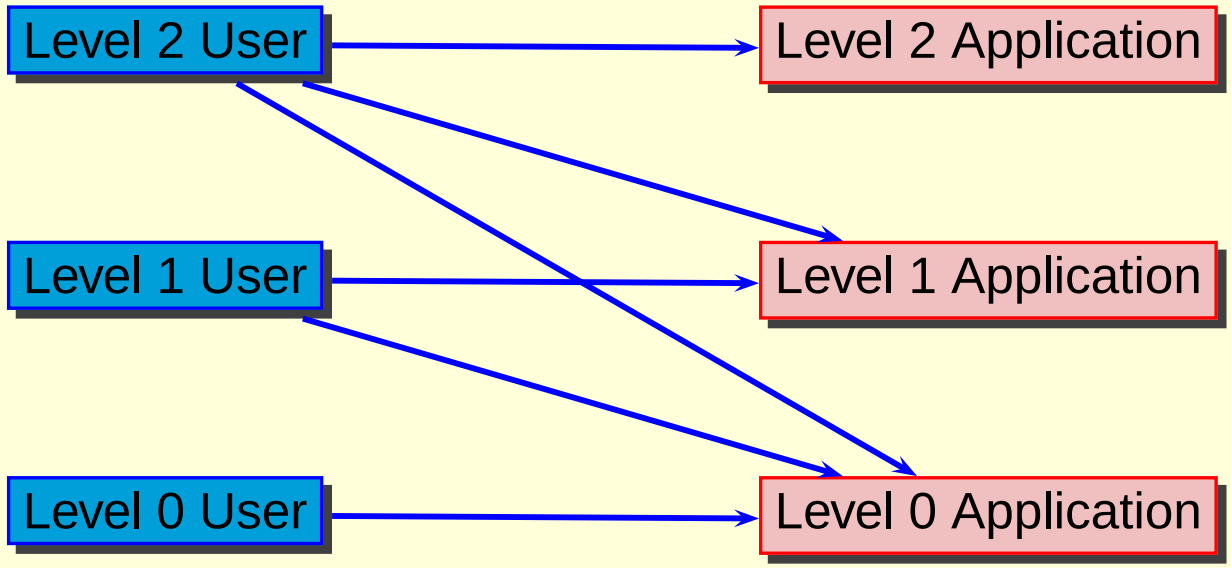
- Level 2 クライアント証明書がないとアクセスできない
- Level 1 Username/Password authentication でもアクセスＯＫ
- Level 0 携帯電話から Subscriber ID 認証でもアクセスＯＫ

最初の３つのアプリに “Level” を割り当てる

- 成績入力システム $\Longrightarrow$ Level 2
- 履修登録システム $\Longrightarrow$ Level 1
- BBS $\Longrightarrow$ Level 0

最近の実験的な試み > Mutiple-tiered secuirty hierarchy

- ユーザの認証に「hierarchy」の概念を導入
- 指定のレベル以上で認証されているユーザのみがアクセス可能
- このような制御が CAS^2 で可能か？



最近の実験的な試み > CAS^2 への secuirty hierarchy の導入

- CAS-ACL に “security level” を定義
  - これまでの CAS-ACL
    - FOR WHICH (URL of Web Application)
    - WHO (User)
    - WHEN (Access Time)
    - FROM WHERE (Client)
  - 追加するもの
    - HOW (Security Level)
- CAS^2 の認証メカニズムに “multiple-tiered AuthN sequence” を導入
- CAS^2 のアクセス権限管理メカニズムを修正

最近の ... > CAS^2 ... > security level in CAS-ACL

```
dn: cn=entry1,ou=gakumu,ou=cas,o=nagoyaUniv
cas-allow: (&(uid=naito)(date>=20051010)
 (date<=20051110)(IP=133.6.130.0/24))
cas-security-hierarchy: X509
cas-service: https://app.*\.mynu\.jp/.+
cas-attributes: uid,mail
```

URL が `https://app.*\.mynu\.jp/.+` にマッチしたとき

- 従来の ACL の照合にパスする
- ユーザは X509 (Level 2) 以上のレベルで認証されている

の時にのみアクセスが許可される

最近の ... > CAS^2 ... > Modify AuthN mechanism in CAS^2

- **原則的な方法**
  - **高レベルの認証方法から順次** fall down **する認証** sequence **を構築する**
  - **どの認証方法でパスしたかを** TGC DB **に記録する**
- **幸いなことに**...

  CAS (version 3) **は** “SpringFramework” **で記述されている**
  - **認証ルーチンを** “Multiple-tiered AuthN sequence” **を実現するように修正**
  - **認証ルーチンの** Dependancy Injection **として** “Multiple-tiered AuthN sequence” **を定義**

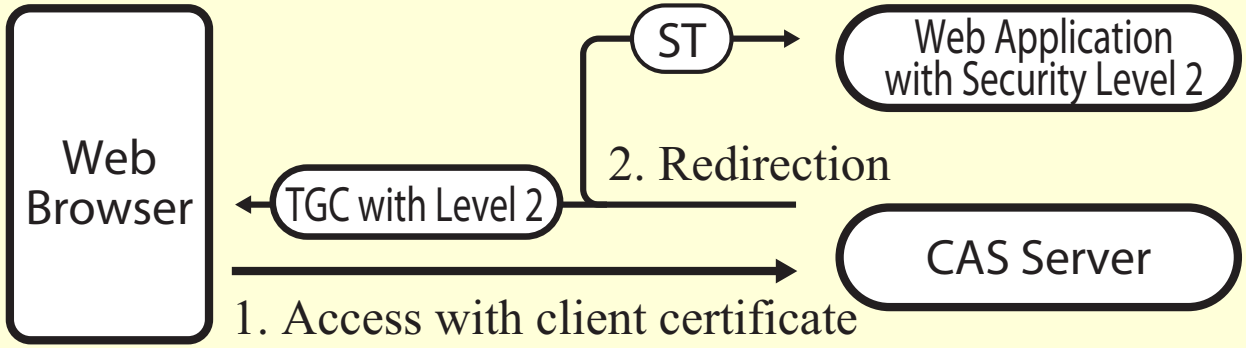
    →認証レベルの定義は容易に変更可能

Example の “3-tiered hierarchy” ならば...

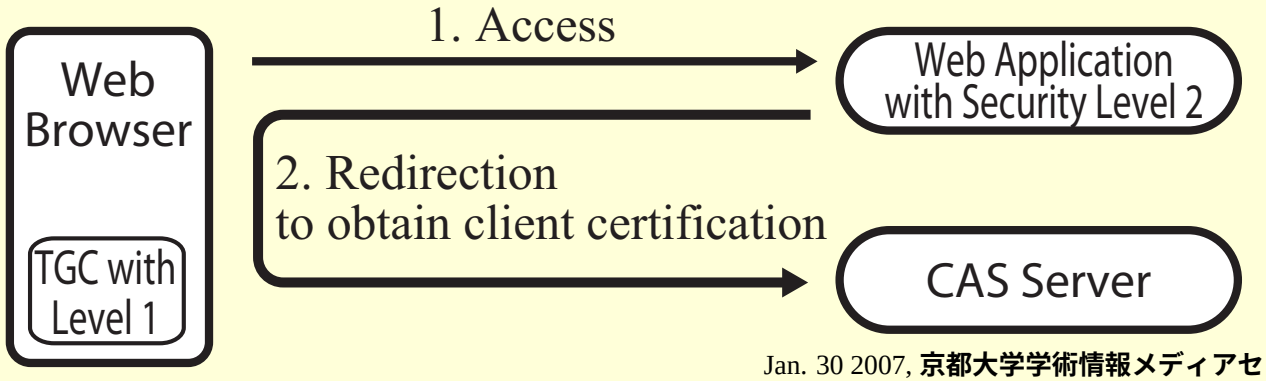
- それぞれのレベルに対応する認証ハンドラを bean として定義
  - bean class="X509CredentialsToPrincipalHandler"
    property name="loginLevel" value="X509"
  - bean class="BindLdapAuthenticationHandler"
    property name="loginLevel" value="PIN_UID"
  - bean class="SubscriberIdLdapAuthenticationHandler"
    property name="loginLevel" value="SUBSCRIBERID"
- “Login Level” を調べる bean に以下を定義

```
<list>
<value>SUBSCRIBERID</value>
<value>PIN_UID</value>
<value>X509</value>
</list>
```

- Case 1 : Level 2 Authentication
  - **ブラウザが「クライアント証明書」を提示して認証を受ける**
  - **ブラウザは「**Level 2**」の「**TGC**」を受理する**
  - Level 2 App. **にアクセス可能**



- Case 2 : **ブラウザが** Level 1 **として認証されている時に** Level 2 App. **にアクセスした場合**
  - **（有効な** ST **を持っていないはずなので）** Web App. **は** CAS Server **に** redirect
  - **ブラウザが「クライアント証明書」を持っていなければ,「**Level 2 App. **へのアクセス拒否」**
  - **ブラウザが「クライアント証明書」を持っていれば新規に 「**Level 2**」の「**TGC**」を発行**



Summary

- 名古屋大学での CAS^2 を利用した SSO/AuthZ 環境について解説した.
  - CAS^2 を利用することで比較的容易に SSO/AuthZ 環境を構築できる.
  - 実際の運用においても, 柔軟な対応が可能になる利点があった.
  - 現実には CAS-ACL を適切に記述することは面倒である.
- よりセキュアな認証環境を実現するための最近の実験的な試み
  - SSO/AuthZ 環境を, 利便性を保ちつつ, よりセキュアにするための試みを行っている
  - この機能を加えた CAS^2 を（今度こそ）近日中に Beta Version を公開予定

References

- **内藤, 梶田, 小尻, 平野, 間瀬,**
  大学における統一認証基盤としての CAS **とその拡張**
  情報処理学会論文誌, **47** (2006) 1127–1135.

- Naito, Kajita, Hirano, Mase,
  Multiple-tiered Security Hierarachy for Web Applications Using Central Authentication and Authorization Service,
  Proceeding of Middleware Workshop on IEEE International Symposium on Applications and the Internet (SAINT 2007), Hiroshima, JAPAN (2007).

Q and A