Architecture of Enterprise Applications III
Single Sign-On

Haopeng Chen

RE*liable,* ***IN****telligent and* ***S****calable Systems Group (****REINS****)*

Shanghai Jiao Tong University

Shanghai, China

e-mail: chen-hp@sjtu.edu.cn

Agenda

- Single Sign-On
  - Overview
  - Kerberos protocol
  - SAML
  - CAS

Single Sign-On - wikipedia

- **Single sign-on** (**SSO**) is a property of access control of multiple related, but independent software systems.
  - With this property a user logs in once and gains access to all systems without being prompted to log in again at each of them.
  - Conversely, **Single sign-off** is the property whereby a single action of signing out terminates access to multiple software systems.

- As different applications and resources support different authentication mechanisms,
  - single sign-on has to internally translate to and store different credentials compared to what is used for initial authentication.

Single Sign-On - wikipedia

REliable, INtelligent & Scalable Systems

- **Benefits** include:
  - Reduces phishing success, because users are not trained to enter password everywhere without thinking.
  - Reducing password fatigue from different user name and password combinations
  - Reducing time spent re-entering passwords for the same identity
  - Reducing IT costs due to lower number of IT help desk calls about passwords
  - Security on all levels of entry/exit/access to systems without the inconvenience of re-prompting users
  - Centralized reporting for compliance adherence.

Common Single Sign-on

- Kerberos based
  - MIT Kerberos protocol
- Smart card based
  - Initial sign-on prompts the user for the smart card.
  - Additional software applications also use the smart card, without prompting the user to re-enter credentials.
  - Smart card-based single sign-on can either use certificates or passwords stored on the smart card.
- OTP token
  - Also referred to as one-time password token.
- Security Assertion Markup Language
  - Security Assertion Markup Language (SAML) is an XML-based solution for exchanging user security information between an enterprise and a service provider.

Kerberos

- Kerberos
  - is a computer network authentication protocol which works on the basis of "tickets" to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

- MIT developed Kerberos to protect network services provided by Project Athena.
  - The protocol was named after the character *Kerberos* (or *Cerberus*) from Greek mythology which was a monstrous three-headed guard dog of Hades.
  - Recent release: 08 Aug 2012 - krb5-1.10.3

Kerberos

- **User Client-based Logon**
  - A user enters a username and password on the client machines.
  - The client performs a one-way function (hash usually) on the entered password, and this becomes the secret key of the client/user.

- **Client Authentication**
  - The client sends a clear text message of the user ID to the AS requesting services on behalf of the user. (Note: Neither the secret key nor the password is sent to the AS.)
  - The AS generates the secret key by hashing the password of the user found at the database (e.g. Active Directory in Windows Server).

Kerberos

- The AS checks to see if the client is in its database. If it is, the AS sends back the following two messages to the client:
  - Message A: *Client/TGS Session Key* encrypted using the secret key of the client/user.
  - Message B: *Ticket-Granting-Ticket* (which includes the client ID, client network address, ticket validity period, and the *client/TGS session key*) encrypted using the secret key of the TGS.

Kerberos

- Once the client receives messages A and B, it attempts to decrypt message A with the secret key generated from the password entered by the user.
  - If the user entered password does not match the password in the AS database, the client's secret key will be different and thus unable to decrypt message A.
  - With a valid password and secret key the client decrypts message A to obtain the *Client/TGS Session Key*. This session key is used for further communications with the TGS. (Note: The client cannot decrypt Message B, as it is encrypted using TGS's secret key.)
  - At this point, the client has enough information to authenticate itself to the TGS.

Kerberos

- **Client Service Authorization**
- When requesting services, the client sends the following two messages to the TGS:
  - Message C: Composed of the TGT from message B and the ID of the requested service.
  - Message D: Authenticator (which is composed of the client ID and the timestamp), encrypted using the *Client/TGS Session Key*.

Kerberos

- Upon receiving messages C and D, the TGS retrieves message B out of message C. It decrypts message B using the TGS secret key. This gives it the "client/TGS session key". Using this key, the TGS decrypts message D (Authenticator) and sends the following two messages to the client:
  - Message E: *Client-to-server ticket* (which includes the client ID, client network address, validity period and *Client/Server Session Key*) encrypted using the service's secret key.
  - Message F: *Client/Server Session Key* encrypted with the *Client/TGS Session Key*.

Kerberos

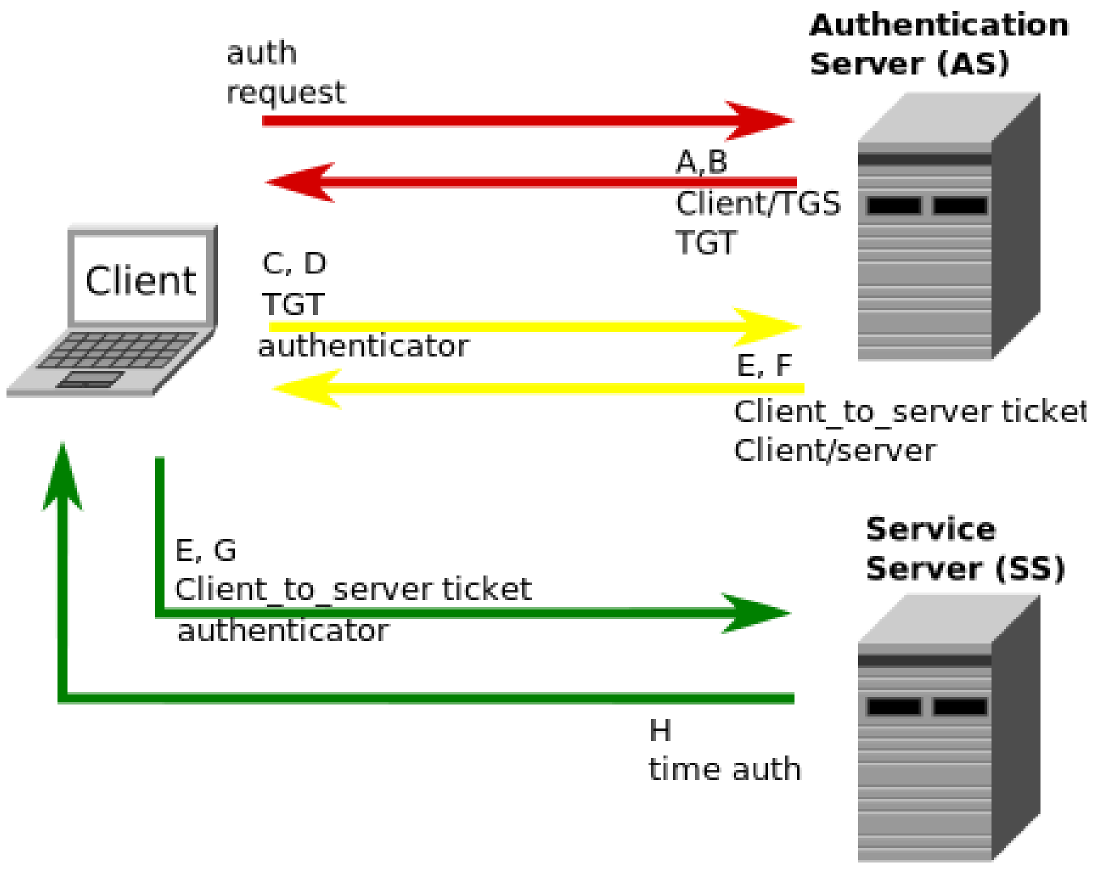
- **Client Service Request**
- Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the SS. The client connects to the SS and sends the following two messages:
  - Message E from the previous step (the *client-to-server ticket*, encrypted using service's secret key).
  - Message G: a new Authenticator, which includes the client ID, timestamp and is encrypted using *Client/Server Session Key*.

Kerberos

- The SS decrypts the ticket using its own secret key to retrieve the *Client/Server Session Key*. Using the sessions key, SS decrypts the Authenticator and sends the following message to the client to confirm its true identity and willingness to serve the client:
  - Message H: the timestamp found in client's Authenticator plus 1, encrypted using the *Client/Server Session Key*.

- The client decrypts the confirmation using the *Client/Server Session Key* and checks whether the timestamp is correctly updated. If so, then the client can trust the server and can start issuing service requests to the server.
- The server provides the requested services to the client.

Kerberos


auth
request
Authentication
Server (AS)
A,B
Client/TGS
TGT
Client
C, D
TGT
authenticator
E, F
Client_to_server ticket
Client/server
E, G
Client_to_server ticket
authenticator
Service
Server (SS)
H
time auth

Kerberos

- Drawbacks and Limitations
  - Single point of failure.
  - Kerberos has strict time requirements, which means the clocks of the involved hosts must be synchronized within configured limits.
  - The administration protocol is not standardized and differs between server implementations.
  - Since all authentication is controlled by a centralized KDC, compromise of this authentication infrastructure will allow an attacker to impersonate any user.
  - Each network service which requires a different host name will need its own set of Kerberos keys. This complicates virtual hosting and clusters.

Kerberos

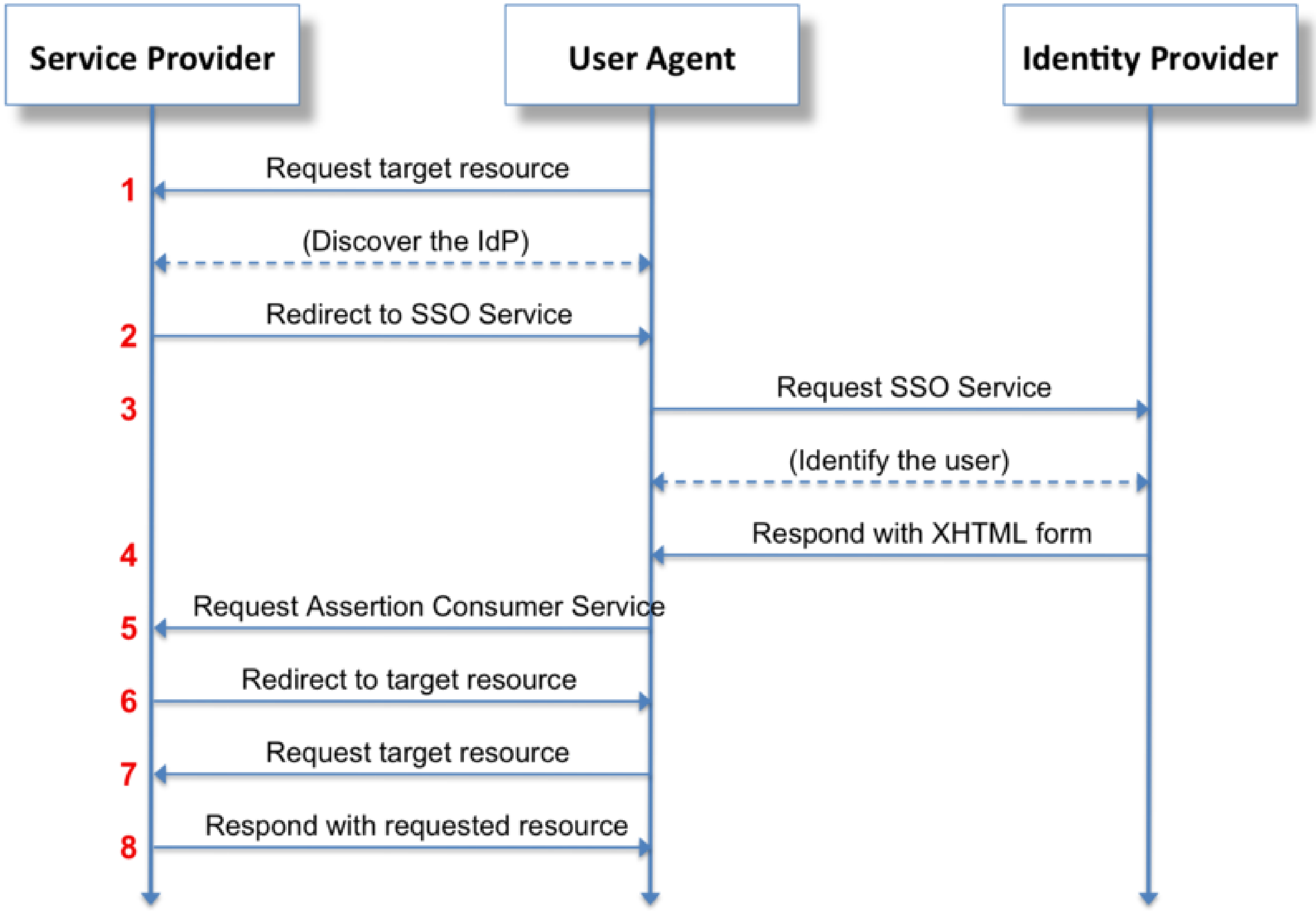
- Kerberos based Single Sign-On
  - Initial sign-on prompts the user for credentials, and gets a Kerberos ticket-granting ticket (TGT).
  - Additional software applications requiring authentication, such as email clients, wikis, revision control systems, etc., use the ticket-granting ticket to acquire service tickets, proving the user's identity to the mail server / wiki server / etc. without prompting the user to re-enter credentials.
  - Windows environment - Windows login fetches TGT. Active Directory-aware applications fetch service tickets, so user is not prompted to re-authenticate.
  - Unix/Linux environment - Login via Kerberos PAM modules fetches TGT. Kerberized client applications such as Evolution, Firefox, and SVN use service tickets, so user is not prompted to re-authenticate.

SAML

- Security Assertion Markup Language is an XML-based open standard data format
  - for exchanging authentication and authorization data between parties,
  - in particular, between an identity provider and a service provider.
- SAML is a product of the OASIS Security Services Technical Committee.
  - SAML dates from 2001.

SAML

- The primary SAML use case is called *Web Browser Single Sign-On (SSO)*.

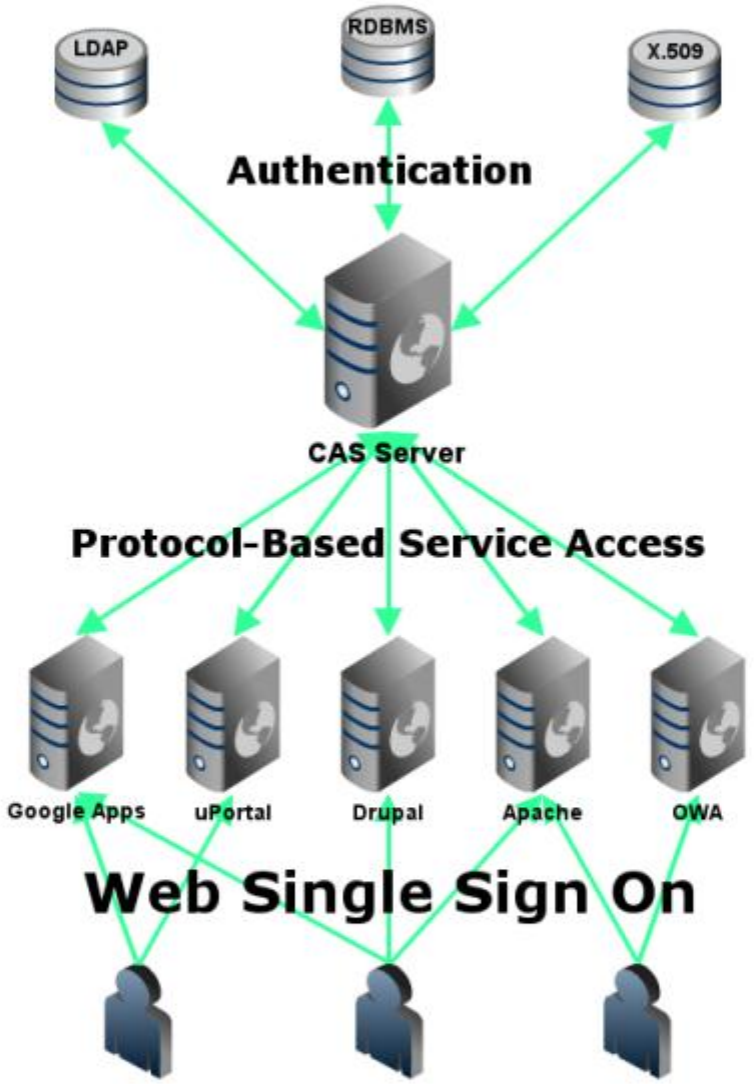

Service Provider
User Agent
Identity Provider
1 Request target resource
(Discover the IdP)
2 Redirect to SSO Service
3 Request SSO Service
(Identify the user)
4 Respond with XHTML form
5 Request Assertion Consumer Service
6 Redirect to target resource
7 Request target resource
8 Respond with requested resource

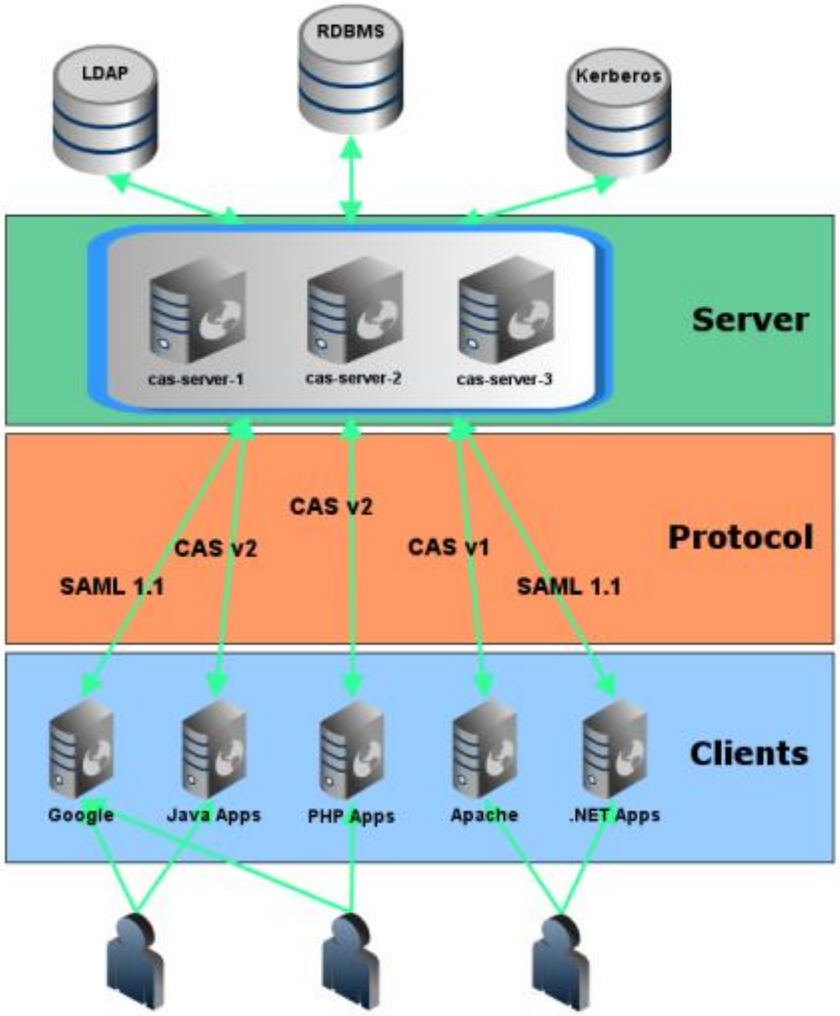
CAS

- Central Authentication Service provides enterprise single sign-on service:
  - An open and well-documented protocol
  - An open-source Java server component
  - A library of clients for Java, .Net, PHP, Perl, Apache, uPortal, and others
  - Integrates with uPortal, BlueSocket, TikiWiki, Mule, Liferay, Moodle and others
  - Community documentation and implementation support
  - An extensive community of adopters

- Latest version: 3.5.0 final

CAS


LDAP
RDBMS
X.509
Authentication
CAS Server
Protocol-Based Service Access
Google Apps
uPortal
Drupal
Apache
OWA
Web Single Sign On


LDAP
RDBMS
Kerberos
cas-server-1
cas-server-2
cas-server-3
Server
CAS v2
CAS v2
CAS v1
SAML 1.1
SAML 1.1
Protocol
Google
Java Apps
PHP Apps
Apache
.NET Apps
Clients

List of URIs to access CAS.

- /login
  - Parameters: service, renew, gateway, warn
- /logout
  - Parameters: url
- /validate
  - Parameters: service, ticket, renew
- /serviceValidate
  - Parameters: service, ticket, pgtUrl, renew
- /proxy
  - Parameters: pgt, targetService
- /proxyValidate
  - Parameters: service, ticket, pgtUrl, renew

Tickets generated by CAS

- Ticket-granting Ticket
- Service Ticket
- Proxy Ticket
- Proxy-granting Ticket
- Proxy-granting Ticket IOU
- Login Ticket

Ticket-granting Ticket

- Ticket granting ticket will be generated when the /login url is passed to CAS server and the credentials provided are successfully authenticated.
- A TGT is the main access into the CAS service layer.
- TGT is an opaque string that contains secure random data and must begin with "TGT-".
- TGT will be added to an HTTP cookie upon the establishment of single sign-on and will be checked further when different applications are accessed

Service Ticket

- The service ticket (ST) will be generated when the CAS url contains service parameter and the credentials passed are successfully authenticated.
- Service ticket is an opaque string that is used by client as a credential to obtain access to a service.
- Service ticket must begin with “ST-“

Proxy Ticket

- In CAS, proxy is a service that wants to access other services on behalf of a particular user.
- Proxy tickets (PT) are generated from CAS upon a services' presentation of a valid Proxy granting Ticket (PGT), and a service identifier for the back-end service to which it is connecting.
- PT are only valid for the service identifier specified to /proxy url when they were generated.
- Proxy tickets should begin with the characters, "PT-".

Proxy-granting Ticket

- Proxy-granting tickets are obtained from CAS upon validation of a service ticket or a proxy ticket. If a service wishes to proxy a client's authentication to a back-end service, it must acquire a proxy-granting ticket.
- Acquisition of this ticket is handled through a proxy callback URL. This URL will uniquely and securely identify the back-end service that is proxying the client's authentication.
- The back-end service can then decide whether or not to accept the credentials based on the back-end service's identifying callback URL.

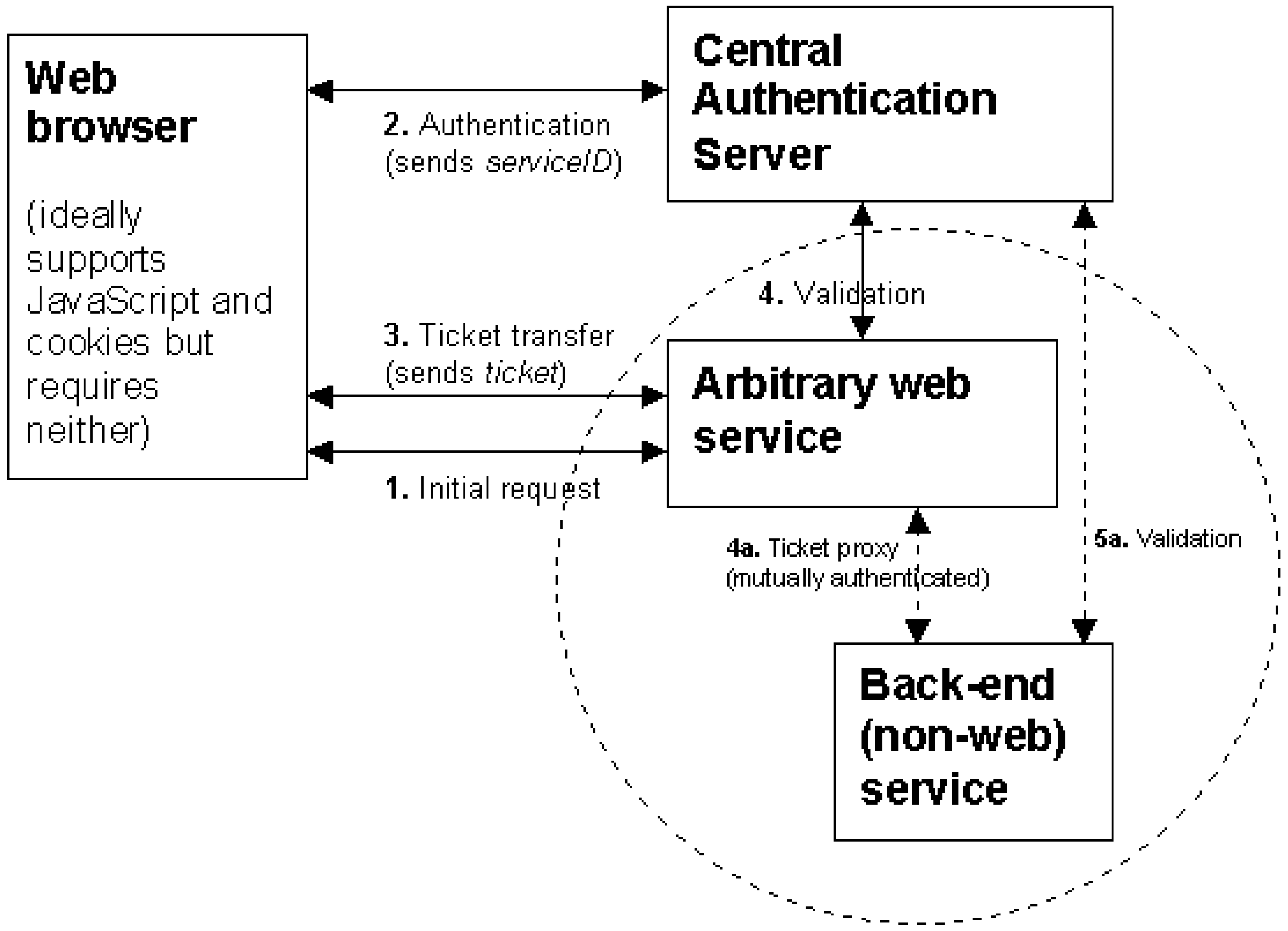
Proxy-granting Ticket IOU

- A proxy-granting ticket IOU is an opaque string that is placed in the response provided by /serviceValidate or /proxyValidate used to correlate a service ticket or proxy ticket validation with a particular proxy-granting ticket.
- Proxy-granting ticket IOUs SHOULD begin with the characters, "PGTIOU-".

Login Ticket

- A login ticket is a string that is generated by /login as a credential requestor and passed to /login as a credential acceptor for username/password authentication.
- Its purpose is to prevent the replaying of credentials due to bugs in web browsers.
- Login tickets SHOULD begin with the characters, "LT-".

CAS Architecture


Web browser
(ideally supports JavaScript and cookies but requires neither)
Central Authentication Server
Arbitrary web service
Back-end (non-web) service
2. Authentication (sends serviceID)
3. Ticket transfer (sends ticket)
1. Initial request
4. Validation
4a. Ticket proxy (mutually authenticated)
5a. Validation

References

- Single Sign On, http://en.wikipedia.org/wiki/Single_sign-on
- Kerberos: The Network Authentication Protocol, http://web.mit.edu/kerberos/
- Kerberos(protocol), http://en.wikipedia.org/wiki/Kerberos_(protocol)
- SAML, http://en.wikipedia.org/wiki/Security_Assertion_Markup_Language
- CAS, http://www.jasig.org/cas
- Jasig CAS Documentation, http://www.unicon.net/files/cas-server-3-4-11-snapshot-manual.pdf
- CAS ppt, http://www.jusfortechies.com/java/cas/ppt.php

Thank You!