syt-gk931-cas

`author: akramreiter`

Protocol

Setting up the server

Clone or download the overlay template.

Run `gradlew copyCasConfiguration` to copy the config files.

Setup a keystore and certificate with `gradlew createKeystore`.

This repository per default uses configuration from the directory `<working_partition>:/etc`. The quick workaround I used to resolve this was creating a symbolic link from `D:/etc` to the overlay template's `etc/` directory.

`gradlew run` starts the server.

The service is hosted on https://localhost:8443/cas/login.

Logging in with the default user `casuser` and the password `Mellon` should work on this page.

Allow apps to access the service

To allow other apps to use the CAS login service, it needs to be explicitly allowed in the service registry.

In this case, it'll be done via the JSON service registry. The necessary package is already listed under dependencies in `build.gradle`, but commented out. Remove the `//` to use it.

```
dependencies {
    // Other CAS dependencies/modules may be listed here...
    compile "org.apereo.cas:cas-server-support-json-service-
registry:${casServerVersion}"
}
```

The path to the location of the services needs to be configured and `initFromJson` needs to be enabled in `/etc/cas/config/cas.properties`. In this example, the services are located at `/etc/cas/services`.

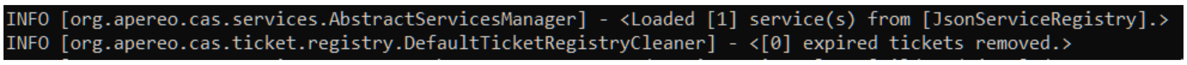
```
cas.serviceRegistry.initFromJson:true
cas.serviceRegistry.json.location:file:/etc/cas/services
```

Create a new file in `etc/cas/services`. The one I used is called `wildcard.json` to reflect that it's a wildcard that allows access universally. This is only for demo purposes and **shouldn't under any circumstances be used in a production environment**.

```
{
  "@class" :              "org.apereo.cas.services.RegexRegisteredService",
  "serviceId" :           "^(https|http)://.*",
  "name" :                "Wildcard",
  "id" :                  493,
  "evaluationOrder" :     99
}
```

The `id` shouldn't overlap with the ids of other services and `evaluationOrder` sets the order in which services are evaluated, as the name implies.

After restarting the server, it should show that 1 service is loaded



```
INFO [org.apereo.cas.services.AbstractServicesManager] - <Loaded [1] service(s) from [JsonServiceRegistry].>
INFO [org.apereo.cas.ticket.registry.DefaultTicketRegistryCleaner] - <[0] expired tickets removed.>
```

Client

This webapp was used as the basis for the client. To connect to the local CAS server, the `web.xml` in `src/main/webapp/WEB-INF` had to be edited. In accordance to the instructions in its readme, all instances of `https://mmoayyed.unicon.net:8443` were replaced with `https://localhost:8443` since both the webapp and the server run on the same system.

The webapp needs a Java keystore with the password `changeit` in `ètc/`, which can be created with `keytool -genkey -keyalg RSA -alias localhost -keystore thekeystore -validity 360`.

If there is no trust store, the webapp throws security errors when receiving a response from the CAS server. To prevent this, a trust store needs to be created from the server's certificate and moved to the webapp's `etc/jetty` directory.

```
keytool -export -alias cas -file cert -keystore thekeystore
keytool -importcert -file cert -keystore thetruststore -alias cas
```

Additionally, the trust store needs to be recognized, which was done via additional Jvm args in the `pom.xml` of the webapp.

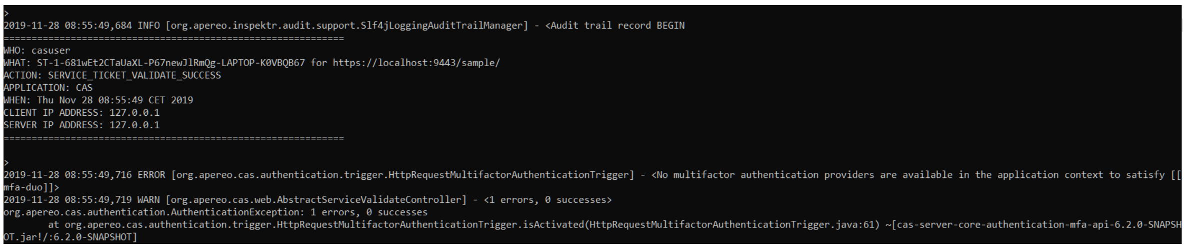
```
<plugin>
                <groupId>org.eclipse.jetty</groupId>
                <artifactId>jetty-maven-plugin</artifactId>
                <version>9.3.6.v20151106</version>
                <configuration>

<jettyXml>${basedir}/etc/jetty/jetty.xml,${basedir}/etc/jetty/jetty-
ssl.xml,${basedir}/etc/jetty/jetty-https.xml</jettyXml>
                    <systemProperties>
                        <systemProperty>
                            <name>org.eclipse.jetty.annotations.maxWait</name>
                            <value>300</value>
                        </systemProperty>
                    </systemProperties>
                    <webApp>
                        <contextPath>/sample</contextPath>

<overrideDescriptor>${basedir}/etc/jetty/web.xml</overrideDescriptor>
                    </webApp>
                    <jvmArgs>-Djavax.net.ssl.trustStore=etc/jetty/thetruststore
-Djavax.net.ssl.trustStorePassword=changeit -Dmaven.wagon.http.ssl.insecure=true
-Dmaven.wagon.http.ssl.allowall=true  -Djavax.net.debug=ssl:handshake -Xdebug -
Xrunjdwp:transport=dt_socket,address=5002,server=y,suspend=n</jvmArgs>
                </configuration>
            </plugin>
```

Running the webapp is now possible via

```
mvn clean package jetty:run-forked
```

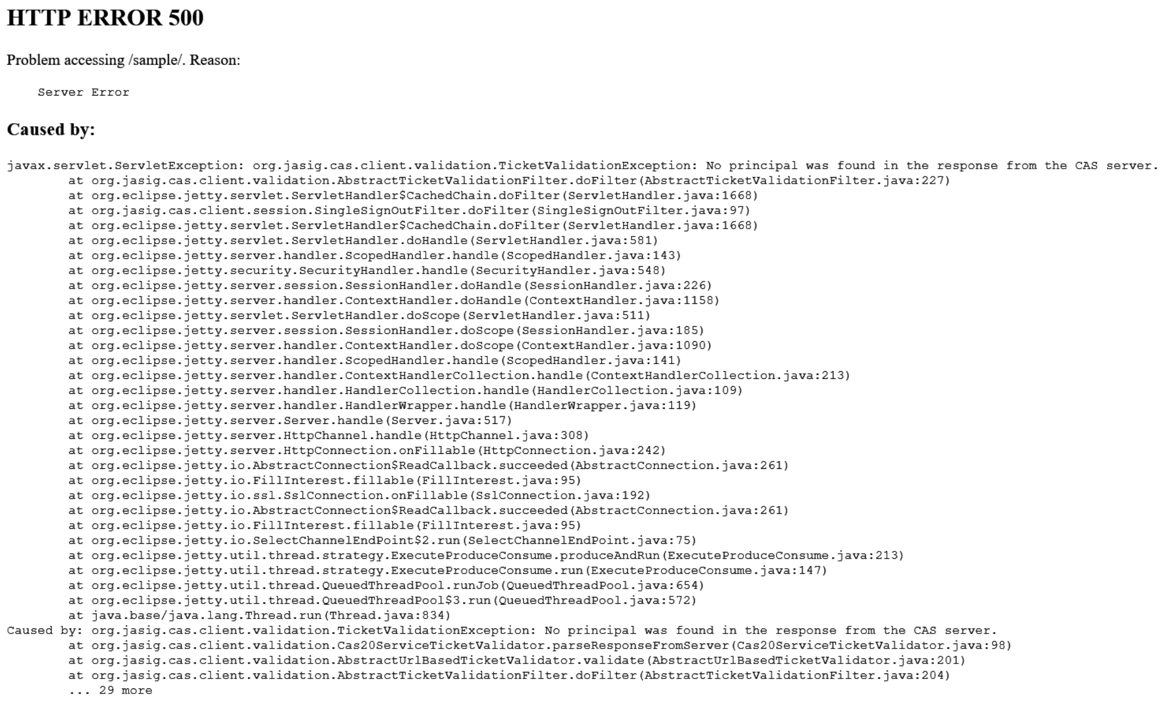
While logging in directly on the CAS server works fine, trying to do so gave me the following error:

Server



```
>
2019-11-28 08:55:49,684 INFO [org.apereo.inspektr.audit.support.Slf4jLoggingAuditTrailManager] - <Audit trail record BEGIN
=============================================================
WHO: casuser
WHAT: ST-1-681wEt2CTaUaXL-P67newJlRmQg-LAPTOP-K0VBQB67 for https://localhost:9443/sample/
ACTION: SERVICE_TICKET_VALIDATE_SUCCESS
APPLICATION: CAS
WHEN: Thu Nov 28 08:55:49 CET 2019
CLIENT IP ADDRESS: 127.0.0.1
SERVER IP ADDRESS: 127.0.0.1
=============================================================


>
2019-11-28 08:55:49,716 ERROR [org.apereo.cas.authentication.trigger.HttpRequestMultifactorAuthenticationTrigger] - <No multifactor authentication providers are available in the application context to satisfy [[
mfa-duo]]>
2019-11-28 08:55:49,719 WARN [org.apereo.cas.web.AbstractServiceValidateController] - <1 errors, 0 successes>
org.apereo.cas.authentication.AuthenticationException: 1 errors, 0 successes
        at org.apereo.cas.authentication.trigger.HttpRequestMultifactorAuthenticationTrigger.isActivated(HttpRequestMultifactorAuthenticationTrigger.java:61) ~[cas-server-core-authentication-mfa-api-6.2.0-SNAPSH
OT.jar!/:6.2.0-SNAPSHOT]
```

Client



HTTP ERROR 500

Problem accessing /sample/. Reason:

```
    Server Error
```

Caused by:

```
javax.servlet.ServletException: org.jasig.cas.client.validation.TicketValidationException: No principal was found in the response from the CAS server.
	at org.jasig.cas.client.validation.AbstractTicketValidationFilter.doFilter(AbstractTicketValidationFilter.java:227)
	at org.eclipse.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1668)
	at org.jasig.cas.client.session.SingleSignOutFilter.doFilter(SingleSignOutFilter.java:97)
	at org.eclipse.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1668)
	at org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:581)
	at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:143)
	at org.eclipse.jetty.security.SecurityHandler.handle(SecurityHandler.java:548)
	at org.eclipse.jetty.server.session.SessionHandler.doHandle(SessionHandler.java:226)
	at org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.java:1158)
	at org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandler.java:511)
	at org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:185)
	at org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:1090)
	at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:141)
	at org.eclipse.jetty.server.handler.ContextHandlerCollection.handle(ContextHandlerCollection.java:213)
	at org.eclipse.jetty.server.handler.HandlerCollection.handle(HandlerCollection.java:109)
	at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:119)
	at org.eclipse.jetty.server.Server.handle(Server.java:517)
	at org.eclipse.jetty.server.HttpChannel.handle(HttpChannel.java:308)
	at org.eclipse.jetty.server.HttpConnection.onFillable(HttpConnection.java:242)
	at org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:261)
	at org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:95)
	at org.eclipse.jetty.io.ssl.SslConnection.onFillable(SslConnection.java:192)
	at org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:261)
	at org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:95)
	at org.eclipse.jetty.io.SelectChannelEndPoint$2.run(SelectChannelEndPoint.java:75)
	at org.eclipse.jetty.util.thread.strategy.ExecuteProduceConsume.produceAndRun(ExecuteProduceConsume.java:213)
	at org.eclipse.jetty.util.thread.strategy.ExecuteProduceConsume.run(ExecuteProduceConsume.java:147)
	at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:654)
	at org.eclipse.jetty.util.thread.QueuedThreadPool$3.run(QueuedThreadPool.java:572)
	at java.base/java.lang.Thread.run(Thread.java:834)
Caused by: org.jasig.cas.client.validation.TicketValidationException: No principal was found in the response from the CAS server.
	at org.jasig.cas.client.validation.Cas20ServiceTicketValidator.parseResponseFromServer(Cas20ServiceTicketValidator.java:98)
	at org.jasig.cas.client.validation.AbstractUrlBasedTicketValidator.validate(AbstractUrlBasedTicketValidator.java:201)
	at org.jasig.cas.client.validation.AbstractTicketValidationFilter.doFilter(AbstractTicketValidationFilter.java:204)
	... 29 more
```

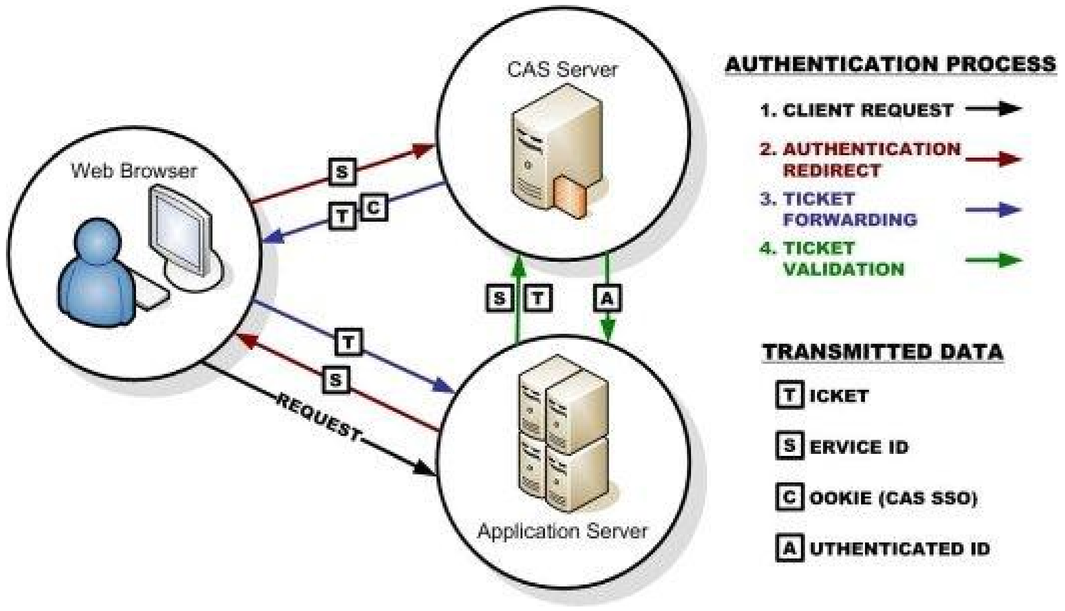
I spent roughly 7 hours working on this task, at least 2 of which were wasted trying to troubleshoot this single error. I wasn't able to find a solution and I have to stop here.

Fragestellungen

Was bedeutet Single Sign On?

SSO bedeutet, dass durch das Einloggen an einem Server der Zugriff auf mehrere Services möglich ist. Dies wird durch einen Central Authentication Server (CAS) und Session-cookies ermöglicht.

Zeigen Sie in einem Diagramm die Ablaeufe von CAS Client, CAS Server, Service und User.



Was ist ein TGT und beschreiben Sie die Funktionsweise eine TGT?

Ein TGT ist effektiv eine Datei, die IP-Adresse des Clients, Gültigkeitsdauer und einen vom Ticket Granting Server (TGS) generierten Key. Der TGS hat 2 wichtige Funktionen:

- auf Ticket Requests von Clients reagieren und TGTs ausstellen
- Authentifizierung mittels TGTs ermöglichen und Service Tickets an den Client übermitteln

Welche Features/Merkmale bietet der Apereo CAS Server an?

- Spring Boot Java Serverkomponente
- Pluggable Authentication (Einbindung von LDAP, MongoDb, etc für auth)
- Support für verschiedene Auth-Protokolle (Oauth2, CAS, SAML, etc)
- Multifactor Authentication (z.B. Google Authenticator)
- Delegated Authentication via external provider (Login via Facebook, Twitter, etc)
- Eingebauter Support für
    - Passwortmanagement
    - Benachrichtigungen
    - Terms of Use
    - Surrogate Authentication
- Registrieren von Client-Anwendungen
- Cross-platform support für Clients

Beschreiben Sie den CAS Server einer CAS Architektur?

Ist ein Server, der für die Authentifizierung für 1 oder mehrere Client Services zuständig ist. Hat 3 Hauptfunktionen für die Interaktion mit Usern und Client Services

- Stellt Tickets an User aus, die einen Client Web Service nutzen wollen, wenn sie sich einloggen (via Login-URL)
- Validiert Tickets, die Client Web Services erhalten (via Validation-URL)
- Invalidiert Tickets, wenn sich User ausloggen

Beschreiben Sie den CAS Client einer CAS Architektur?

Ist ein Webservice, das nur von vom CAS Server authentifizierten Usern genutzt werden kann. Nicht eingeloggte User werden an die Login-URL des CAS Server umgeleitet und müssen sich dort einloggen. Bei eingeloggten Usern wird das Ticket des Users vom CAS Server validiert bevor Zugriff auf den Client Service möglich ist.

Was ist die Aufgabe eines Tickets in einem TGT Service?

Mit einem TGT können Service Tickets vom TGS angefordert werden, die den Zugriff auf gesicherte Services ermöglichen.

Sources

- Apereo CAS Wiki
- Apereo CAS overlay template
- Baeldung CAS SSO with Spring Security
- CAS sample webapp
- Techopedia TGS
- CAS Infographic source