Stand: April 2023

Rollenbasierte Domänenabsicherung

Ziel

Mit **rollenbasierter Domänenabsicherung** können einzelne Domänen der Tools **für authentifizierte Benutzer**, basierend auf den ihnen zugeordneten Rollen, ein- bzw. ausgeblendet werden. So werden über spezielle Rollen *die* Domänen beschrieben, auf die der Zugriff erlaubt sein soll. Alle anderen Domänen werden "ausgeblendet".

Als **Paradigma** wird dabei die **transparente "Perspektive"** (oder "View") verwendet: Anfragen zur Liste aller Domänen werden nur mit den zugelassenen Domänen beantwortet und Zugriffsversuche auf andere Domänen werden so beantwortet, als gäbe es diese gar nicht. (Konkret werden solche Anfragen mit `UnknownObjectException` pariert statt mit `AuthenticationException` o.ä.) So ist es einem Nutzer auch nicht möglich, durch gezielte Anfragen herauszufinden, welche weiteren Domänen in der Instanz vorhanden sind.

Die **"Filterung"** der Domänen erfolgt **im Backend**, so dass die Zugriffe über **alle Schnittstellen entsprechend eingeschränkt** werden, sofern sie authentifiziert und mit aktivierter rollenbasierter Domänenabsicherung erfolgen.

Das zweistufige Rollensystem mit `admin`- und `user`-Rollen bleibt von rollenbasierter Domänenabsicherung unberührt und kann als komplementär dazu betrachtet werden.

Aktivierung

Die rollenbasierte Domänenabsicherung ist momentan in den Tools **gICS** und **E-PIX** für die Schnittstellen **WEB** und **SOAP** implementiert. Um sie zu nutzen, müssen die Schnittstellen der Tools grundsätzlich abgesichert verwendet werden: es muss also die Authentifizierung der einzelnen Schnittstellen aktiviert sein. Das funktioniert für alle Schnittstellen (wie gehabt) z.B. über `.env`-Variablen in der Docker-Konfiguration:

- für **Web per OIDC** (über Elytron-OIDC aktivierbar) in `envs/ttp_TOOL.env` mit
  - `TTP_EPIX_WEB_AUTH_MODE=keycloak`
  - `TTP_GICS_WEB_AUTH_MODE=keycloak`
- für **Web per gRAS** (über Elytron-JDBC) in `envs/ttp_TOOL.env` mit
  - `TTP_EPIX_WEB_AUTH_MODE=gras`
  - `TTP_GICS_WEB_AUTH_MODE=gras`
- für **SOAP per OIDC** (über KeyCloak-API) in `envs/ttp_TOOL.env` mit

    - `TTP_EPIX_SOAP_KEYCLOAK_ENABLE=true`
    - `TTP_GICS_SOAP_KEYCLOAK_ENABLE=true`

Um nun zusätzlich die rollenbasierte Domänenabsicherung zu steuern, gibt es drei Varianten:

- `DISABLED`
  Die rollenbasierte Domänenabsicherung ist deaktiviert, d.h. **alle Benutzer** haben Zugriff auf **alle Domänen**.
- `FORCED`
  Die rollenbasierte Domänenabsicherung wird für alle Benutzer erzwungen. D.h. für alle Benutzer ist der Zugriff nur auf *die* Domänen möglich, für die ihnen entsprechende Domänenrollen zugeordnet sind. Insbesondere haben in diesem Modus **Benutzer ohne** domänenbasierte Rollen Zugriff auf **keine Domänen**.
- `IMPLIED`
  Die Aktivierung der rollenbasierten Domänenabsicherung wird aus den Rollen der Benutzer abgeleitet. D.h. wenn Benutzer domänenbasierte Rollen haben, wird für sie die rollenbasierte Domänenabsicherung aktiviert und für sie ist der Zugriff nur auf *die* Domänen möglich, für die ihnen entsprechende Domänenrollen zugeordnet sind. Insbesondere haben in diesem Modus **Benutzer ohne** domänenbasierte Rollen Zugriff auf **alle Domänen**.

Ohne weiteres Zutun ist die rollenbasierte Domänenabsicherung implizit aktiviert, `IMPLIED` ist also der Standardmodus. Um die anderen Modi zu aktivieren, müssen (jeweils einzeln für die Tools) entsprechende `.env`-Variablen in der Docker-Konfiguration (in `envs/ttp_TOOL.env`) gesetzt werden, z.B.:

- `TTP_EPIX_AUTH_DOMAIN_ROLES=FORCED`
- `TTP_GICS_AUTH_DOMAIN_ROLES=DISABLED`

Alternativ können manuell in der `standalone.xml` des **WildFly** entsprechende Systemproperties eingetragen werden:

```
<system-properties>
  ...
  <property name="ttp.auth.epix.domain.roles" value="FORCED"/>
  <property name="ttp.auth.gics.domain.roles" value="DISABLED"/>
  ...
</system-properties>
```

Freischaltung von Domänen durch spezielle Rollen

Um bei aktivierter rollenbasierter Domänenabsicherung für einzelne Benutzer konkrete Domänen freizuschalten, müssen den Benutzern im **AllowList-Prinzip** in der Administration eines **OIDC-Servers** (z.B. Keycloak) oder in der **gRAS-Datenbank** spezielle Rollen zugeordnet werden.

Diese Zuordnung erfolgt per Namenskonvention: eine Domänenrolle ist ein **Muster** (Pattern), das eine **konkrete Domäne per Namen** oder auch eine **ganze Klasse von Domänennamen** beschreiben kann. Als **Mechanismus** werden dabei vereinfachte **Glob-Wildcards** verwendet (ohne Zeichenklassen, Bereiche oder Komplementierung). Das Matching erfolgt immer **case-insensitiv**.

Eine Domänenrolle hat die Form `:TOOL:DOMAIN`, wobei `TOOL` und `DOMAIN` natürlich mit Wildcards ausgedrückt werden können. Formal passt eine Domänenrolle immer mindestens auf die *Regex* `^:[^:]+:[^:]+`). Einige illustrierende Beispiele:

- `:*:*` erlaubt den Zugriff auf alle Domänen im E-PIX und im gICS
- `:epix:*` erlaubt den Zugriff auf alle Domänen im E-PIX
- `:epix:demo` erlaubt den Zugriff auf die Domäne "Demo" im E-PIX
- `:*:demo` erlaubt den Zugriff auf die "Demo"-Domänen in allen Tools
- `:gics:mii*` erlaubt im gICS den Zugriff auf alle mit "MII" beginnende Domänen
- `:gics:xyz ?? v2.?` erlaubt im gICS z.B. den Zugriff auf "XYZ DE v2.0" und "XYZ EU v2.1" aber nicht "XYZ v2.0" oder "XYZ DE v2"
- `:*:jmeter*` erlaubt den Zugriff auf alle JMeter-Test-Domänen im E-PIX und im gICS

Zur Illustration hier noch eine Übersicht für die Kombination der drei Aktivierungsmodi mit speziellen Domänenrollen beispielhaft für den Zugriff auf die Domäne `MII` im gICS:

Modus	**Ohne Domänenrollen**	`:*:*`	`:*:mii`	`:gics:mii`	`:gics:demo`	`:gics:*`	`:epix:mii`
`DISABLED`	YES	YES	YES	YES	YES	YES	YES
`FORCED`	**NO**	YES	YES	YES	**NO**	YES	**NO**
`IMPLIED`	YES	YES	YES	YES	**NO**	YES	**NO**

Fehlersuche

Hilfreich für das Debugging möglicher Probleme könnte der `DEBUG`-Modus verschiedener Logger sein. Für die Docker-Konfiguration kann dieser mit folgenden `.env`-Variablen aktiviert werden:

```
WF_CONSOLE_LOG_LEVEL=DEBUG
TTP_EPIX_LOG_LEVEL=DEBUG
TTP_GICS_LOG_LEVEL=DEBUG
TTP_AUTH_LOG_LEVEL=DEBUG
TTP_WEB_LOG_LEVEL=DEBUG
```

Die Variablen sind verteilt auf `ttp-common.env` und `ttp_TOOL.env` in `envs/`.