Configure PREAUTH authentication

Background

Users that access the INTERGATOR web-interface directly are authenticated by intergator either using single-sign-on (SSO) or username and password.

When INTERGATOR is integrated into Confluence, users are already authenticated by Confluence. The Confluence Plugin then sends search requests to INTERGATOR on behalf of the user and visualizes the results.

In this scenario Confluence is located in between the user and intergaor but it does not have the user credentials (like password or SSO information) to authenticate the user with INTERGATOR.

For such use cases there is an INTERGATOR authentication mode called PreAuth. When this mode is configured, INTERGATOR relies on the upstream compunent (e.g. Confluence) to authenticate the user. The upstream application then sends the username to INTERGATOR in an HTTP header (typicallly X-User) or cookie and INTERGATOR accepts it without further checking.

Security considerations

Without PreAuth, all INTERGATOR APIs require authentication. With PreAuth mode enabled all INTERGATOR APIs have to support this mode. That means that any user that can talk to these APIs can impersonate any other user and e.g. donwnload documents from connected data sources.

To prevent this you have to make sure that only legitimate users and machines can access the INTERGATOR server and API proxy API. This can be done e.g. by a local firewall on the INTERGATOR machine(s) or by putting INTERGATOR into a separate network with restricted access.

The INTERGATOR webinterface can be configured in a way that only a dedicated IP address can use PreAuth mode. Requests from all other IP addresses have to be authenticated with username and password or SSO.

PreAuth configuration

PreAuth mode has to be enabled in the INTERGATOR server and the web-interface. The API proxy just passes it through.

This article assumes that your INTERGATOR is already connected to a Microsoft Active Direcory.

INTERGATOR server configuration

The server allows to configure a list of authentication providers. When a user logs in all authentication providers are invoked sequentially until the user is authenticated. Now we configure the PreAuth povider **before** the existing Active Directory provider.

Open the INTERGATOR Management Center and go to *XML mode Server Sicherheitskonfiguration*

Inside the **<providers>** XML element you find an **<item>** with a **<factoryId>active-directory-auth-provider-factory</factoryId>** inside. This configures your Active Directory connection. Keep this! Do not change it!

Put the following <item>-element and descendants directly **before** the item of your Active Directory item:

PreAuth provider snippet to insert

```
<item>
  <factoryId>pre-auth-auth-factory</factoryId>
  <properties>
    <item>
      <key>principalType</key>
      <value>user-principal-name</value>
    </item>
  </properties>
</item>
```

The result should be similar to this:

Server Sicherheitskonfiguration (server security configuration)

```
<configuration xsi:type="serverSecuritySettings" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <label>Server-Sicherheitskonfiguration</label>
    <customAuthChain>
        <providers>
            <item>
                <factoryId>pre-auth-auth-factory</factoryId>
                <properties>
                    <item>
                        <key>principalType</key>
                        <value>user-principal-name</value>
                    </item>
                </properties>
            </item>
            <item>
                <factoryId>active-directory-auth-provider-factory</factoryId>
                <properties>
                    <item>
                        <key>java.naming.security.principal</key>
                        <value>aUser@yourDomain</value>
                    </item>
                    <item>
                        <key>java.naming.security.credentials</key>
                        <value>...some_encrypted_password...</value>
                    </item>
                    <!-- further item-elements can follow, this is environment specific -->
                </properties>
            </item>
        </providers>
...
<!-- don't touch the resolver -->
```

That's all for the server side - no restart needed. Check if normal authentication via AD still works and make sure the INTERGATOR server and API proxy API can not be accesed by unauthorized users (see *Security considerations* above).

Web-interface configuration

Add PREAUTH properties

Create a file **web-interface/intergator/WEB-INF/properties/intergator.confluence-preauth.properties** with the following content:

```
intergator.auth.modes.default=PREAUTH
intergator.auth.modes.default.embedded=PREAUTH
intergator.auth.trusted.header=X-PREAUTH-301238
```

Restrict access to specific hosts

Edit **web-interface/intergator/WEB-INF/intergator.xconf** and add a similar **match** rule:

- Replace the IP address in the fixed-string-attribute with the IP of your Confluence server

intergator.xconf

```
<intergator xml:space="preserve">
        <properties name="ig-properties">

        ...

                <!-- Allow PREAUTH only from specific host -->
                <match>
                        <match fixed-string="192.0.2.0" value="{request:remote-addr}">
                                <url expand="static" optional="true" path="/WEB-INF/properties/intergator.
confluence-preauth.properties" />
                        </match>
                </match>

    ....
```

Restart the INTERGATOR web-interface

This depends on the operating system. It is necessary to load the new configuration.

Add a custom authentication resolver

The custom resolver is only needed if your intergator **is not connected to an Active Directory**.

Create file **server/lib/groovy/de/ifbus/intergator/authsystems/custom/ConfluencePreauthResolver.groovy** with the following content:

ConfluencePreauthResolver.groovy

```
/*
 * Copyright (c) 2020 interface projects GmbH
 *  All rights reserved. This code is property of interface projects GmbH.
 *  Modification and/or further distribution without permission of
 *  interface projects GmbH is prohibited.
 */
package de.ifbus.intergator.authsystems.custom;

import de.ifbus.intergator.core.security.NoSuchPrincipalException
import de.ifbus.intergator.core.security.ScAuthResolver
import de.ifbus.intergator.core.security.ScId
import de.ifbus.intergator.core.security.ScPrincipal
import de.ifbus.intergator.core.security.ScPrincipalBuilder
import de.ifbus.intergator.core.security.ScPrincipalType
import de.ifbus.intergator.core.security.ScQualifier

import java.util.logging.Level
import java.util.logging.Logger

class ConfluencePreauthResolver implements ScAuthResolver {
    private static final Logger log = Logger.getLogger(MyAuthResolver.class.getName())

    private static final String DIRECTORY = "customexampledirectory"
    private static final String IDTYPE_ID = "user"

    MyAuthResolver() { log.log(Level.FINE, "MyAuthResolver instantiated") }

    List<ScPrincipal> resolve(ScId securityId) throws Exception, NoSuchPrincipalException {
        log.log(Level.FINE, "MyAuthResolver resolving $securityId")

        // anonymous logon
        if (securityId == null) {
            List<ScPrincipal> principals = new ArrayList<>(); return principals
        }

        // get userPrincipalName from SecId [value=user@example.com, qualifiers=[userprincipal-name]]
        String userPrincipalName = securityId.getValue()

        List<ScPrincipal> principals = new ArrayList<>()

        // Add the user name princpial
        principals.add(createUserPrincipal(userPrincipalName))

        log.log(Level.FINE, "ConfluencePreauthResolver resolved $securityId to $principals")

        return principals
    }

    // custom code that creates the user principal
    ScPrincipal createUserPrincipal(String userPrincipalName) {        // the unique user identifier
        String id = userPrincipalName
        return new ScPrincipalBuilder()
                .setType(ScPrincipalType.USER)
                .setDirectory(DIRECTORY)
                .setSystemId(DIRECTORY + ":" + IDTYPE_ID + ":" + id)  //restart server on change
                .setDisplayName(userPrincipalName != null ? userPrincipalName : id)
                .addId(userPrincipalName, ScQualifier.USERNAME)
                .addId(userPrincipalName, ScQualifier.DISPLAY_NAME).build()
    }
}
```