Integrate Third Party Gadget with Finesse on SSO Mode

Contents

Introduction

This document describes what is needed for the integration of a 3rd party gadgets with Finesse while the system is on Single Sign-on (SSO) mode. An example is also given for NON SSO mode.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco Finesse
- SSO
- Finesse 3rd party gadgets

Components Used

The information in this document is based on these software and hardware versions:

- Cisco Finesse version 11.6
- SSO
- 3rd party gadget
- 3rd party REST service.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

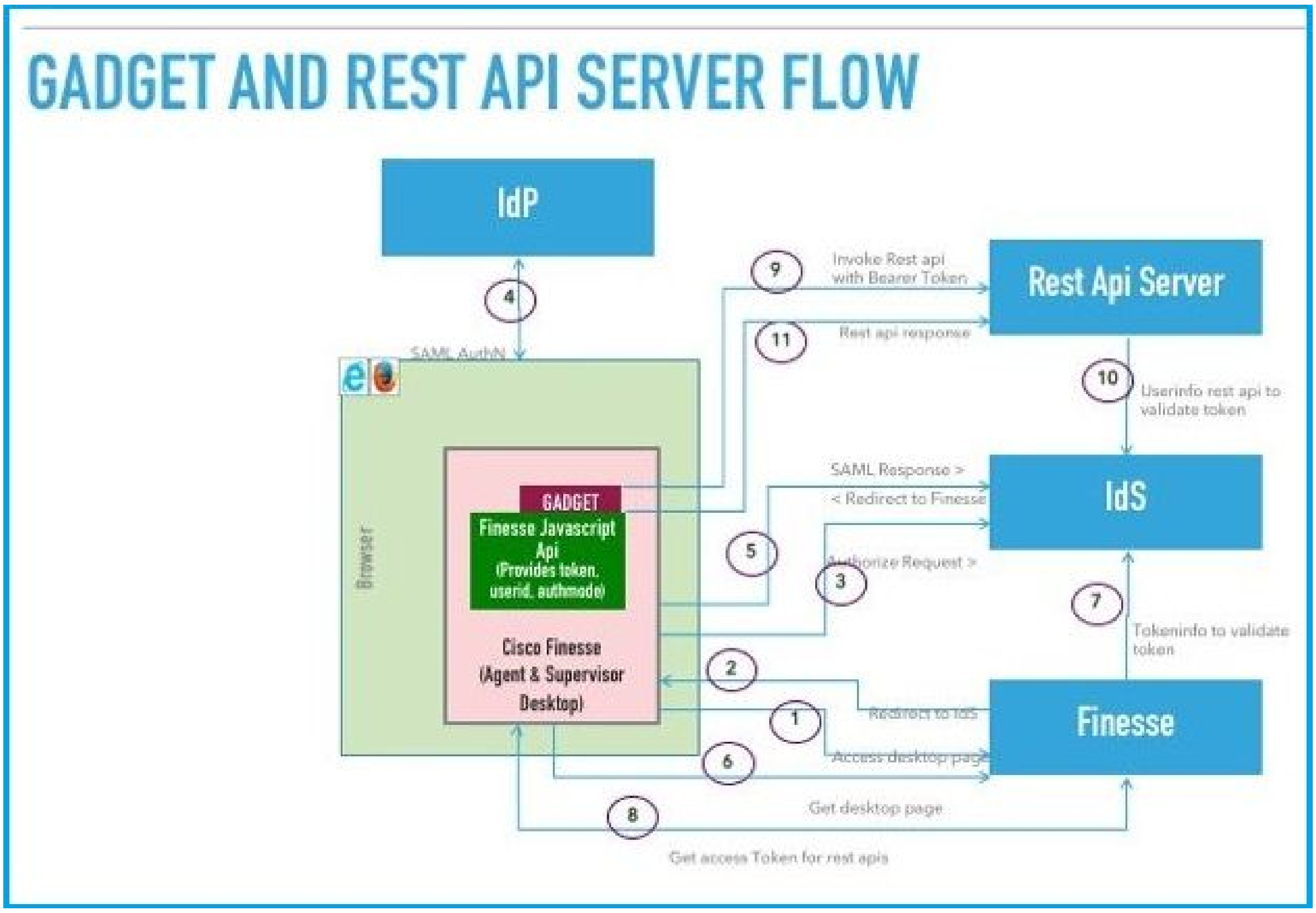
These are the initial steps while the agent attempts to log in and authenticate with SSO or NONSSO.

the second step describes what needs to be considered after successful authentication in case of SSO and NONSSO.

1. At the time of desktop log in, Finesse detects the System Auth mode (SSO/NONSSO) and based on the Auth mode, appropriate Loginpage is displayed. Users see the IDP Login page in case of SSO mode and Finesse Login page in case of NONSSO mode.
2. After successful authentication, all requests are authenticated based on system Auth mode. For SSO deployments, all requests to Finesse carry access token as part of request header. The token is validated against IDP server for successful authentication. However, for requests to 3rd party web services, the Auth header has to be set based on the authentication scheme implemented by the 3rd party web service. In case of NONSSO deployment, all requests carry the **Basic** Auth header with base64 encoded username and password. All requests in this case are validated against the Finesse local database.

Explanation of Basic Model of Interaction for SSO mode

This *image* shows the basic model of interaction between a 3rd party gadget, Finesse, IDS, and a 3rd party REST service, when the system is in SSO mode.


GADGET AND REST API SERVER FLOW
IdP
4
SAML AuthN
Browser
GADGET
Finesse Javascript Api (Provides token, userid, authmode)
Cisco Finesse (Agent & Supervisor Desktop)
9
Invoke Rest api with Bearer Token
Rest Api Server
11
Rest api response
10
Userinfo rest api to validate token
SAML Response >
< Redirect to Finesse
IdS
5
Authorize Request >
3
7
Tokeninfo to validate token
2
Redirect to IdS
1
Finesse
Access desktop page
6
Get desktop page
8
Get access Token for rest apis

Image

Here is the description for every step shown in the image.

1. Agent/Supervisor accesses the Finesse desktop URL.
(Example: [https://finesse.com:8445/desktop)](https://finesse.com:8445/desktop)

2. Finesse detects that authentication mode is SSO and redirects the browser to IDS.
3. Browser sends the redirect authorize request to IDS. At this point, IDS detects whether *the user* has a valid access token or not. If *the user* doesn't have a valid access token, IDS redirects to the Identity Provider (IdP).
4. If the request is redirected to IdP, IdP provides the L*ogin* page for authenticating *the user.*
5. The SAML assertion from IdP is sent to the IDS, which redirects back to the Finesse desktop.
6. Browser does a GET of the Finesse desktop page.
7. Finesse gets the access token from IDS with the SAML auth code.
8. Desktop gets the access token to be used to authenticate subsequent REST APIs.
9. 3rd party gadget loads into the desktop and invokes a 3rd party REST API with the access token (bearer) in the auth-header.
10. 3rd party REST service validates the token with IDS.
11. 3rd party REST response is returned to the gadget.

Configuration of gadgets.io.makerequest for SSO and NONSSO mode

Step 1. For Finesse REST API calls made via Shindig , gadgets need to add "Bearer" authorization header in gadgets.io.makeRequest headers.

Step 2. Gadgets need to make native gadgets.io.makeRequest calls for all REST requests, the authorization header has to be set inside the request params.

For NON SSO deployments, This is the Auth Header.

```
"Basic " + base64.encode(username : password)
```

For SSO deployments, this is the Auth header.

```
"Bearer "  + access_token
```

Access token can be retrieved from **finesse.gadget.Config** object.

```
 access_token = finesse.gadget.Config.authToken
```

The new authorization header mustl be added to the request params.

```
 params[gadgets.io.RequestParameters.HEADERS].Authorization = "Basic " + base64.encode(username
: password);
params[gadgets.io.RequestParameters.HEADERS].Authorization = "Bearer " + access_token;
```

Step 3. A utility method **getAuthHeaderString** has been added inside **utilities.Utilities**. This utility method takes the config object as argument and returns the authorization header string. Gadgets can make use of this utility method to set the authorization header in request params.

```
params[gadgets.io.RequestParameters.HEADERS].Authorization=
finesse.utilities.Utilities.getAuthHeaderString(finesse.gadget.config);
```

Note: For API requests to 3rd party web services, the auth header has to be set based on the authentication scheme implemented by the 3rd party web service. Gadget developers have the freedom to use basic auth or bearer token based authentication, or any other authentication mechanism of their choice.