WSSO configuration

Configuring Soffid WSSO

The system is configured using Apache configuration files, plus some ECMA script files that can be located anywhere you want.

In general, Soffid WSSO acts intercepting and modifying any request made to Apache. This request can be processed by Apache itself or forwarded to another web or application server using ProxyPass module.

SoffidPostData directive

Syntax:

```
SoffidPostData [path-regex] [system=...] [account=...] [password=....] [contains=....] [flags]
```

```
SoffidPostData [path-regex] [system=...] [account=] [post= "DataToBePosted"] [replace | append | prepend | merge] [contains=...] [flags]
```

When the user agent performs a POST request to a path that matches the specified regular expression, Soffid WSSO will intercept it. Mind the path is missing any host, protocol or port qualifier.

If the requested file matches the regular expression, and the user agent is using the POST method, Soffid WSSO will modify the posted content. Optionally you can specify a regular expression that must match the original content of the sent message.

The first step performed by Soffid WSSO is to parse the posted data to get the user account sent. The user account sent will be the one whose name matches the account parameter of the SoffidPostData directive. If SoffidPostData directive lacks “account=...” the default (j_user) parameter name will be used.

The next step is to retrieve the password that belongs to the specified account on the system set at SoffidPostData directive.

The final step is to add or change the password sent. The name of the password parameter should be set by the SoffidPostData directive. If none is specified, the default (j_password) parameter name will be used.

For more sophisticated applications, the full content of the post data can be specified the syntax post="DataToBePosted.". On this data to be posted, the administrator can specify some expressions that will be replaced by the proper value:

- ${secret.xxx} where xxx names for a secret stored at the user secrets vault.
- ${account} will be resplaced by the used account.
- ${password} will be replace by the account password.

The data sent by the browser will be combined with the new posted data in four different ways:

- replace: the data sent by the browser is completely ignored.
- append: the new data will be appended to the original one.
- prepend: the new data will precede the original one.
- merge: each parameter of the original request will keep its position despite its value is to be changed. New parameters will be appended after the original ones.

Additionally, the following flags are allowed:

- force: By default, the password will be injected only when no password is sent, or it's empty. If you'd like to overwrite the password sent by the user agent, this flag must be specified.
- requiresAccount: If specified, Soffid WSSO will not try to guess the account to use if it is not send by the user agent.

Example:

```
SoffidPostData .* /j_security_check system=soffid account=j_user
"j_user=${account}&j_password=${password}"
```

```
SoffidPostData .* /j_security_check system=soffid account=j_user password=j_password
```

SoffidBasicAuthorization directive

Syntax:

```
SoffidBasicAuthorization [path-regex] system
```

This Appache directive allows WSSO to inject user name & password on web applications using HTTP BASIC authentication scheme. If the path being accessed matches the regular expression, Soffid WSSO will inject the “Authorization: Basic” header as stated at RFC 2617.

The system must match a managed system, also named Agent, listed at Soffid IAM Console.

Example:

```
ProxyBasicAuthorization ^/secure/.*$ "soffid"
```

SoffidOnLoadScript directive

Syntax:

```
SoffidOnLoadScript [path-regex] content-regex maxSize scriptFile
```

To improve the user experience it's necessary to hide the underlying application-specific user authentication form. In order to do this, the web pages generated by the applications can be modified in order to skip unnecessary forms or to request a global login authentication.

ProxyOnLoadScript directive triggers the execution of an ECMA script that is able to modify the page generated by the application server. As long as the script execution is a rather heavy task, WSSO must accurately detect which pages must trigger the script execution. Currently, the script execution can be triggered by contents and by page path.

Allowed parameters:

path-regex	Regular expression that will be matched against the page path, excluding any trailing parameter, sent after a question mark.
content-regex	Regular expression that will be matched against the page contents.
maxSize	Maximum page size that will be handled. Whether the page size exceeds this limit, it will be sent to the user agent.
scriptFile	File containing the script to be executed. Note the script must be UTF-8 encoded.

Regards this directive could potentially manage any kind of media generated by a web server, it is designed to managed XML or HTML files. Anyway, image resources will never trigger the script execution.

SoffidCookieName directive

Syntax:

```
SoffidCookieName cookie-name
```

Sets the name of the cookie used by Soffid WSSO to track the single sign on session

SoffidCookieName directive

Syntax:

```
SoffidCookieDomain domain-name
```

Sets the domain of the cookie used by Soffid WSSO to track the single sign on session. By default, the cookie is attached to the virtual server name

SoffidCookiePassthrough directive

Syntax:

```
SoffidCookieDomain domain-name
```

By default the WSSO engine will hide actual cookies to the browser, and only the Soffid cookie will be sent. In order to share the cookie names and values with the browser, one can add a regular expression to match the cookies to unhide.

To unhide all cookies, use:

```
SoffidCookiePassthrough .*
```

Revision #3
Created 7 June 2022 14:18:40
Updated 17 October 2024 09:27:05