Solid-cURL

Daniel Schraudner[1]

[1]Friedrich-Alexander-Universität Erlangen-Nürnberg, Nuremberg, Germany

Abstract

We implemented Solid-cURL, a tool that extends cURL be adding Solid-OIDC authentication. Solid-cURL can be used by Solid developers and researchers that aim to use a tool of the simple elegance as cURL but do not want to handle the Solid authentication manually for every request. We described the different components of Solid-cURL and how they interact with each other.

Keywords

Solid, Tool, Open ID Connect

1. Introduction

In their daily work, Solid developers and researchers often have to manually send specific HTTP request to Solid servers. Doing this with a web browser is often not very practical as one first has to login with Solid-Open ID Connect (Solid-OIDC) [1] using a Solid app. The requests that can be send afterwards are only those that are allowed by the app.

Web developer and researchers in this situation used the widespread tool cURL[1]. cURL is a command line program that allows to specify the parameters of an HTTP request (method, headers, etc.) in every detail. cURL executes this request and displays the result to the user.

Example 1. Creating a new RDF resource in a container on a Solid Pod that is publicly writeable:

```
curl -X POST https://dschraudner.solidcommunity.net/public/
-d "<> a <http://ex.org/A\> ." -H "Content-Type: text/turtle"
```

As cURL allows to specify arbitrary headers for a request it could theoretically be used to send authenticated requests to Solid servers by copying the Solid-OIDC specific headers (access token [2] and Demonstrating Proof of Possession (DPoP) token [3]) to the cURL command line parameters. This approach, however is very impractical, as the DPoP header changes every time the request method or URI changes; the access token automatically expires after some time.

Example 2. Creating a new RDF resource in a container on a Solid Pod with authentication:

```
curl -X POST https://dschraudner.solidcommunity.net/public/
```

The 1st Solid Symposium Poster Session, co-located with the 2nd Solid Symposium, May 02 – 03, 2024, Leuven, Belgium

daniel.schraudner@fau.de (D. Schraudner)

0000-0002-2660-676X (D. Schraudner)

[1]https://curl.se/

```
-d "<> a <http://ex.org/A\> ." -H "Content-Type: text/turtle"
-H "Authorization: DPoP eyJhbGciOiJFUzI1NiIsInR5cCI6ImF0K2p3dCIsImtpZCI
6IlJTWXloeE4wb1lsb0JOUVQ1OExZTkhpSzNOTWhRa1QyQ3FZMVJzbzd6N28ifQ.eyJ3ZW
JpZCI6Imh0dHBzOi8vdG9tLnNvbGlkLmFpZmIua2l0LmVkdS9wcm9maWxlL2NhcmQjbWUi
LCJqdGkiOiJ2dUJQVHVIcmRTZlVOS0ZCc0tQelEiLCJzdWIiOiJodHRwczovL3RvbS5zb2
xpZC5haWZiLmtpdC5lZHUvcHJvZmlsZS9jYXJkI21lIiwiaWF0IjoxNzE3NjY1NTI0LCJl
eHAiOjE3MTc2NjkxMjQsInNjb3BlIjoiIiwiY2xpZW50X2lkIjoiNl9sc0J4V2p0UzM3TV
hOLUdCQm8xIiwiaXNzIjoiaHR0cHM6Ly9zb2xpZC5haWZiLmtpdC5lZHUvIiwiYXVkIjoi
c29saWQiLCJjbmYiOnsiamt0IjoiQm1kY0tDdVJOTktsYy1yUmxsZzRKRnJsN3VDdnlSd
GUxeWVoRkpjRnFSayJ9fQ.PUHYhDLT4l9nHkmsXZXe0I_7nYJCQuq5vyBb8d1eYVlV55gu
JDacrN-HKNnPH6DrOkcCHjvqQxesGLvz7qXvUw"
-H "DPoP: eyJhbGciOiJFUzI1NiIsImp3ayI6eyJjcnYiOiJQLTI1NiIsImt0eSI6IkVD
IiwieCI6ImtHSk81QUx5emZnQlNsYXZTOUJrX2FOV1ZLR1pqaTY2NkJ1YTNGWUoyX28iLC
J5IjoiWHVlRWJfdmNVX2c0cGdaQS1TMHg2R3BYWE02Z2RwMUVWV19jTVdGb3VPSSIsImFs
ZyI6IkVTMjU2In0sInR5cCI6ImRwb3Arand0In0.eyJodHUiOiJodHRwczovL2Jhbmsuc2
9saWQuYWlmYi5raXQuZWR1L29mZmVyLzEiLCJodG0iOiJHRVQiLCJqdGkiOiI3YzJkNGRh
Zi00ZWYyLTQ1MWQtODY1MS1hMWE1ZjUwOThjZDgiLCJpYXQiOjE3MTc2NjU1MjV9.aK_Si
ndnZNLhjeSQnfX551lwMm6oW7Hw9RGXWNqESGPO3JqWEE9sSImATEfJR_iOF-nrqXz1zdH
BxztdwoG1Ww"
```

In this paper we present Solid-cURL. Solid-cURL is a program that tries to mimic the command line interface and behaviour of cURL. Additionally it allows users to log in via Solid-OIDC. The credentials can be stored and used for future HTTP request by specifying a command line parameter.

Example 3. Creating a new RDF resource in a container on a Solid Pod using authentication with solid-curl

```
solid-curl -X POST https://dschraudner.solidcommunity.net/public/
-d "<> a <http://ex.org/A\> ." -H "Content-Type: text/turtle" -u daniel
```

In the remainder of this paper, we will give first go through related work, then give an overview of the different components of Solid-cURL, introduce our implementation of Solid-cURL and give a short outlook.

2. Related Work

Penny[2], Solid Filemanager[3], Solid IDE[4], and SolidOS[5] are examples of Solid apps that work like file managers. They allow to create new resources and containers on the Pod and allow to alter and delete existing ones. They, however, do not let the user exercise such a fine-grained control that is possible by manually sending HTTP request using cURL (e.g. setting specific headers).

[2] https://penny.vincenttunru.com/
[3] https://otto-aa.github.io/solid-filemanager/
[4] https://jeff-zucker.github.io/solid-content-manager/
[5] https://github.com/SolidOS/solidos

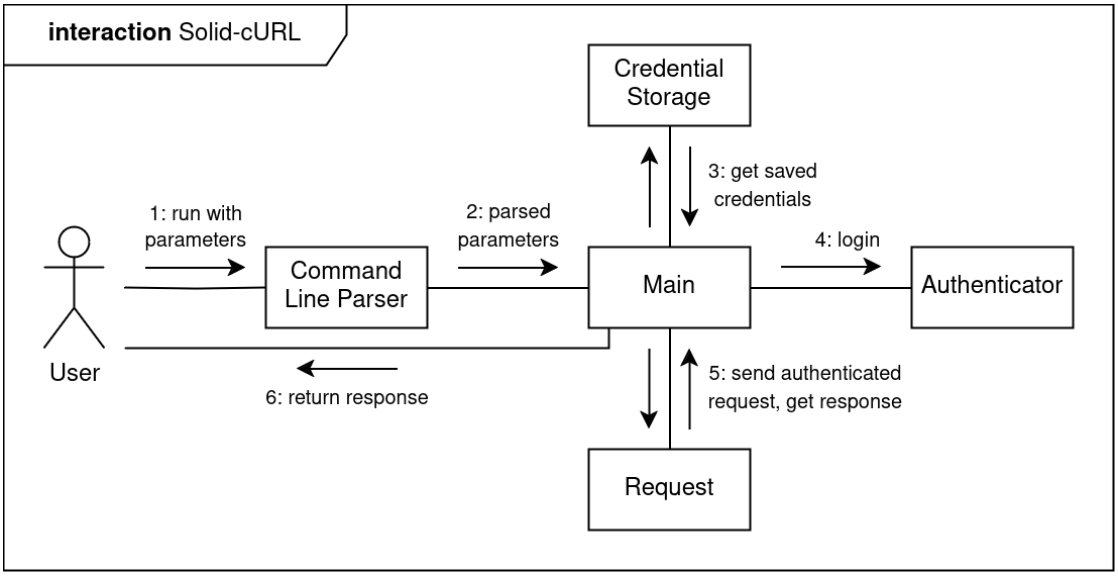

Figure 1: UML interaction diagram showing the different components of Solid-cURL and their interaction

Solid-shell[6] is a command line tool for managing data on a Solid Pod. It allows the user to log in and use different file system-specific operations like copy, move, recursiveDelete, etc. Also the names of the different HTTP methods are available as commands, they, however, only allow to specify the URI and the body; headers and other additional settings that cURL is capable of controlling are hidden for the user.

Bashlib[7] is another command line tool whose main focus is to be used together with the Community Solid Server [4]. It has a lot of capabilities like the usual file system operations but it can also query a resource with SPARQL [5] or manipulate ACL permissions. Here, too, the user has no fine-grained control over the exact HTTP request that is sent to the Solid Pod.

3. Components

In Figure 1 a UML interaction diagram showing the principal components can be seen and their interaction.

The Command Line Parser is responsible for parsing and interpreting the command line input of the user and giving it to the Main component of Solid-cURL. Based on the parameters (more specifically, the Solid user specified as parameter) the Main component uses the Credential Storage component to retrieved previously saved credentials. The credentials are then given to the Authenticator component which carries out the login process (i.e. it retrieves and access token and creates a DPoP). Access token and DPoP are given to the Request component who carries out the actual request according to the command line parameters and with the correct

[6] https://github.com/jeff-zucker/solid-shell/
[7] https://github.com/SolidLabResearch/Bashlib

Solid-OIDC headers set. The response is then printed to the user on the command line.

4. Implementation

We implemented Solid-cURL in Javascript. The source code is available via GitHub[8] and an executable script can be installed using the Node Package Manager[9].

For the Command Line Parser component we used the Commander.js library[10]. For the Credential Storage component we used the operating system's keychain accessed using the keytar library[11]. For the authenticator component we used the solid-client-authn-node library[12] from Inrupt. For the request component we also used the solid-client-authn-node library or the fetch provided by it, respectively.

5. Outlook

For the future we want to extend the capabilities (regarding the possible options and parameters) of Solid-cURL to match those of the original cURL. For this we started to rewrite Solid-cURL in C++[13] to be able to use the libcurl library. cURL is also directly based on this library which makes it easier to mimic cURL capabilities using this libraries. Implementing Solid-cURL in C++ could also positively impact its performance, as JavaScript is usually compiled just-in-time which leads to a warm-up delay when starting the program [6].

For C++, however, there exists to our knowledge no library for Solid-OIDC which means we will have to handle all the different tokens used in the process manually.

Acknowledgments

This work is partially funded by the German Federal Ministry of Education and Research via the MANDAT project (FKZ 16DTM107A).

References

[1] A. Coburn, D. Zagidulin, A. Migus, R. White, et al., Solid-oidc, Specification. Solid (2022).
[2] D. Hardt, The OAuth 2.0 authorization framework, Technical Report, 2012.
[3] D. Fett, Authlete, Campbell, OAuth 2.0 Demonstrating Proof of Possession (DPoP), Technical Report, 2012.
[4] J. Van Herwegen, R. Verborgh, The community solid server: Supporting research & development in an evolving ecosystem, Semantic Web Journal. https://doi. org/10.5281/zenodo 7595116 (????).

[8] https://github.com/wintechis/solid-curl
[9] https://www.npmjs.com/package/solid-curl
[10] https://github.com/tj/commander.js?
[11] https://github.com/atom/node-keytar
[12] https://github.com/inrupt/solid-client-authn-js
[13] https://purl.org/solid-curl/cpp

[5] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of sparql, ACM Transactions on Database Systems (TODS) 34 (2009) 1–45.
[6] M. Selakovic, M. Pradel, Performance issues and optimizations in javascript: an empirical study, in: Proceedings of the 38th International Conference on Software Engineering, 2016, pp. 61–72.