After Brazil's Data Protection Law: Authorizing the Access to Data in Solid Apps

Jefferson O. Silva
silvajo@pucsp.br
Pontifical Catholic University of São Paulo
São Paulo, Brazil

Newton Calegari
newton@nic.br
Brazilian Network Information Center - NIC.br
São Paulo, Brazil

Diogo Cortiz
Inria Paris-Rocquencourt
São Paulo, Brazil

ABSTRACT

Write the abstract here.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Solid, Access control, Decentralized web, Frameworks, Guardian

ACM Reference Format:
Jefferson O. Silva, Newton Calegari, and Diogo Cortiz. 2018. After Brazil's Data Protection Law: Authorizing the Access to Data in Solid Apps. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

With the approval of the Brazilian General Data Protection Law (LGPD),[1] several software companies need to redesign the applications that handle the personal data of Brazilian citizens. LGPD is based on the General Data Protection Regulation (GDPR),[2] which aims at protecting the personal data of EU individuals. In total, around 120 countries adopt comprehensive privacy laws and regulations to protect personal data held by private and public bodies [1].

Tim Berners-Lee and colleagues [REF] propose a platform called Solid (derived from "**So**cial **li**nked **d**ata"), which can be described as a set of principles, conventions, and tools for building decentralized Web applications. An application is considered decentralized when it does not hold users' personal data [REF]. LGPD considers personal any data that directly or indirectly leads to the identification of a user [REF]. Solid is based on the principle that users should have full ownership of their personal data. Currently, applications (e.g., Facebook, LinkedIn, Santander) work as "data silos" and all the personal data created in these platforms are controlled by the application companies. In constrast, decentralized Web applications provide complete separation between users' data and the applications that create and consume this data. While users store data in Web-accessible personal online datastores (pods), applications access users data relying as much as possible in W3C standards and Semantic Web technologies [REF]. Pods are independent of applications, which means that the users can change the application that create or consume their personal data at anytime. Users can also grant or restrict access to their pods using Web Access Controls (WAC).

One implication of using Solid in the context of LGPD is that decentralized Web applications need to respond differently according to the citizenship of the user.

RQ1: How decentralized applications can implement fine-grained access control.

RQ2: How decentralized applications can implement fine-grained access control. We propose

2 BACKGROUND

In this section, we offer some background on LGPD, decentralized Web, and on access control.

2.1 Brazilian Data Protection Law

The LGPD is strong inspired by the European GDPR. The Brazilian Bill, as the European one, defines cross-border jurisdiction, thus the Bill is applicable to any organizations processing personal data of Brazilian residents, whether it is headquartered in Brazil or not.

LGPD has also included the right of data portability, the right of access to personal data by the owner, and the right of erasure. Differently of the GDPR, which imposes 30 days for the controllers to comply with these requests, the LGPD imposes 15 days.

The Brazilian law also requires companies to nominate a Data Protection Officer (DPO) who will be in charge of monitoring the adoption of best practices for personal data protection and for reporting to the National Data Protection Authority (ANPD).

[1] http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm
[2] http://data.consilium.europa.eu/doc/document/ST-9565-2015-INIT/en/pdf

LA WEB, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

```
# Contents of https://alice.databox.me/docs/file1.acl
@prefix  acl:  <http://www.w3.org/ns/auth/acl#>  .

<#authorization1>
    a              acl:Authorization;
    acl:agent      <https://alice.databox.me/profile/card#me>;  # Alice's WebID
    acl:accessTo   <https://alice.databox.me/docs/file1>;
    acl:mode       acl:Read,
                   acl:Write,
                   acl:Control.
```

Figure 1: Example WAC Document

```
# Contents of https://alice.databox.me/docs/shared-file1.acl
@prefix  acl:  <http://www.w3.org/ns/auth/acl#>.

# Group authorization, giving Read/Write access to a group, which is
# specified in the 'work-groups' document.
<#authorization2>
    a              acl:Authorization;
    acl:accessTo   <https://alice.example.com/docs/shared-file1>;
    acl:mode       acl:Read,
                   acl:Write;
    acl:agentGroup <https://alice.example.com/work-groups#Accounting>;
```

Figure 2: WAC document with group permission

```
# Contents of https://alice.example.com/work-groups
@prefix   acl:   <http://www.w3.org/ns/auth/acl#>.
@prefix  vcard:  <http://www.w3.org/2006/vcard/ns#>.

<> a  acl:GroupListing.

<#Accounting>
    a              vcard:Group;
    vcard:hasUID   <urn:uuid:8831CBAD-1111-2222-8563-F0F4787E5398:ABGro
    dc:created     "2013-09-11T07:18:19+0000"^^xsd:dateTime;
    dc:modified    "2015-08-08T14:45:15+0000"^^xsd:dateTime;

    # Accounting group members:
    vcard:hasMember  <https://bob.example.com/profile/card#me>;
    vcard:hasMember  <https://candice.example.com/profile/card#me>.
```

Figure 3: WAC document listing group members

The regulation defines the concepts of personal data as "any data, isolated or aggregated to another, that may allow the identification of a nutral person or subject them to a certain behavior" [REF] (IAPP: https://iapp.org/news/a/the-new-brazilian-general-data-protection-law-a-detailed-analysis/); sensitive data refers to data that may be subject to discriminatory practices, such as political opinion, sexual life, religious belief, genetic and biometric data, and it should have additional security layers; Unless it is possible to reverse-engineering the anonimized data, the law does not apply to this kind of data.

In a technical perspective, the efforts related to the decentralization of the Web help to build systems that are privacy-friendly, respecting user's privacy and in compliance with the regulations. Ainda nao sei se esse paragrafo fica nessa section ou na proxima.

2.2 Decentralized Web

Teste [3]

2.3 Access Control in Solid

Access control is typically split into two distinct procedures: authentication, and authorization. While authentication is concerned with determining whether an agent (e.g., user, group) is whom it claims to be, authorization is responsible for verifying if the agent is allowed to access a protected resource (e.g., document) or operation (e.g., read, write, append).

The Solid project uses the Web Access Control (WAC) specification for controlling the access to protected resources. WAC specifies a decentralized cross-domain access control system, similar to existing access control models. According to the specification, WAC has the following key features:

(1) The resources are identified by URLs and can refer to any web documents or resources.
(2) It is declarative – access control policies are written in regular web documents.
(3) Users and groups are also identified by URLs (specifically, by WebIDs).
(4) It is cross-domain – all of its components, such as resources, agent WebIDs, and even the documents containing the access control policies, can potentially reside on separate domains.

Figure 1 shows an example of a WAC document that specifies that Alice (as identified by her WebID `https://alice.databox.me/profile/card#me`) has full access (read, write, and control) to one of her web resources, located at `https://alice.databox.me/docs/file1`.

In Figure 2, we can see that it is possible to give access to a group of agents using the `acl:agentGroup` predicate. In this case, members of the group `Accounting` can read and write the Alice's resource located at `https://alice.example.com/docs/shared-file1`.

A group is a collection of members (or WebIDs) that needs to be specified in a different file. Figure 3 depicts the listing of a group. In this case, `Bob` and `Candice` belong to the `Accounting` group. Additionally, it is possible to give access to all agents (public access) or yet to all authenticated agents. It is also possible to classify web apps as trusted. Furthermore, not every document needs its own individual access control list file. Rather, it is possible to create an authorization to a container, which is a web location that contain multiple resources.

3 ESFINGE GUARDIAN

In this section, we present the Esfinge Guardian[3] framework. As depicted in Figure 5, Guardian is composed of eight main elements.

[2]

[3]https://github.com/EsfingeFramework/guardian

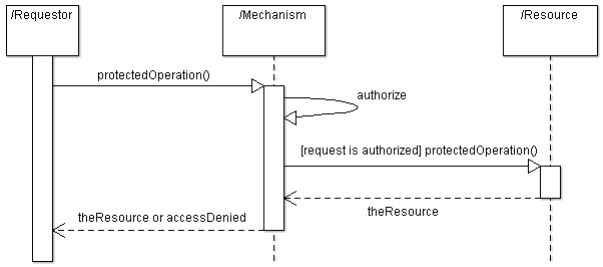

Figure 4: A conceptualization of the interception mechanism [4]

`AuthorizationContext`. It is the central entity that holds all the information required for an authorization, which includes the data for the subject, resource, and environment. That means all other entities should provide AuthorizationContext with enough information for authorization to occur. It is also an interface with the user, meaning that all other points must be hidden from the user.
`GuardianInterceptor`: is the entity responsible for abstracting the different existing interception technologies such as aspect-orientation, CGLib, dynamic proxy etc.
`Invoker`: is an entity with the ability to mimic the operation performed by the subject on a protected resource. In the Esfinge Guardian framework, this element can execute methods; however, it is important to note that is just one of the possibilities since the architectural model is general.
`Populator`: It is the entity that contains the data extraction logic for authorization. Information for authorization can be anywhere such as databases, files, shared variables, user session, arguments, Internet, etc. For this reason, Populator is an entity that knows how to obtain information from all these places.
`PopulatorProcessor`: Entity that gathers and executes all defined Populators in the application.
`Authorizer`: Entity that implements the logic of the access control policy and may use information stored in AuthorizationContext if necessary. There must be at least one Authorizer. Every Authorizer must provide its response to the AuthorizationProcessor, usually a "yes" or "no"; however, it must be possible to include other response types such as "Indeterminate".
`AuthorizerProcessor`: Entity that contains the combining algorithm for all the Authorizers defined in the application.
`AuthorizationMetada`: Entity that indicates which resources or their operations must be intercepted by the authorization mechanism. A requirement is that this element must be of metadata type, so that it can be used declaratively. Esfinge Guardian uses Java annotations as the implementation of this element; however, it can be considered a general marking element that is independent of a specific technology.

4 CASE EXAMPLE

```
public class HierarchyAuthorizer
 implements Authorizer<RespectHierarchy> {

  public Boolean authorize(
                  AuthorizationContext c,
                    RespectHierarchy rh) {

  Set<String> roles =
      ctx.subject("roles");
  //retrieve other relevant information
      from ctx
  return //hierarchy authorization logic;
  }
}
```

5 DISCUSSION

Here goes some discussion.

6 RELATED WORK

A bit about others work.

7 CONCLUSION

Here goes a conclusion.

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

REFERENCES

[1] David Banisar. 2011. Data Protection Laws Around the World Map. *SSRN Electronic Journal* (2011). https://doi.org/10.2139/ssrn.1951416
[2] E M Guerra, J O Silva, and C T Fernandes. 2015. A Modularity and Extensibility Analysis on Authorization Frameworks. 2, 1 (2015).
[3] Axel Polleres, Maulik R Kamdar, Javier David Fernandez Garcia, Tania Tudorache, and Mark A Musen. 2018. A more decentralized vision for linked data. (2018).
[4] J.O. Silva, E.M. Guerra, and C.T. Fernandes. 2013. An extensible and decoupled architectural model for authorization frameworks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7974 LNCS. https://doi.org/10.1007/978-3-642-39649-6-44

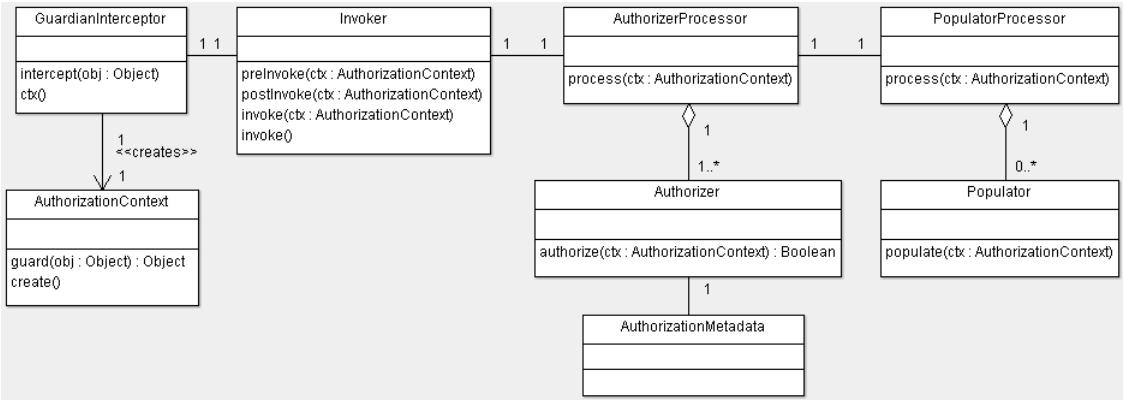

Figure 5: Esfinge Guardian class diagram [4]