

BACHELOR THESIS
COMPUTING SCIENCE



RADBOD UNIVERSITY

Using IRMA as an authentication
method for eduVPN

Author:

Wouter van Battum
s1011825

First supervisor/assessor:

prof. dr. B.P.F. Jacobs (Bart)
b.jacobs@cs.ru.nl

External supervisors:

R. Spoor (Rogier)
rogier.spoor@SURF.nl
F. Kooman (François)
f.kooman@tuxed.net

Second assessor:

Botros (Leon)
leon.botros@ru.nl

January 27, 2021

Abstract

SURF, the collaborative organisation for ICT in Dutch education and research, wants to make their VPN service, eduVPN, available for a wider audience. Currently, institutions log into the service by using the authentication method of SURF, which is only available to students, researchers and institutions' employees. Therefore, a prototype for an authentication method is created using the open source and privacy friendly project called IRMA.

At first the programming language that will be used for the prototype will be decided. Second, the front-end and back-end from IRMA will be integrated into eduVPN. Third, the servers and reverse proxy will be configured such that the servers can communicate securely. These steps leads to the development of the prototype in which users can log into the VPN server using their email address attribute from IRMA.

Contents

1	Introduction	3
1.1	IRMA	3
1.2	My Contribution	3
2	Preliminaries	6
2.1	IRMA terminology	6
2.1.1	Attribute	6
2.1.2	Requestor token	6
2.2	Browser related	6
2.2.1	401 Unauthorized Error	6
2.2.2	Content Security Policy (CSP)	6
2.2.3	REST API	7
2.3	Proxy server	7
2.3.1	Reverse proxy	7
3	Programming Language	8
3.1	Front-end	8
3.2	Back-end	8
4	Front-end integration	9
4.1	Integration	9
4.2	Authentication	9
5	Back-end integration	11
5.1	Authentication	11
6	Configuration of the servers	13
6.1	IRMA server	13
6.2	eduVPN	14
6.3	Reverse proxy	14
6.4	Overall configuration	15

7	Attribute disclosure	16
7.1	Pseudo-anonymity	16
7.2	Attribute	16
7.2.1	Alter the existing attribute	16
7.2.2	Creating a new attribute	17
8	Limitations	18
8.1	Trade-offs	18
8.1.1	Popup window or integrated element	18
8.1.2	Requestor token or JWT	18
8.1.3	Verifying the session result	19
8.2	Encountered problems	19
8.2.1	Defer attribute	19
8.2.2	Loading JWT in the front-end	19
8.2.3	Starting the session in the front-end	20
8.2.4	Session token is in the browser	20
8.2.5	Configuring the reverse proxy	21
9	Conclusions	22
9.1	Sub-questions	22
9.2	Main research question	22
10	Future work	24
10.1	Attribute	24
10.2	Email address	24
10.3	Keep the IRMA server running	24
11	Acknowledgements	25
A	Front-end code	28
A.1	irmaAuthentication.php	28
A.2	irma_impl.js	29
B	eduVPN configuration	30
C	Back-end code	35
D	IRMA server configuration	40
E	Reverse proxy configuration	41
F	IRMA Markdown file	43

Chapter 1

Introduction

Nowadays, privacy is getting more and more attention everywhere. It is becoming more important to get a secure internet connection. With a VPN you already shield yourself from eavesdroppers in your network. Currently, there are a lot of providers for a VPN service. However, SURF¹ currently provides a VPN service for students, researchers and employees of institutions, this service is called eduVPN². However, SURF wants to expand their reach and make eduVPN available for everyone. In order to do so they need a privacy friendly authentication method. The Privacy by Design Foundation³ came up with a solution: IRMA.

1.1 IRMA

IRMA (I Reveal My Attributes) is an open-source project from the Privacy by Design Foundation. IRMA lets you authenticate yourself using “attributes”, revealing only the necessary information [1].

An attribute contains certain types of information about the user. For instance, there is an attribute “Over 18” which you can only obtain when you are over eighteen. These attributes enable you to only disclose certain information about yourself and thereby not unnecessarily releasing sensitive information [2].

1.2 My Contribution

Currently, IRMA is written in JavaScript, Go and Node.js. However, eduVPN is written in PHP. Therefore, this research aims to investigate how to securely integrate IRMA as an authentication method for eduVPN. Thereby, developing a prototype in which IRMA is integrated into eduVPN.

¹<https://www.surf.nl/en/about-surf>

²<https://www.surf.nl/en/eduvpn/about-eduvpn?dst=n1173>

³<https://privacybydesign.foundation/en/>

This research is divided in the following sub-questions:

- (1) In which programming language should the prototype be written?
- (2) How can the front-end be integrated with IRMA?
- (3) How can the back-end be integrated with IRMA?
- (4) How should the servers be configured such they can communicate securely?
- (5) Which attribute should eduVPN ask for during the authentication?

The first sub-question will aid this research, because IRMA and eduVPN are written in different languages. Therefore, a well-considered choice has to be made. The second sub-question will aid the research, because for the front-end, decisions have to be made on which information should be available in the browser and which information should be kept away from the browser? The third sub-question will aid the research, because in the back-end decisions have to be made about which information to request from the IRMA server and which retrieved information should be verified. The fourth question is of importance to this research, because if you have integrated IRMA into eduVPN, how should the servers of IRMA and eduVPN and the reverse proxy be configured such that they can communicate securely with each other? The fifth sub-question will aid this research, because the Privacy by Design foundation provides some attributes that everyone can acquire. However, the question is which attribute should be chosen or should they create their own attribute? And why?

After answering these sub-questions, we will be able to draw a conclusion to the main research question. Furthermore, the prototype that is developed during this research is based on the answers of these questions.

Outline

The outline of this paper is as follows, in chapter 2 the terminology in order to understand this research is explained. Chapter 3 will discuss which programming language should be used for the prototype. In chapter 4, is discussed what happens in the front-end during a session and how IRMA's front-end is integrated in eduVPN's front-end. Chapter 5 will discuss through the back-end code. In chapter 6, will be discussed how to configure the IRMA server, eduVPN server and the reverse proxy. Chapter 7 discusses which attribute should be used when the prototype is taken to production. In chapter 8 will the problems that were encountered during the research be discussed and how they have been solved. Furthermore, in this chapter will be discussed which trade-offs were made in the process of developing the prototype. Chapter 9 will contain the conclusions that can

be drawn from the research and sub-questions. The steps that need to be taken in order to be able to use the prototype will be discussed in chapter 10. In chapter 11 the people who aided the research will be thanked and the division of work will be elaborated.

Chapter 2

Preliminaries

In the previous chapter the problem of eduVPN is defined and discussed. In this chapter the necessary terminology in order to understand the research is explained.

2.1 IRMA terminology

2.1.1 Attribute

According to the definition of [3] an attribute is “A small piece of data, generally containing a statement about the attribute owner”. This could be for instance that the owner of that the attribute is over eighteen.

2.1.2 Requestor token

The requestor token is a string that is specified in the configuration file from the IRMA server. If the IRMA server is setup such that a token must be provided, only applications that have such a token can communicate with the IRMA server.

2.2 Browser related

2.2.1 401 Unauthorized Error

According to the MDN contributors [4] the 401 error response code means that the request has not been applied because the request did not contain the correct credentials for the target source.

2.2.2 Content Security Policy (CSP)

According to the MDN contributors [5] the CSP is an added layer of security on top of the regular security a browser provides, such that attacks like Cross

Site Scripting and data injection attacks are mitigated. Therefore protects your server from data theft to the distribution of malware.

2.2.3 REST API

Red Hat [6] states that a REST API is an API that conforms to the constraints of REST, where REST is a set of architectural constraints to be used for creating web services. REST allows the requester to access and manipulate textual representations of resources by using predefined operations [7].

2.3 Proxy server

2.3.1 Reverse proxy

According to the definition from NGINX [8] a reverse proxy is a type of proxy server that typically sits behind the firewall in a private network and directs client requests to the appropriate back-end server. Furthermore, it provides extra security. Because for instance the proxy can close all the ports except 433 if you only want to communicate over HTTPS.

Chapter 3

Programming Language

IRMA and eduVPN are written in different languages. Therefore, a decision has to be made in which language we are going to write the prototype. This problem will be discussed in this chapter.

3.1 Front-end

For the front-end, there had to be chosen between PHP and JavaScript. We chose for a JavaScript inside a PHP file so we could have the best possible collaboration. This was easily done, as we could insert a JavaScript script inside PHP through: `<script type= "javascript">`.

3.2 Back-end

For the back-end, the decision was less trivial. Because, the back-end code for IRMA is also written in JavaScript and contains helpers that can be used in the handling of the sessions. Although, the JavaScript from IRMA has those functions, we have chosen to use PHP as the programming language for the back-end. Such that the communication with the rest of the eduVPN server did not become over complicated. Furthermore, the IRMA server has a REST API¹ which could be used for session handling. For the back-end we have chosen to use PHP. This is because eduVPN is written in PHP.

¹<https://irma.app/docs/api-irma-server/>

Chapter 4

Front-end integration

In the previous chapters the problem is explained, the preliminaries are discussed and the programming language that will be used are discussed. In this chapter we are going to elaborate the front-end code.

4.1 Integration

For the front-end we had to choose between a popup or an integrated form from IRMA. We have chosen for a popup because this would let us keep the style sheet and layout from eduVPN. This also meant that we did not need to integrate the style sheet of IRMA in the style sheet of eduVPN.

For functionality we had to include an IRMA JavaScript package named “irma.js”. The package provides the functionality of generating the popup window that shows the QR-code, which is used for the authentication. The layout is shown in Figure 4.1.

In order to comply to a strict Content Security Policy in production, we had to put the front-end code in the `irma.impl.js` file. This file is can be found in Appendix A.

4.2 Authentication

For the authentication there are multiple ways in which the front-end can be configured [9]. Because the sensitive data needs to be kept outside of the browser, we have chosen to configure the IRMA plugin in such a way that it only shows the QR-code.

The authentication process starts directly when the login page is opened. The QR-code that is shown is build by the IRMA plugin and is based on the session pointer it gets from the back-end. If the QR-code is scanned, the popup window waits for the user to authenticate themselves. If the user has shared their attribute in the IRMA app, the popup window closes.

Otherwise, the popup window shows an error message saying that something went wrong. If the authentication succeeded, the front-end submits an empty form to the back-end as a sign that the session result can be fetched from the IRMA server.

The complete code from the front-end can be found in Appendix A.

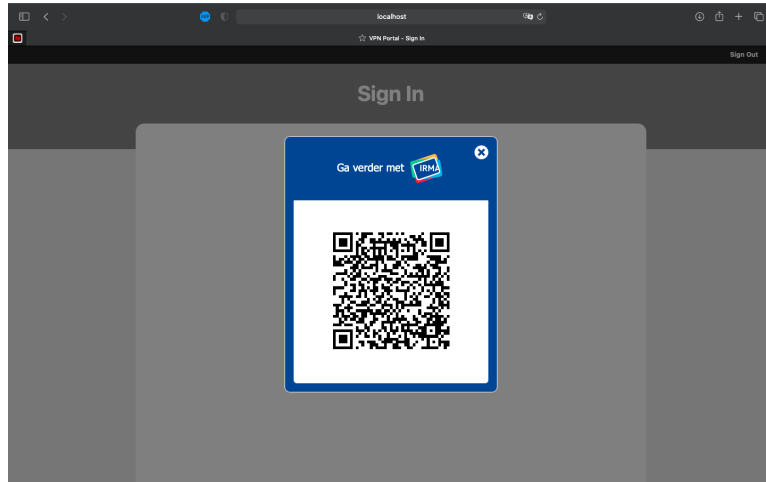


Figure 4.1: Layout of the front-end

Chapter 5

Back-end integration

In the previous chapters the front-end is explained and the necessary information about the problem is discussed. In this chapter, the back-end code will be elaborated.

5.1 Authentication

The variables that are used for starting the verification are `userIdAttribute` and `secretToken`. These variables are specified in the configuration file `config.php` file, which is in Appendix B. The location on where to find the IRMA server is saved as a default value in the eduVPN configuration and therefore does not need to be specified here.

At first, the back-end sends a POST request [10] with the attribute that needs to be disclosed by the user. This POST request has an Authorization header that contains the token which the eduVPN server uses to identify itself at the IRMA server. At this point the IRMA server creates a session and responds with the session pointer and session token. From this response the variable `sessionPtr` is extracted at the back-end and sent to the front-end for the generation of the QR-code. The value of the variable `token` is stored in the current session. The token is not sent to the front-end, because then an adversary is not able to extract the session token from the network traffic. Therefore, the adversary cannot communicate with the IRMA server about the current session.

If the authorisation succeeded, the form in the front-end is submitted and that gives the back-end the signal to send a GET request to the IRMA server asking for the result [10]. After the back-end server has gotten the response from the IRMA server it checks whether the `proofStatus` is present in the response. If it is present the back-end server checks whether `proofStatus` is “VALID”. If there is no `proofStatus` or it is not “VALID”, a 401 error is thrown. Furthermore, the back-end checks whether the status of the session is finished, this means that the value of `status` is “DONE”. At last,

the back-end checks if the session ended without any errors. Therefore, it checks for the presence of the field **error**, if that field is not present the session ran without errors. If the results from the verification checks are correct, the back-end server sets the variable **userIdAttribute** to the value of **rawvalue**. The variable **rawvalue** is the value which the user identified itself with, in the case of the prototype this is an email address. If the back-end server is not able to extract **rawvalue** from the response, then the server will throw a 401 error.

If the variable **userIdAttribute** is set to **rawvalue**, the browser is redirected to the VPN homepage. On the homepage under the section “Account” the email address that is used for the authentication can be found at “userId”.

The complete code from the back-end can be found in Appendix C.

Chapter 6

Configuration of the servers

Now that we know from the previous chapters what the front-end and back-end does, we will look elaborate in this chapter how the IRMA server, eduVPN server and reverse proxy should be configured.

6.1 IRMA server

In order to configure the IRMA server such that it communicates securely with the eduVPN server, the option `production` needed to be set to `true`. This meant that the defaults for the configuration options would be stricter [11]. Because we wanted to restrict the access to the IRMA server as much as possible, the eduVPN server needs to identify itself with a requestor token. Furthermore, setting the `production` variable to `true`, meant that we either specify an email address or set the variable `no_email` to `true`. We chose to set `no_email` to `true`, because the prototype is not used for production and therefore we do need the updates about changes in the IRMA software or ecosystem. However, in production the email address should be specified.

What also is specified in the configuration file is the URL on which the IRMA app can find the IRMA server. The address and port on which the eduVPN server can find the IRMA server is specified in the `listen_addr` and `port` variables. For security reasons the server only allows users to disclose their email address attribute. Such that if for example an adversary does not have a valid email address but has a different valid attribute, the adversary is not able authenticate themselves using the other attribute.

For the prototype is chosen to use the email address as the attribute that needs to be disclosed in order to be granted access to the VPN service. An example IRMA server session log is shown in Figure 6.1 (for privacy reasons the sensitive data is deducted).

The IRMA configuration file can be found in Appendix D.

6.2 eduVPN

Before the eduVPN environment is ready to run with IRMA as the authentication method, the configuration needs to be altered. This means altering the `/etc/vpn-user-portal/config.php` file. This is done by changing the authentication method to `IrmaAuthentication` and by specifying the `IrmaAuthentication` section such that the server knows which attribute should be disclosed and what the requestor token is. The requestor token must match the token in the IRMA server configuration. The location on which the eduVPN server can reach the IRMA server is not specified. This is because the default has been set to the location specified in the IRMA configuration. These changes lead to the following configuration:

```
// ...

'authMethod' => 'IrmaAuthentication',

// ...

'IrmaAuthentication' => [
    'userIdAttribute' => 'pbdn.sidn-pbdn.email.email',
    'secretToken' => 'dz00SwTqr0tJxpH7uJ9GL0PZMf30CELF',
],
```

The entire configuration file can be found in Appendix B.

For the prototype, the attribute that needs to be disclosed is the email address attribute. This attribute is only chosen for proof of concept, a more privacy-friendly attribute would be for instance the `over18` attribute. In section 9.2 is elaborated what attribute should be used if the prototype is taken to production.

An example eduVPN server session log is shown in Figure 6.2 (for privacy reasons the sensitive data is deducted).

6.3 Reverse proxy

Because the IRMA and eduVPN server run behind a reverse proxy, the reverse proxy also needed to be configured, such that it would properly handles the sessions. Only one line needed to be added to the `<VirtualHost *:443>` section:

```
ProxyPass "/irma/" "http://localhost:8088/irma/"
```

Because the requests from the eduVPN server to the IRMA server and vice versa are inside an internal network behind the reverse proxy, there is no need to use HTTPS. The entire reverse proxy configuration can be found in Appendix E.

6.4 Overall configuration

An organized overview is created to increase readability for users that want to use IRMA as the authentication method in the eduVPN environment. This overview is written as a Markdown file. It can be found in Appendix F.

[illegible]

Figure 6.1: IRMA server log

```
% sh -l/launch.sh
% PHP 7.3.24-(to be removed in future macOS) Development Server started at Fri Jan 8 14:29:39 2021
Listening on http://localhost:8082
Document root is /Users/wouter/Projects/LC-v2/vpn-user-portal/web
Press Ctrl-C to quit.
PHP 7.3.24-(to be removed in future macOS) Development Server started at Fri Jan 8 14:29:39 2021
Listening on http://localhost:8008
Document root is /Users/wouter/Projects/LC-v2/vpn-server-api/web
Press Ctrl-C to quit.
[Fri Jan 8 14:29:44 2021] 127.0.0.1:53837 [200]: /
[Fri Jan 8 14:29:45 2021] 127.0.0.1:53848 [200]: /css/screen.css?mtime=1609355076
[Fri Jan 8 14:29:45 2021] 127.0.0.1:53852 [200]: /js/irma_impl.js?mtime=1609515692
[Fri Jan 8 14:29:45 2021] 127.0.0.1:53853 [200]: /js/irma.js?mtime=1609356592
[Fri Jan 8 14:30:28 2021] 127.0.0.1:54279 [302]: /_irma/verify
[Fri Jan 8 14:30:31 2021] 127.0.0.1:54291 [200]: /api.php/is_disabled_user?user_id=
[Fri Jan 8 14:30:31 2021] 127.0.0.1:54312 [200]: /api.php/user_update_session_info
[Fri Jan 8 14:30:31 2021] 127.0.0.1:54285 [302]: /
[Fri Jan 8 14:30:31 2021] 127.0.0.1:54316 [200]: /api.php/is_disabled_user?user_id=
[Fri Jan 8 14:30:31 2021] 127.0.0.1:54317 [200]: /api.php/system_messages?message_type=modt
[Fri Jan 8 14:30:32 2021] 127.0.0.1:54345 [200]: /home
[Fri Jan 8 14:30:32 2021] 127.0.0.1:54322 [200]: /css/screen.css?mtime=1609355076
[Fri Jan 8 14:30:32 2021] 127.0.0.1:54326 [404]: /favicon.ico - No such file or directory
[Fri Jan 8 14:30:50 2021] 127.0.0.1:54503 [302]: /logout
[Fri Jan 8 14:30:51 2021] 127.0.0.1:54507 [200]: /home
[Fri Jan 8 14:30:51 2021] 127.0.0.1:54514 [200]: /js/irma.js?mtime=1609356592
[Fri Jan 8 14:30:51 2021] 127.0.0.1:54515 [200]: /css/screen.css?mtime=1609355076
[Fri Jan 8 14:30:51 2021] 127.0.0.1:54516 [200]: /js/irma_impl.js?mtime=1609515692
[Fri Jan 8 14:31:03 2021] 127.0.0.1:54625 [302]: /_irma/verify
[Fri Jan 8 14:31:04 2021] 127.0.0.1:54633 [200]: /api.php/is_disabled_user?user_id=
[Fri Jan 8 14:31:04 2021] 127.0.0.1:54634 [200]: /api.php/user_update_session_info
[Fri Jan 8 14:31:04 2021] 127.0.0.1:54632 [302]: /
[Fri Jan 8 14:31:04 2021] 127.0.0.1:54644 [200]: /api.php/is_disabled_user?user_id=
[Fri Jan 8 14:31:04 2021] 127.0.0.1:54642 [200]: /api.php/system_messages?message_type=modt
[Fri Jan 8 14:31:04 2021] 127.0.0.1:54636 [200]: /home
[Fri Jan 8 14:31:04 2021] 127.0.0.1:54644 [200]: /css/screen.css?mtime=1609355076
```

Figure 6.2: eduVPN server log

Chapter 7

Attribute disclosure

In the previous chapters the development of the prototype is discussed and what the prototype entailed. In this chapter the attribute that should be used in production is discussed.

7.1 Pseudo-anonymity

Currently, the prototype uses the email address as the attribute that needs to be disclosed by the user. However, this is not exactly privacy friendly. Therefore, in order to increase anonymity the user should be identified with a numerical ID or a pseudonym. This means that we would achieve pseudo-anonymity. For users of course the most optimal solution to achieve full anonymity. However, that means that when the user is online, they can do whatever they want because it cannot be linked back to them. For SURF, pseudo-anonymity would be the best option, because users can still be anonymous online but are not anonymous for SURF. Therefore, when an user would violate the user agreement or performs illegal actions online, the user can still be identified and penalised.

7.2 Attribute

7.2.1 Alter the existing attribute

SURF already has an attribute in the IRMA database. Currently, this attribute is only available to students, researchers and institutions' employees. The attribute contains the following information [12]:

- (1) The institute that provided the attribute.
- (2) The position of the user at the institution.
- (3) The ID number of the user at the institution.

- (4) The full name of the user as registered by the institution.
- (5) The front name of the user as registered by the institution.
- (6) The family name of the user as registered by the institution.
- (7) The email address of the user at the institution

In order to achieve pseudo-anonymity this attribute needs to be altered such that is available to everyone. This can be done by for instance leaving (1) empty, setting (2) to for instance “public” and setting (3) to the membership ID. Furthermore, (4)-(7) should be provided by another IRMA attribute such that the information that is provided can be trusted.

7.2.2 Creating a new attribute

The second option is to create a new attribute containing the information that is needed by SURF. In order to achieve pseudo-anonymity this attribute would also needs some sort of membership ID which can be related to a single user. The user information in this attribute should also be provided by IRMA, such that the information is trustworthy. There is a side note, because creating a new attribute is not done for free, so this option comes with additional costs [13].

Chapter 8

Limitations

In the previous chapters the prototype is explained and discussed. In this chapter we will review the process that got us to the prototype as it currently is. In particular, the trade-offs we had to make and the problems we encountered will be discussed.

8.1 Trade-offs

8.1.1 Popup window or integrated element

When the `irma.js` file is used, there is a choice of using an integrated form in your web page or a popup window. We chose for a popup window, because this meant we did not had to integrate the IRMA style sheet into the eduVPN style sheet. Currently this choice is out convenience, such that we did not need to go through the IRMA style sheet and extract the necessary elements. The only disadvantage occurs when you want to add more buttons or elements to the web page. Then, the user would need to close the IRMA popup window and refresh the page in order to get another popup window. The popup window could also be shown when clicked on a button. However, because currently the users only need the popup window, we chose not to use a button.

8.1.2 Requestor token or JWT

The second trade-off we had to make was that we had to choose between authenticating via a requestor token or via JWT. After a problem with using JWT in the browser we decided to use the requestor token instead. If this token is kept secret, and nobody bypasses the reverse proxy, no one else can access the IRMA server. However, if this token is leaked, adversaries can only start a new session on the IRMA server and disclose the email attribute. This is, because the IRMA server is configured such that it only

allows the requestor to disclose the email address attribute. Furthermore, Mr Kooman is not a fan of JWT. Therefore it was an easy decision.

8.1.3 Verifying the session result

The `irma.js` file provides the functionality of checking if the attribute that the user disclosed is valid. However, if the REST API is used it is also possible to let the back-end handle the verification. We have chosen for the latter, because this meant that we could also keep the attribute that the user disclosed in the back-end. Therefore we have chosen to verify the results ourselves in the back-end. The request that the back-end sends to the IRMA server and the response of the IRMA server to the back-end are shown in Figure 7.1 (for privacy reasons sensitive data is deducted).

```
[2021-01-08T13:31:03Z] TRACE => request from= headers=map[Accept:[*/]] method
=GET proto=HTTP/1.1 type=requestor url=/session/qltNoM4QQNyw42cTTG2m/result
[2021-01-08T13:31:03Z] TRACE <= response duration=359.643us response={"token":"qltNoM4QQNyw42cTTG
2m","status":"DONE","type":"disclosing","proofStatus":"VALID","disclosed":[{"rawvalue":"
","value":{"":"","en":"","n
1":"","id":"pddf.pddf.email.email","status":"PRESENT","issuancetime":1
600905600}}]} status=200
```

Figure 8.1: The request for the result and the response

8.2 Encountered problems

8.2.1 Defer attribute

Problem

The first problem that we encountered was that that we got the error from the browser that the variable `irma` could not be found. This was because at first the original JavaScript form had the defer option enabled at the import of the `irma.js` file. This meant that the `irma.js` file had not been loaded yet but the JavaScript already wanted to access the variable `irma`, which is defined in `irma.js`.

Solution

The solution to this problem was neither deferring the import of `irma.js` nor the JavaScript code that handles the front-end.

8.2.2 Loading JWT in the front-end

Problem

The second problem occurred when we tried to sign the session requests using JWT. This could be done by including a `Node.js` package made by

the Privacy by Design Foundation. This would mean that we needed the `require()` function in our JavaScript. However, a browser cannot load `Node.js` modules.

Solution

As a solution we decide to let go of the JWT and use the requestor token instead. This meant we needed to include an “Authorization header” in the POST-request to the IRMA server when the authentication session would be initiated.

8.2.3 Starting the session in the front-end

Problem

At first, we tried to handle the entire session in the front-end by using the functionality from `irma.js`. However, this meant that we could not extract the session token. This session token is needed to verify the session result in the back-end.

Solution

The solution was using the IRMA REST API, which meant that we needed to perform a POST request in the front-end and extract the session token from the response. Then, the session token needed to be sent to the back-end, and the session pointer to the IRMA plugin.

8.2.4 Session token is in the browser

Problem

From the previous problem another problem arose. Starting the session in the front-end meant that the session token and the secret token from the authorization header would be visible in the browser. This is of course far from desirable because if an adversary got hold of the secret token. The adversary would then be able to start sessions to our IRMA server himself. Otherwise, if the adversary got a hold on the session token, he could for instance delete the session before it was finished or we could verify the result.

Solution

The solution to this problem was starting the session from the back-end. This meant that no sensitive data would be visible if an adversary would sniff the network traffic between the front-end and the reverse proxy.

8.2.5 Configuring the reverse proxy

Problem

When configuring the reverse proxy we ran into the problem that when we started a session we could see a QR-code in the web browser. However, it disappeared after a second and returned an error.

Solution

The solution was that we needed to add backslashes to the reverse proxy configuration. After we did this, it worked but another problem arose.

Problem

The second problem with the reverse proxy was that once we created an IRMA session, and the browser asked for the status of the session at the IRMA server, that the IRMA server said it did not know the session and therefore returned an error, a bad request error. The underlying problem was that the reverse proxy stripped of a part of the URL: `"/irma/"`.

Solution

In order to solve this problem we had to extend the URL in the reverse proxy configuration such that the requests were redirected to the correct endpoint. This meant appending the `"/irma/"` section to the URL. We went from `"ProxyPass '/irma/' http://localhost:8088/"` to `"ProxyPass '/irma/' http://localhost:8088/irma/"`. This solved the issue.

Chapter 9

Conclusions

In the previous chapters, the problem is described, the preliminaries are explained, the sub-questions have been answered, and the steps that need to be taken to implement an IRMA server for eduVPN have been described. In this chapter, the conclusion that can be based on the previous chapters is drawn.

9.1 Sub-questions

The conclusions that can be drawn from the sub-questions are as follows:

- (1) The front-end can be written in your preferred programming language. However, it needs to have a JavaScript block in the front-end to use the functionality provided by the IRMA plugin.
- (2) The front-end can be integrated with IRMA by using the functionality of the `irma.js` file.
- (3) The back-end can be integrated with IRMA by using the REST API functionality to construct the correct requests to perform the session.
- (4) The server should be configured behind a reverse proxy such that the traffic between the servers is hidden from the outside.
- (5) The attribute that should be asked for during the authentication should be existing attribute from SURF, with more options such the everyone can obtain the attribute.

9.2 Main research question

From the sub-questions and research we can conclude that in order to integrate IRMA into eduVPN securely, you need to follow the steps shown in this research. However, then it would not be privacy friendly because the

email address is used as the attribute that needs to be disclosed. The development of the attribute is essential in adapting the prototype for privacy friendly production usage.

Chapter 10

Future work

In the previous chapters the the prototype is explained and discussed. However, in order to take this prototype into production, a few adaptations have to be made. These adaptations are for further research and work. Therefore, these adaptations will be discussed in this chapter.

10.1 Attribute

For testing purposes the email address attribute is used. However, for production purposes the attribute discussed in chapter 9 should be developed and used.

10.2 Email address

When the prototype is adapted for production, the IRMA server configuration should contain an email address of one of the main developers. This email address will be stored by the Privacy by Design Foundation. This email address is only used to notify the developers about changes in the IRMA software or ecosystem. This is important because if those changes are not implemented or the IRMA server is not adapted, the IRMA server could become no longer compatible with the rest of the IRMA ecosystem. [14].

10.3 Keep the IRMA server running

Currently, the IRMA server of the prototype runs on a virtual machine and I start the server when I connect to the virtual machine. This means that the IRMA server does not always run and the prototype is not always available. For the production the IRMA server should be hosted somewhere where it is permanently hosted.

Chapter 11

Acknowledgements

This research and prototype could not have been possible without the support of one of my supervisors, François Kooman. He helped me by writing a large part of the back-end code. He wrote the back-end based on what I had researched about what the back-end should do, for instance the POST request to the IRMA server and which values to retrieve from the responses. He did this because he is much more sophisticated with programming in PHP than me. So overall, we discussed what needed to be done and then he wrote the back-end code, such that it looked and worked at its best. I wrote the front-end and Kooman extended it such that it would comply to the strict CSP setting. Furthermore, Kooman improved the IRMA Markdown file I wrote such that it contained less text and was adapted to the final updates.

Bibliography

- [1] Privacy by Design Foundation, *Privacy Policy*, 2020. [Online]. Available: <https://privacybydesign.foundation/privacy-policy-en/#top-of-page>.
- [2] —, *IRMA in detail*. [Online]. Available: <https://privacybydesign.foundation/irma-explanation/>.
- [3] —, *Technical overview*, 2020. [Online]. Available: <https://irma.app/docs/overview/>.
- [4] MDN Contributors, *401 Unauthorized*, 2020. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/401>.
- [5] —, *Content Security Policy (CSP)*, 2020. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>.
- [6] Red Hat Inc., *What is a REST API*, 2021. [Online]. Available: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>.
- [7] Wikipedia Contributors, *Representational state transfer*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer.
- [8] NGINX developers, *What Is a Reverse Proxy Server?* n.d. [Online]. Available: <https://www.nginx.com/resources/glossary/reverse-proxy-server/>.
- [9] I. Derksen, *IRMA client*, 2020. [Online]. Available: <https://github.com/privacybydesign/irma-frontend-packages/blob/6a8899c855da6d82f40b38b0f5088462d385084c/plugins/irma-client/README.md>.
- [10] Privacy by Design Foundation, *IRMA API server*, 2020. [Online]. Available: <https://irma.app/docs/api-irma-server/>.
- [11] —, *irma server*, 2020. [Online]. Available: <https://irma.app/docs/irma-server/#production-mode>.
- [12] —, *IRMA attributes*, n.d. [Online]. Available: <https://privacybydesign.foundation/attribute-index/en/pbdf.pbdf.surfnet.html>.

- [13] —, *Integrate IRMA in your website*, n.d. [Online]. Available: <https://privacybydesign.foundation/irma-verifier/#issue>.
- [14] —, *Email address*, 2020.
[Online]. Available: <https://irma.app/docs/email/>.

Appendix A

Front-end code

A.1 irmaAuthentication.php

```
1 <?php $this->layout('base', ['pageTitle' => $this->t('Sign In  
  ')]); ?>  
2 <?php $this->start('content'); ?>  
3 <!--  
4     irma.js obtained from https://gitlab.science.ru.nl/irma/  
      github-mirrors/irma-frontend-packages/-/jobs/111202/  
      artifacts/browse/irma-frontend/dist  
5     @see https://github.com/privacybydesign/  
      irma-frontend-packages/tree/master/irma-frontend  
6 -->  
7 <script src="<?php echo $this->getAssetUrl($requestRoot, 'js/  
  irma.js'); ?>"></script>  
8 <script src="<?php echo $this->getAssetUrl($requestRoot, 'js/  
  irma_impl.js'); ?>"></script>  
9 <!--  
10     the _irma/verify endpoint is triggered after the  
      attribute release with the  
11     IRMA app is complete. This is only used to inform the  
      backend that the  
12     IRMA server needs to be queried to obtain the  
      attribute...  
13 -->  
14 <div id="irmaAuth" data-session-ptr="<?php echo $this->e(  
  $sessionPtr); ?>">  
15     <form method="post" action="<?php echo $requestRoot; ?>  
      _irma/verify">  
16     </form>  
17 </div>  
18 <?php $this->stop('content'); ?>
```

A.2 irma_impl.js

```
1  "use strict";
2
3  document.addEventListener("DOMContentLoaded", function() {
4      const sessionPtr = document.getElementById("irmaAuth").
        dataset.sessionPtr;
5      const irmaFrontend = irma.newPopup({
6          debugging: false,
7
8          session: {
9              start: false,
10             mapping: {
11                 sessionPtr: function() {
12                     return JSON.parse(sessionPtr);
13                 }
14             },
15             result: false
16         }
17     });
18
19     irmaFrontend.start().then(function(response) {
20         document.querySelector("div#irmaAuth form").submit();
21     }).catch(function(error) {
22         console.error("Couldn't do what you asked", error);
23     });
24
25 });
```

Appendix B

eduVPN configuration

```
1 <?php
2
3 return [
4     // override default branding style (templates/CSS) with
      custom style.
5     // NOTE: the styling/branding MUST be installed for this
      to work!
6     //'styleName' => 'eduVPN',
7     //'styleName' => 'LC',
8
9     //'authMethod' => 'FormPdoAuthentication',          // PDO
      (database)
10    //'authMethod' => 'FormLdapAuthentication',          // LDAP
11    //'authMethod' => 'FormRadiusAuthentication',        // RADIUS
12    //'authMethod' => 'SamlAuthentication',              // SAML (
      php-saml-sp)
13    //'authMethod' => 'MellonAuthentication',            // SAML (
      mod_auth_mellon)
14    //'authMethod' => 'ShibAuthentication',              // SAML (
      Shibboleth)
15    'authMethod' => 'IrmaAuthentication',
16
17    // Default Session Expiry
18    // The session expiry will be used to determine the "Not
      After" of the
19    // issued X.509 certificates and the moment at which to
      start rejecting
20    // the OAuth tokens.
21    'sessionExpiry' => 'P90D',                          // 90 days
22    //'sessionExpiry' => 'PT12H',                        // 12 hours
23    //'sessionExpiry' => 'P1D',                          // 1 day
24
25    // LDAP
26    'FormLdapAuthentication' => [
27        // *** OpenLDAP / FreeIPA ***
28        'ldapUri' => 'ldaps://ipa.example.org',
29        'bindDnTemplate' => 'uid={{UID}},cn=users,cn=accounts'
```



```

30         ,dc=example,dc=org',
31         //'permissionAttribute' => 'eduPersonEntitlement',
32         //'permissionAttribute' => 'memberOf',
33
34         // *** Active Directory ***
35         //'ldapUri' => 'ldap://ad.example.org',
36         //'bindDnTemplate' => 'DOMAIN\{{UID}}',
37         //'baseDn' => 'dc=example,dc=org',
38         //'userFilterTemplate' => '(sAMAccountName={{UID}})',
39         //'permissionAttribute' => 'memberOf',
40     ],
41
42     // RADIUS
43     'FormRadiusAuthentication' => [
44         'serverList' => [
45             [
46                 'host' => 'radius.example.org',
47                 'secret' => 'testing123',
48                 //'port' => 1812,
49             ],
50         ],
51         //'addRealm' => 'example.org',
52         //'nasIdentifier' => 'vpn.example.org',
53     ],
54
55     // SAML (php-saml-sp)
56     'SamlAuthentication' => [
57         // 'OID for eduPersonTargetedID
58         'userIdAttribute' => 'urn:oid
59             :1.3.6.1.4.1.5923.1.1.1.10',
60         // OID for eduPersonPrincipalName
61         //'userIdAttribute' => 'urn:oid
62             :1.3.6.1.4.1.5923.1.1.1.6',
63
64         // ** AUTHORIZATION | PERMISSIONS **
65         // OID for eduPersonEntitlement
66         //'permissionAttribute' => 'urn:oid
67             :1.3.6.1.4.1.5923.1.1.1.7',
68         // OID for eduPersonAffiliation
69         //'permissionAttribute' => 'urn:oid
70             :1.3.6.1.4.1.5923.1.1.1.1',
71
72         // override the SP entityId, the default is:
73         // https://vpn.example.org/vpn-user-portal/_saml/
74         // metadata
75         //'spEntityId' => 'https://vpn.example.org/saml',
76
77         // (Aggregate) SAML metadata file containing the IdP
78         // metadata of IdPs
79         // that are allowed to access this service
80         'idpMetadata' => '/path/to/idp/metadata.xml',
81
82         // set a fixed IdP for use with this service, it MUST
83         // be available in

```

```

76 // the IdP metadata file
77 'idpEntityId' => 'https://idp.example.org/saml',
78
79 // set a URL that performs IdP discovery, all IdPs
    listed in the
80 // discovery service MUST also be available in the
    IdP metadata file,
81 // NOTE: do NOT enable idpEntityId as it will take
    precedence over
82 // using discovery...
83 // 'discoUrl' => 'http://vpn.example.org/php-saml-ds/
    index.php',
84
85 // AuthnContext required for *all* users
86 // 'authnContext' => ['urn:oasis:names:tc:SAML:2.0:ac:
    classes:TimesyncToken'],
87
88 // Users with certain permissions obtained through
89 // "permissionAttribute" MUST also have ANY of the
    listed
90 // AuthnContexts. If they currently don't, a new
    authentication is
91 // triggered to obtain it
92 // 'permissionAuthnContext' => [
93 //     'urn:example:LC-admin' => ['urn:oasis:names:tc:
        SAML:2.0:ac:classes:TimesyncToken'],
94 // ],
95
96 // Allow for overriding global sessionExpiry based on
    SAML
97 // "permissionAttribute" value(s)
98 // 'permissionSessionExpiry' => [
99 //     'urn:example:LC-admin' => 'PT12H',
100 // ],
101 ],
102
103 // SAML (mod_auth_mellon)
104 'MellonAuthentication' => [
105 // OID for eduPersonTargetedId
106 'userIdAttribute' => 'MELLON_urn:oid:1
    _3_6_1_4_1_5923_1_1_1_10',
107 // OID for eduPersonPrincipalName
108 // 'userIdAttribute' => 'MELLON_urn:oid:1
    _3_6_1_4_1_5923_1_1_1_6',
109
110 // ** AUTHORIZATION | PERMISSIONS **
111 // OID for eduPersonEntitlement
112 // 'permissionAttribute' => 'MELLON_urn:oid:1
    _3_6_1_4_1_5923_1_1_1_7',
113 // OID for eduPersonAffiliation
114 // 'permissionAttribute' => 'MELLON_urn:oid:1
    _3_6_1_4_1_5923_1_1_1_1',
115 ],
116

```

```

117 // SAML (Shibboleth)
118 'ShibAuthentication' => [
119     'userIdAttribute' => 'persistent-id',
120     //'userIdAttribute' => 'eppn',
121
122     // ** AUTHORIZATION | PERMISSIONS **
123     //'permissionAttribute' => 'entitlement',
124     //'permissionAttribute' => 'affiliation',
125 ],
126
127 // the permission required to be able to access the "
128 // admin" portion of
129 // the portal, see "permissionAttribute" in the
130 // authentication
131 // configuration sections
132 //'adminPermissionList' => ['urn:example:LC-admin'],
133
134 // list of userIds that have access to the admin
135 'adminUserIdList' => ['admin'],
136
137 // Require Users to use 2FA
138 'requireTwoFactor' => false,
139 //'requireTwoFactor' => true,
140
141 // Available 2FA methods
142 //'twoFactorMethods' => ['totp'], // TOTP
143 'twoFactorMethods' => [], // 2FA disabled
144
145 // supported languages in the UI, the first one mentioned
146 // is the default
147 'supportedLanguages' => [
148     'en_US' => 'English',
149     //'nl_NL' => 'Nederlands',
150     //'nb_NO' => 'norsk bokmal',
151     //'da_DK' => 'Dansk',
152     //'fr_FR' => 'Francais',
153 ],
154
155 'Api' => [
156     'consumerList' => [
157         //'_CLIENT_ID_' => [
158         //     'redirect_uri_list' => [
159         //         '_REDIRECT_URI_1_',
160         //         '_REDIRECT_URI_2_',
161         //     ],
162         //     'display_name' => '_DISPLAY_NAME_',
163         //     'require_approval' => true,
164         //     'client_secret' => '_SECRET_',
165         // ],
166     ],
167
168     // Enable Remote Access, i.e. users from other VPN
169     // servers listed in
170     // the below remoteAccessList files to access this

```

```

167         VPN server through
168         // the OAuth API
169         'remoteAccess' => false,
170         'remoteAccessList' => [
171             'production' => [
172                 'discovery_url' => 'https://static.eduvpn.nl/
disco/secure_internet.json',
173                 'public_key' => 'E50n0JTtyUVZmcWd+I/
FXRm32nSq8R2ioyW7dcu/U88=',
174             ],
175             // 'development' => [
176                 // 'discovery_url' => 'https://static.eduvpn.
nl/disco/secure_internet_dev.json',
177                 // 'public_key' => '
zzls4TZTXHEyV3yxaxag1DZw3tSpIdBoaa0jUGH/Rwg
=',
178             // ],
179         ],
180
181         'IrmaAuthentication' => [
182             // Specify the URL to your (local) IRMA server.
183             // OPTIONAL, DEFAULT: http://localhost:8088
184             // 'irmaServerUrl' => 'http://localhost:8088',
185
186             // The attribute used for the user ID in the service
187             'userIdAttribute' => 'pbfdf.sidn-pbfdf.email.email',
188
189             // The token to talk to the session endpoint of the
IRMA server, make
190             // sure it matches the one configured in the IRMA
server config
191             'secretToken' => 'dz00SwTqr0tJxpH7uJ9GL0PZMf30CELF',
192         ],
193
194         // Connection to vpn-server-api
195         'apiUser' => 'vpn-user-portal',
196         'apiPass' => 'veZmnEea0WWlBFFtayWcax9TuldGDb4M',
197         'apiUri' => 'http://localhost/vpn-server-api/api.php',
198     ];

```

Appendix C

Back-end code

```
1 <?php
2
3 /*
4  * eduVPN - End-user friendly VPN.
5  *
6  * Copyright: 2016-2019, The Commons Conservancy eduVPN
7  *   Programme
8  * SPDX-License-Identifier: AGPL-3.0+
9  */
10 namespace LC\Portal;
11
12 use LC\Common\Config;
13 use LC\Common\Http\BeforeHookInterface;
14 use LC\Common\Http\Exception\HttpException;
15 use LC\Common\Http\RedirectResponse;
16 use LC\Common\Http\Request;
17 use LC\Common\Http\Response;
18 use LC\Common\Http\Service;
19 use LC\Common\Http\ServiceModuleInterface;
20 use LC\Common\Http\SessionInterface;
21 use LC\Common\Http\UserInfo;
22 use LC\Common\HttpClient\HttpClientInterface;
23 use LC\Common\Json;
24 use LC\Common\TplInterface;
25
26 class IrmaAuthentication implements ServiceModuleInterface,
27     BeforeHookInterface
28 {
29     /** @var \LC\Common\TplInterface */
30     protected $tpl;
31
32     /** @var SessionInterface */
33     private $session;
34
35     /** @var \LC\Common\HttpClient\HttpClientInterface */
36     private $httpClient;
```

```

36
37     /** @var \LC\Common\Config */
38     private $config;
39
40     public function __construct(SessionInterface $session,
41                                TplInterface $tpl, HttpClientInterface $httpClient,
42                                Config $config)
43     {
44         $this->session = $session;
45         $this->tpl = $tpl;
46         $this->httpClient = $httpClient;
47         $this->config = $config;
48     }
49
50     /**
51      * @return void
52      */
53     public function init(Service $service)
54     {
55         $service->post(
56             '/_irma/verify',
57             /**
58              * @return \LC\Common\Http\Response
59              */
60             function (Request $request) {
61                 if (null === $sessionToken = $this->session->
62                     get('_irma_auth_token')) {
63                     throw new HttpException('token not found
64                         in session', 400);
65                 }
66
67                 $irmaStatusUrl = sprintf('%s/session/%s/
68                     result', $this->config->requireString('
69                     irmaServerUrl'), $sessionToken);
70                 $httpResponse = $this->httpClient->get(
71                     $irmaStatusUrl, [], []);
72                 // @see https://irma.app/docs/api-irma-server
73                 // #get-session-token-result
74                 $jsonData = Json::decode($httpResponse->
75                     getBody());
76                 if (\array_key_exists('error', $jsonData)) {
77                     throw new HttpException('Error: '.
78                         $jsonData['error'], 401);
79                 }
80
81                 // the "proofStatus" key is only available
82                 // when the
83                 // authentication finished, here we make sure
84                 // it is 'VALID'
85                 if (!\array_key_exists('proofStatus',
86                     $jsonData)) {
87                     throw new HttpException('missing "
88                         proofStatus"', 401);
89                 }
90             }
91         );
92     }
93

```

```

76         if ('VALID' !== $jsonData['proofStatus']) {
77             throw new HttpException('"proofStatus"
78                 MUST be "VALID"', 401);
79         }
80         // the 'status' key is only 'DONE' when the
81         // authentication finished, here we make sure
82         // it is 'DONE'
83         if (!\array_key_exists('status', $jsonData)) {
84             throw new HttpException('missing "status"
85                 ', 401);
86         }
87         if ('DONE' !== $jsonData['status']) {
88             throw new HttpException('"status" MUST be
89                 "DONE"', 401);
90         }
91         // the 'error' key is only available when an
92         // error occurred
93         // here we make sure we continue without an
94         // error
95         if (\array_key_exists('error', $jsonData)) {
96             throw new HttpException('an error occurred
97                 ', $jsonData['error'], 401);
98         }
99
100         $userIdAttribute = $this->config->
101             requireString('userIdAttribute');
102         $userId = null;
103
104         // extract the attribute we want
105         foreach ($jsonData['disclosed'][0] as
106             $attributeList) {
107             if ($userIdAttribute === $attributeList['
108                 id']) {
109                 $userId = $attributeList['rawvalue'];
110             }
111         }
112
113         if (null === $userId) {
114             throw new HttpException('unable to
115                 extract "'. $userIdAttribute. '" from
116                 the disclosed attribute(s)', 401);
117         }
118
119         $this->session->set('_irma_auth_user',
120             $userId);
121
122         // return to where the users started at
123         return new RedirectResponse($request->
124             requireHeader('HTTP_REFERER'), 302);
125     }

```

```

115     );
116 }
117
118 /**
119  * @return \LC\Common\Http\UserInfo|\LC\Common\Http\
120     Response|null
121  */
122 public function executeBefore(Request $request, array
123     $hookData)
124 {
125     if (Service::isWhitelisted($request, ['POST' => ['/_
126         _irma/verify']])) {
127         return null;
128     }
129
130     if (null !== $authUser = $this->session->get('_
131         _irma_auth_user')) {
132         return new UserInfo(
133             $authUser,
134             []
135         );
136     }
137
138     // @see https://irma.app/docs/getting-started/#
139     perform-a-session
140     $httpResponse = $this->httpClient->postJson(
141         $this->config->requireString('irmaServerUrl').'_/
142         session',
143         [],
144         [
145             '@context' => 'https://irma.app/ld/request/
146                 disclosure/v2',
147             'disclose' => [
148                 [
149                     $this->config->requireString('
150                         userIdAttribute'),
151                 ],
152             ],
153         ],
154         [
155             'Authorization: '.$this->config->
156                 requireString('secretToken'),
157         ]
158     );
159
160     $jsonData = Json::decode($httpResponse->getBody());
161     if (!array_key_exists('sessionPtr', $jsonData)) {
162         throw new HttpException('"sessionPtr" not
163             available JSON response', 500);
164     }
165     // extract "token" and store it in the session to be
166     used

```



```

158      // @ verification stage
159      if (!\array_key_exists('token', $jsonData)) {
160          throw new HttpException('"token" not available in
              JSON response', 500);
161      }
162      $sessionToken = $jsonData['token'];
163      $this->session->set('_irma_auth_token', $sessionToken
          );
164
165      // extract sessionPtr and make available to frontend
166      $sessionPtr = Json::encode($jsonData['sessionPtr']);
167
168      $response = new Response(200, 'text/html');
169      $response->setBody(
170          $this->tpl->render(
171              'irmaAuthentication',
172              [
173                  'sessionPtr' => $sessionPtr ,
174              ]
175          )
176      );
177
178      return $response;
179  }
180 }

```

Appendix D

IRMA server configuration

```
1  production: true
2  no_email: true
3  # listen only on "localhost" as traffic goes either directly
4  # to localhost, *or* through the reverse proxy
5  listen_addr: "127.0.0.1"
6  port: 8088
7  # this is the URL used by the app to connect to the IRMA-go
   server through the
8  # (reverse) proxy
9  url: "https://vpn.example/irma"
10
11 requestors:
12   vpn:
13     # the attribute to be used for the user ID
14     disclose_perms: ["pbdn.sidn-pbdn.email.email"]
15     auth_method: "token"
16     # key to allow VPN portal to talk to server. Generate one
       using e.g.
17     # `pwgen -s 32 -n 1`
18     key: "dz00SwTqr0tJxpH7uJ9GLOPZMf30CELF"
```

Appendix E

Reverse proxy configuration

```
1 <VirtualHost *:80>
2     ServerName http://irma.spoor.nu:80
3     UseCanonicalName on
4
5     LogLevel warn
6     ErrorLog logs/irma.spoor.nu_error_log
7     TransferLog logs/irma.spoor.nu_access_log
8
9     Redirect permanent / https://irma.spoor.nu/
10 </VirtualHost>
11
12 <VirtualHost *:443>
13     ServerName https://irma.spoor.nu:443
14     UseCanonicalName on
15
16     LogLevel warn
17     ErrorLog logs/irma.spoor.nu_ssl_error_log
18     # Do not log (valid) web browser requests
19     #TransferLog logs/irma.spoor.nu_ssl_access_log
20
21     SSLEngine on
22
23     #SSLCertificateFile /etc/pki/tls/certs/irma.spoor.nu.crt
24     #SSLCertificateKeyFile /etc/pki/tls/private/irma.spoor.nu
25     .key
26     #SSLCertificateChainFile /etc/pki/tls/certs/irma.spoor.nu
27     -chain.crt
28
29     # Let's Encrypt
30     SSLCertificateFile /etc/letsencrypt/live/irma.spoor.nu/
31     cert.pem
32     SSLCertificateKeyFile /etc/letsencrypt/live/irma.spoor.nu/
33     privkey.pem
34     SSLCertificateChainFile /etc/letsencrypt/live/irma.spoor.nu/
35     chain.pem
36
37     # Security Headers
```

```
33     Header always set Strict-Transport-Security "max-age
34         =15768000"
35     ProxyPass "/irma/" "http://localhost:8088/irma/"
36
37     # Redirect requests to the portal (302)
38     RewriteEngine on
39     RewriteRule    "^/$"    "/vpn-user-portal/"    [R]
40 </VirtualHost>
```

Appendix F

IRMA Markdown file

IRMA Authentication

NOTE: IRMA authentication is NOT supported, you are on your own!

NOTE: The IRMA server is NOT part of the VPN software packages. **YOU** are responsible for its installation, configuration, installing updates, keep it secure and in general keep it running!

NOTE: IRMA authentication is NOT production ready! Check the bottom of this document for open issues.

We assume that you already have a working VPN server with valid TLS certificate. See [deployment](#) if you do not already.

IRMA Server Installation & Configuration

Download and install the IRMA server according to the [documentation](#). Use the following configuration file, e.g. `irma.yml`:

```
# use stricter defaults for the configuration options
production: true
no_email: true
# listen only on "localhost" as traffic goes either directly
# to localhost, "*" through the reverse proxy
listen_addr: "127.0.0.1"
port: 8088
# this is the URL used by the app to connect to the IRMA-go server through the
# (reverse) proxy
url: "https://vpn.example/irma"

requestors:
  vpn:
    # the attribute to be used for the user ID
    disclose_perms: ["pbf.sidn-pbf.email.email"]
    auth_method: "token"
    # key to allow VPN portal to talk to server. Generate one using e.g.
    # `pwgen -s 32 -n 1`
    key: "dz00SwTqr0tJxpH7uJ9GL0PZMf30CELF"
```

To start the IRMA server:

```
$ irma server -c irma.yml
```

Portal Configuration

Modify `/etc/vpn-user-portal/config.php` by changing `authMethod` to `IrmaAuthentication` and adding the `IrmaAuthentication` section. For example:

```
// ...

'authMethod' => 'IrmaAuthentication',

// ...
```

```
'IrmaAuthentication' => [  
  // Specify the URL to your (local) IRMA server.  
  // OPTIONAL, DEFAULT: http://localhost:8088  
  //'irmaServerUrl' => 'http://localhost:8088',  
  
  // The attribute used for the user ID in the service  
  'userIdAttribute' => 'pbdn.sidn-pbdn.email.email',  
  
  // The token to talk to the session endpoint of the IRMA server, make  
  // sure it matches the one configured in the IRMA server config  
  'secretToken' => 'dz00SwTqr0tJxpH7uJ9GL0PZMf30CELF',  
],
```

Change the Apache configuration to add the reverse proxy line to allow the IRMA app to talk to the IRMA server. Modify `/etc/httpd/conf.d/${WEB_FQDN}.conf` and add the following line in the `<VirtualHost *:443>` section:

```
ProxyPass    "/irma/"    "http://localhost:8088/irma/"
```

Restart Apache:

```
$ sudo systemctl restart httpd
```