Magdeburger Journal zur Sicherheitsforschung

Gegründet 2011 | ISSN: 2192-4260
Herausgegeben von Stefan Schumacher
Erschienen im Magdeburger Institut für Sicherheitsforschung

This article appears in the special edition »In Depth Security – Proceedings of the DeepSec Conferences«.
Edited by Stefan Schumacher and René Pfeiffer

Extending a Legacy Platform

Providing a Minimalistic, Secure Single-Sign-On-Library

Bernhard Göschlberger and Sebastian Göttfert

Despite decades of security research and authentication standards there is still a vast amount of systems with custom solutions and embedded user databases. Such systems are typically hard to securely integrate with others. We analysed an existing system of an organisation with approximately 12.000 sensitive user data records and uncovered severe vulnerabilities in their approach. We developed a minimal, secure Single-Sign-On-Solution and demonstrated the feasibility of implementing both a minimal Identity Provider and a minimal Service Provider with only a few lines of code. We provided a simple blueprint for an Identity Provider and an easy to use Service Provider Library. Therefore this organisation is now able to integrate arbitrary web based systems. Moreover, others can follow the proposed approach and tailor similar solutions at low cost.
Keywords: Single-Sign-On, Identity Provider, Legacy Systems

Citation: Göschlberger, B. & Göttfert, S. (2016). Extending a Legacy Platform: Providing a Minimalistic, Secure Single-Sign-On-Library. *Magdeburger Journal zur Sicherheitsforschung*, *11*, 686–690. Retrieved April 10, 2016, from http://www.sicherheitsforschung-magdeburg.de/publikationen/journal.html

Version 2016/04/07 14:03

1 Introduction

Nowadays the internet is still full of web applications with custom authentication based on user databases. To a high degree those systems are not actively maintained and further developed any more despite still being actively used. In the context of this paper we use the term legacy system to refer to those kinds of web applications.

This paper illustrates a real world scenario, we encountered during our work for a non-profit organisation. This organisation is managing user data from approximately 12.000 individuals and approached us to extend their existing web platform, which was by our terminology a legacy system. The user records contained sensitive information such as name, age, sex, occupation, address, telephone, email and education.

As the core system was maintained by a company with limited resources, it was only possible to ask for minor modifications. It was not possible to either access the user database directly or run the extension within the same domain. The user credentials (username, hashed password) are also stored in the user record database. As user credentials and other data are frequently updated by system users, a replication of the data (even a periodic one) would not have been sufficient. Also it was impossible to migrate the authentication information to a standard authentication server and modify the legacy system respectively.

The maintaining company had already developed a solution for reusing the legacy system's authentication mechanism upon the organisation's request. We investigated the provided solution which was based on a web service method protected by an API key and discovered several vulnerabilities. The service provided a credential check which could have been used to retrieve arbitrary user credentials using a brute force attack given the API key. The fact that the API key was never changed and used for multiple services increased the risk of it being uncovered or leaked. From a usability viewpoint the solution also appeared to be inadequate as it required multiple logins.

We decided to implement the extension as a separate web application and interconnect the systems using WebSSO.

The remainder of the paper is structured as follows: In section 2 we give an overview of the state of art for SSO. Section 3 covers our individual approach and our design decisions before we conclude in section 4.

2 Background on SSO

Single-Sign-On serves the purpose to authenticate (and sometimes authorise) users against multiple services without having to login multiple times. A popular SSO solutions developed in the last century was the Kerberos protocol. It makes use of three step ticket granting approach and uses symmetric encryption for message exchange (cf. Neuman and Ts' O, 1994).

1. A user has to request a ticket granting ticket (TGT) from an authentication server. The ticket is symmetrically encrypted with a session key, that is encrypted with the hashed password of the user.
2. The client uses this TGT to request a client-to-server-ticket from the ticket granting server for the use of a specific service. The ticket is encrypted using the secret key of the respective service.
3. The client uses the client-to-server-ticket to authenticate the service request and the service verifies the ticket by decrypting it with its secret key.

Kerberos fitted the needs of fat clients quite well, but wasn't designed for the web. With rising popularity of web applications the need for a different protocol focussing on WebSSO became apparent. In 2001 the first version of SAML (security assertion markup language) was published by the Organization for the Advancement of Structured Information Standards (OASIS). As of today, SAML 2.0 represents the most widely used standard for WebSSO.

SAML 2.0 has a rich and diverse feature set, which cannot be covered in depth here. Instead a brief introduction in SAML based WebSSO is given. For further information the reader is invited to have a look at the specification[1].

The standard consists of a document standard for assertions and protocol standards. Assertions are statements about a subject and are represented as an XML document. Protocol standards define how assertions are exchanged between identity providers (IdP) and service providers (SP). The SAML Bindings specification[2] defines the message exchange for different scenarios including WebSSO (see K. D. Lewis and J. E. Lewis, 2009). The most common bindings for this particular use case are HTTP Redirect and HTTP POST (cf. Armando, Carbone, Compagna, Cuellar and Tobarra, 2008). As the HTTP Redirect Binding relies on passing information through GET parameters the length is limited and might not be sufficient in all cases. Therefore the HTTP POST Binding is more flexible as it is not limited in size, however requires the browser to have JavaScript enabled as the redirect is triggered as follows:

```
window.onload = function () {
  document.forms[0].submit();
}
```

Figure 1 illustrates a typical WebSSO authentication flow with SAML 2.0. When the user attempts to request a protected resource at the SP, the SP creates an authentication request (AR) containing a unique re-

1 https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf r. 2016-03-25

2 https://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf r. 2016-03-25

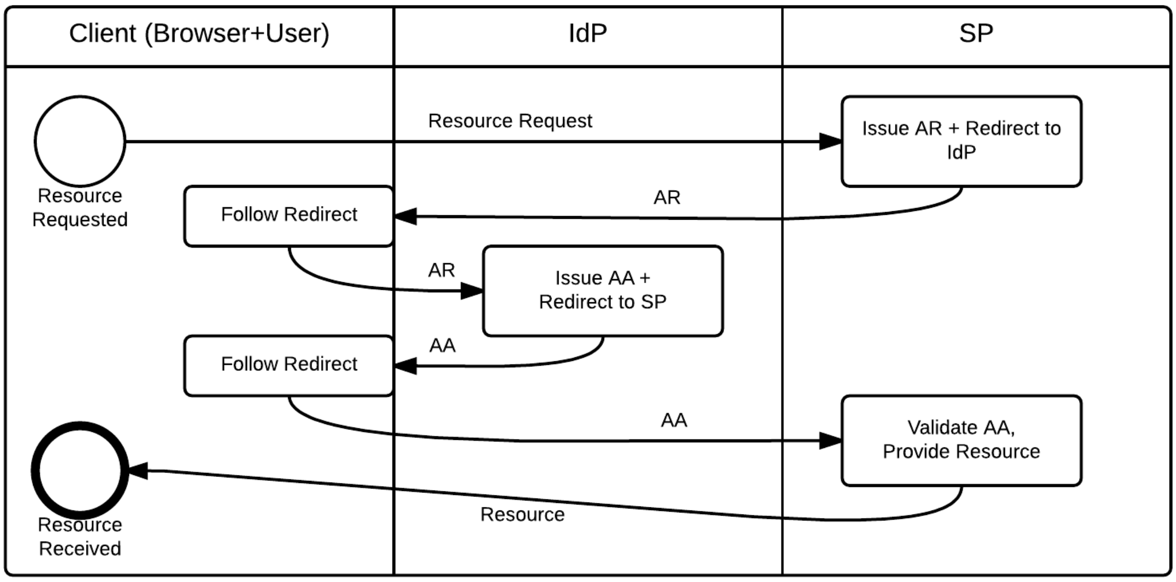

Figure 1: SAML WebSSO Flow

quest ID and redirects the user to the IdP. If the user is not yet authenticated to the IdP, the IdP challenges the user to provide valid credentials (omitted in illustration). Once the user is authenticated to the IdP, it builds an authentication assertion (AA) containing the request ID and signs it with its private key. The user is then redirected to the SP with the AA. The SP can validate the authenticity of the AA and grants the user access to the requested resource.

In the process of open standardisation SAML became very versatile at the cost of increased complexity. Also its XML based approach causes overhead. To overcome these issues JSON Web Token (JWT)[3] has been proposed as a less verbose solution that is easier to implement. It was designed with a modern service oriented web in mind. The subsequent use of related standards such as JSON Web Signature (JWS)[4] and JSON Web Encryption (JWE)[5] contribute to security, simplicity and versatility.

A JWT is an URL safe BASE64 encoded JSON Object containing claims as key/value-pairs. To secure the token it can be encrypted using JWE and/or integrity protected using JWS. JWE and JWS wrap the token as payload and use separate segments to specify a header providing information about the chosen encryption or signature mode and additional segments depending on the chosen mode. The segments (header, payload, additional segments) are separated by a single dot.

The structure of a nested JWT is dependent on the chosen encryption and or signing modes. The following example illustrates JWS wrapping a JWE containing a JWT claim set:

1. header (encryption)
2. key
3. initialisation vector
4. ciphertext (encrypted payload)
    a) header (signature)
    b) payload (JWT claim set)
    c) signature

JWT itself defines neither protocols nor bindings. It is possible to implement a WebSSO flow using SAML Protocols and SAML Bindings and replacing the SAML Assertions with JWTs. For a more extensive overview on JWT see e.g. Jones, 2011.

3 Custom WebSSO Library

The circumstances pointed out in section 1 required us to minimise the involvement of the company maintaining the legacy system. We wanted to provide them with a library that minimised their effort for implementing the SSO IdP part. We were inclined to reduce the actual IdP implementation to a few lines of code and a minimal configuration overhead. Existing libraries, tools and solutions were not able to fulfill our requirements and we decided to implement our own lightweight library based on best practices and existing solutions. In section 3.1 we describe our chosen approach and the authentication flow before we elaborate on our library implementation in section 3.2.

3 https://tools.ietf.org/html/rfc7519 r. 2016-03-25

4 https://tools.ietf.org/html/rfc7515 r. 2016-03-25

5 https://tools.ietf.org/html/rfc7516 r. 2016-03-25

3.1 Design of Authentication Flow

We decided to use the same flow as used by a SP initiated SAML 2.0 authentication request as it requires the minimal amount of messages necessary for a secure SSO mechanism. In fact, the IdP only needs to handle one single request appropriately. This aligns well with our intention to reduce complexity for the IdP.

We further decided to use HTTP POST Bindings in both directions to avoid the aforementioned potential limitations of the Redirect Binding and keep the implementation overhead low. SOAP based bindings (including HTTP Artifact Binding) would have caused substantially more implementation overhead on both sides.

The usage of HTTP POST Binding allowed us to map our necessary parameters to different form fields:

1. SP - IdP (Authentication Request)
   a) nonce
   b) return_url
2. IdP - SP (Authentication Assertion)
   a) user
   b) signature

The form field *nonce* corresponds to the Request ID in SAML. It is a randomly generated one time secret used to encrypt the message payloads symmetrically. It is encrypted using the public key of the IdP.

The SP also provides a *return_url* to the IdP specifying where it has to redirect the user after a successful authentication. This approach entails the opportunity of adding arbitrary SPs without modifying the IdP. If however, the IdP wishes to restrict this authentication flow to certain SPs it may implement respective rules based on the URL. The *return_url* is also encrypted with the public key of the IdP. Asymmetric encryption was chosen over symmetric encryption (using the nonce). The rationale behind this decision is, that we wanted a nonce to only be used for encrypting one single message – the Authentication Assertion.

Once the IdP has challenged the user for his credentials it issues an Authentication Assertion. Such a challenge is met through a valid session (e.g. browser cookie). If users have no valid session yet the IdP asks them to log in. The nonce received from the Authentication Request is decrypted with the IdPs private key and used for the symmetric encryption of the fields.

The Authentication Assertion has a form field *user* containing the symmetrically encrypted payload. This payload can be arbitrary and should comprise all necessary data the SP has permission to request. The payload may vary depending on the *return_url* and is not bound to any data format. This lowers the bar for potential future SP developers.

The unencrypted payload is signed using the IdPs private key. The resulting signature is included in the form field *signature*.

3.2 Library Implementation

The library comprises two parts - a service provider minilib and an identity provider minilib. Both parts of the PHP sample implementation contain about 200 source lines of code altogether. The whole library provides solely three methods - one per incoming request.

The IdP minilib provides the following method:

```
prepare_redirect(idp_private_key, user,
                 c_nonce, c_return_url)
```

The method takes the private key of the IdP, the payload (*user*) and the encrypted input coming from an Authentication Request (*c_nonce* and *c_return_url*). The library decrypts the *nonce* and *return_url* with *idp_private_key*, builds a signature from the user data with *idp_private_key* and encrypts *user* with the decrypted *nonce*.

The library user retrieves all components to issue an Authentication Assertion and redirect the user.

The SP minilib provides the following methods:

```
prepare_redirect(idp_pub_key, return_url)
check_authentication(idp_pub_key, nonce,
                     c_user, signature)
```

The *prepare_redirect* method provides the necessary information to initiate the authentication flow. It takes the public key of the IdP and the desired *return_url*. The method generates a secure random nonce and encrypts both – nonce and return_url – with the public key of the IdP. The SP then only needs to render this data as hidden form fields, to initiate the form submission, and to store the nonce in the session data.

The *check_authentication* method validates a received Authentication Assertion. It decrypts the *c_user* payload with the request nonce and verifies the result using *signature* and *idp_public_key*. If successful, the method returns the decrypted user payload. Through the thorough use of cryptography, the SP can rely on confidentiality, integrity, and authenticity of the result of this method.

4 Conclusion

We consider the particular problem we were facing as a quite common one and feel that it is oftentimes addressed poorly. Our project demonstrates that it is possible to tailor a solution based on the core principles and ideas of proven standards. We consider the SAML WebSSO Authentication flow and especially the SAML POST Binding as the easiest secure way to implement SSO. As far as assertion (or claim) representation is concerned, JWTs compact format is very promising and should be considered in more projects. However, our project required a more flexible, tailor-made solution. The experience we gained during the project showed that a simple, secure WebSSO solution can be built hassle-free. We provide the PHP library resulting from our project as open source[6]. It can be used as is or serve as a blueprint for other custom WebSSO-projects where the use of standards is out of scope.

5 About the Authors

Bernhard Göschlberger, MLBT MSc BSc is researcher and software developer at the Research Studios Austria FG (RSA FG). His research focus is on technology enhanced learning with a special interest in microlearning, social learning, learning analytics and data protection.

Sebastian Göttfert, BSc is student at Johannes Kepler University Linz and was RSA FG project team member for the presented work. He is currently writing his master's thesis on software best practices for Big Data.

References

Armando, A., Carbone, R., Compagna, L., Cuellar, J. & Tobarra, L. (2008). Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. In *Proceedings of the 6th ACM workshop on Formal methods in security engineering* (Pages 1–10). ACM.

Göschlberger, B. & Göttfert, S. (2016). Extending a Legacy Platform: Providing a Minimalistic, Secure Single-Sign-On-Library. *Magdeburger Journal zur Sicherheitsforschung*, *11*, 686–690. Retrieved April 10, 2016, from http://www.sicherheitsforschung-magdeburg.de/publikationen/journal.html

Jones, M. B. (2011). The emerging JSON-based identity protocol suite. In *W3C workshop on identity in the browser* (Pages 1–3).

Lewis, K. D. & Lewis, J. E. (2009). Web Single Sign-On Authentication using SAML. *International Journal of Computer Science Issues*, 41.

Neuman, B. C. & Ts' O, T. (1994). Kerberos: An authentication service for computer networks. *Communications Magazine, IEEE*, *32*(9), 33–38.

6 https://github.com/bgoeschi/minSSO/ r. 2016-03-25