SARL: A revocation mechanism for long lived assertions on Shibboleth

Introduction

Sarl is the abbreviation of the Saml revocation list. It provides the revocation functionality in the Single Sign On (SSO) systems. In the case of the long lived Saml assertions users can logged in through variant devices (e.g., mobile, desktop, tablet, laptop) to multiple services via a SSO system. Usually validation tokens such as saml assertions are expired in a short time period. However, there are cases that a long lived assertion is required when the authentication process needs to be as less frequent as it can. The current project drives a solution for valid long lived assertions as a requirement of degrading authentication process time. Nevertheless, there are cases that assertion should be invalidated and a user must logged out.

For instance, an authentication token generated at the authentication process could have a validation period of a week, month or more. Unlikely, a device for which an authentication process occurred could get stolen. We demonstrated that a revocation process can be a solution as an authentication token can be removed, denying access to a service from a stolen device in this case.

There are variant SSO systems such as Shibboleth, CAS, OpenAM. Most of them use two mainly components. A Service Provider (SP) which provides a service to the users and an Identity Provider (Idp) which is responsible for authenticating a user and generating the authentication token. Each token will be send to the SP each time a user is requesting a service and after a successful authentication.

The current implementation can be deployed in various SSO systems. However, our proof of concept has been deployed and tested to Shibboleth SSO system. Shibboleth is a well known open source SSO platform and is currently used by academia and commerce. Shibboleth uses as well the SP and IdP components. The security token that the Idp generates and the SP consumes is entitled as saml assertion. The heterogeneous applicability of our current implementation (Sarl) in various SSO systems is derived by the use of simple and essential tools such as Operating Systems commands and web tools.

For instance, the collection of the required information called attribute values are gathered by parsers whereas the log out functionality is based on cookies. Parsers uses linux commands to extract necessary information from the Shibboleth log files. Furthermore, when a revocation process takes place the Sarl application set cookies validation time to expire in order to log out a user.

Saml Revocation List (Sarl)

Sarl implementation corresponds to three major software components. Firstly, a Saml revocation list (Sarl) manager located in the Idp side. Secondly, a Saml revocation (Sar) controller located in the SP side. Thirdly, a Saml revocation (Sar) DB which contains the revoked assertion IDs and could be implemented separate from the rest infrastructure.

For demonstration purposes we have implemented an application entitled length checker which runs in the SP side and is under Shibboleth control, meaning that is in the folder that Shibboleth has been configured to control and provide access.

Shibboleth Log files

Shibboleth keeps track of each process adding records to log files for SP and Idp. For both logs the verbosity of the output has been set to DEBUG mode. In the SP side the log file that stores all the necessary information is entitled as shibd.log and in the Idp side as idp-process.log. Both contains the same structure of the generated saml assertions. However, Shibd.log collects the session information which is created at the time a user is accessing a service in the SP part. The session indicated by the saml_session_id attribute is a very important value used to log out a user.

Saml assertions

Together with, Saml assertions are generated each time a user is authenticated from the Idp. Each assertion is redirected to the SP which has to check for the validity of each of them. Valid assertions are consumed by the SP providing access to users for particular services. The attributes that are used in the current application are listed below:

- **ID**: is the saml ID and is used as unique identifier of the session.
- **Uid**: is the username of the user and is used to correlate each user with the specific assertions that are assigned to her.
- **Sarl_url**: is the url where the revocation database is remotely located and the Sarl software should search.

A part of a saml assertion in illustrated below.

```
<saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" ID="_fba17d7b7cb5e0f592c3bbb91dd8ae02"
IssueInstant="2014-04-07T16:36:47.005Z" Version="2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
……..
<saml2:AttributeStatement>


        <saml2:Attribute FriendlyName="uid" Name="urn:oid:0.9.2342.19200300.100.1.1"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
                        cosic
                </saml2:AttributeValue>
        </saml2:Attribute>

        <saml2:Attribute FriendlyName="cn" Name="urn:oid:2.5.4.3" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
                        cosic
                </saml2:AttributeValue>
        </saml2:Attribute>

        <saml2:Attribute FriendlyName="sarl_url" Name="sarl_url" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
                        http://sarl.example.org
                </saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute      FriendlyName="sn"      Name="urn:oid:2.5.4.4"      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri">
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
                        cosic
                </saml2:AttributeValue>
        </saml2:Attribute>
</saml2:AttributeStatement>

</saml2:Assertion>
```

Saml_session_id

Saml_session_id is included as a record in the Shibd.log in the SP side as it is illustrated below.

```
inserted record (_8b1db34f5b7f23fbd2b94ced376a4c7c) in context (_1784822347886413cec0146516100b45) with expiration
(1400593934)
```

The saml session id it is necessary to log out a user as the value of this id is used to identify a specific cookie that is assigned to a specific session. Provided that, the session value is the same as the cookie value in each interaction.

Sarl Components

Sarl Db Server

The Sarl DB is a database (sarl_db) that includes all the revoked records of the saml assertions. It is an one table database with a table name sarl_id. The structure of the database is as follows:

Field	Type	Key	Extra
id	int(11)	PRIMARY KEY	auto_increment
uid	text		
saml_id	text		
Revoked	tinyint(1)		

Id: an id that is used as a primary key,

uid: the user id as it is provided by the saml assertion generated at the time a user log's in

saml_id: the id of the saml as it is provided by the saml assertion at the time a user log's in

Revoked: indicates the status of a record and particularly is refereed to the saml id. Is a boolean value field but the functionality can be extended to support variant status (e.g., revoked, suspended)

Sar Manager

Saml revocation list (Sarl) manager provides to the Shibboleth administrator an overview of the Saml assertions and is located in the Idp side. It provides both the valid and the revoked ones in a list. The distinction of the Saml condition is color indicated providing an easy understanding way as the red is the revoked and green the valid ones. Apart from the user friendliness of the saml status condition display the revocation process is also very easy to maintain. At the left of each Saml assertion record a check box is able to be accessed by the administrator. Once a record is checked and the "Revoke" button is pressed it automatically mark's the particular "id" as revoked (red) and an insertion to the Sar DB process occurred. It means that the saml_id is inserted in the revocation database for further use.

In that case a logged in user when tries to use a service in the Service Provider side will fail to continue as the log out process would triggered. Obviously, the SP should support the revocation functionality with the related code deployed in the SP side.

For instance, the Sarl manager initially parses the Idp log file searching for the attribute values such as uid, saml_id, sarl_url. A list of these triplets is created and stored in a file. The Sarl manager script is reading each line of the file and for each sarl_url it connects to a remote Sar DB server. A query to the connected database with a specific saml_id returns a boolean result of true or false for the existence or not of the id as a record in the DB. It is colored as red if the record exists and green in other case.

When a revocation action is performed Sar Manager queries again the Sar DB using the specific sarl_url. In the case of the saml_id existence the message "The Saml ID: <saml_id> for the user: <uid> has already been revoked" indicating that the particular assertion id holds as a record in the database already. In the case of non existence an insertion of the record is performed to the Sar DB with the attributes uid, saml_id and Revoked and a message "The Saml ID: <saml_id> for the user: <uid> has now been revoked" is printed as a feedback.

Revoked Assertions List

As has been described a Sar DB stores the saml_id records that have been revoked. The revoked_assertions.php script is stored in the Idp side and provides the administrator with an overview of the revoked saml assertions. Moreover, the current version of revoked assertions list supports different DBs providing the administrator an overall view in the case of multiple Sar DB existence.

For instance, when the revoked_assertions.php script runs a parsing of the Idp log file (idp-process.log) action occurs collecting each sarl_url value and store it in a file. The script reads the file and identifies only the variant ones (e.g., sarl1.example.org and sarl2.example.org). Furthermore, it queries the aforementioned servers, dump all the records and displays each one in a separate line indicating the attributes values of uid and saml_id.

Saml Revocation List (Sarl) controller

Saml revocation List (Sarl) controller is located in the Service Provider side. Is the main component of the implementation and responsible for identifying revoked sessions and applying countermeasures for logging out a specific session-user. The script runs in three steps, every time an application runs. Firstly it runs the parser collecting attribute values. Secondly it checks whether a saml_id must be revoked. Thirdly in the case of revoked assertion a log out process in triggered.

For instance, the parser collects and stores in a file the attribute values of saml_id, saml_uid, sarl_url and saml_session_id. These tuples are collected and stored in an array with the structure as it is illustrated below.

```
saml_id =>
        uid,
        sarl_url,
        saml_session_id
```

Given each saml_id the controller queries a dedicated database server (Sar DB) with a url indicated by the attribute value sarl_url. In our case a dedicated server corresponds to a particular saml_id. The query is constructed using the saml_id expecting as a return a boolean value for the "Revoked" field (SELECT Revoked FROM Sarl_id Where saml_id = <saml_id>). In the case of false (0) response value no further action is talking place as the saml_id is not listed in the revocation records. In other case the logout process is triggered.

The log out process is deployed in two phases. Firstly, all cookies regarding to a particular saml_session_id located in the SP are deleted. The php function setcookie expires the aforementioned cookie with expiration value of -<time>sec (e.g., -1000 sec). Secondly, the cookie in the Idp side has to get expired as well. In order this process to occur another script is triggered located in the Idp side. Again the setcookie function is used to set the expiration of a cookie some seconds before the current time.

Appendix

Sar Manager

1. Execute a command line script parsing the idp-process.log file.
2. Create the tuples saml_id, saml_uid, sarl_url and store them in a file
3. Read the file line by line
4. For each tuple read the attribute values which are separated by a special character (e.g., :)
5. Store the attribute values to the aforementioned variables (saml_id, saml_uid, sarl_url)
6. Connect to a particular Sar DB indicated in each tuple using the sarl_url.
7. Query the database sarl_db for the existence of the saml_id record.
8. If the saml_id exist:
    1. an html form row is created displaying the message "User: <saml_uid> Saml id: <saml_id>" where the last one is colored red indicating that the assertion has already been revoked
9. If the saml_id does not exist:
    1. An html form row is created displaying the message "User: <saml_uid> Saml id: <saml_id>" where the last one is colored green indicating that the assertion has not been revoked.
10. When a user select assertions and press the "Revoke" button then
11. A connect to a particular Sar DB occurs indicated by a sarl_url value.
12. Query the database sarl_db for the existence of the saml_id record.
13. If the assertion has been revoked already
    1. The message is displayed "The Saml ID: <saml_id> for the user: <saml_uid> has already been revoked"
    2. A refresh of the webpage action is performed
14. If the assertion has not yet been revoked
    1. An insertion process is triggered adding the saml_id as a record to the sarl_db
    2. The message is displayed "The Saml ID: <saml_id> for the user: <saml_uid> has now been revoked"
    3. A refresh of the webpage action is performed changing the color of the listed assertion from green to red.

Revoked Assertions List

15. Execute a command line script parsing the idp-process.log file.

16. Create the attribute value sarl_url and store them in a file
17. Read the file line by line
18. For each attribute value identify the different ones
19. Store them in an array
20. Connect to a particular Sar DB indicated by the array values.
21. Query the database sarl_db and dump all the records selecting the uid and saml_id fields
22. Display all the results in a list

Sarl controller Algorithm

23. Execute a command line script parsing the shibd.log file.
24. Create the tuples saml_id, saml_uid, sarl_url, saml_session_id and store them in a file
25. Read the file line by line
26. For each tuple read the attribute values which are separated by a special character (e.g., |)
27. Store it's attribute values to the aforementioned variables (saml_id, saml_uid, sarl_url, saml_session_id) in an array with the format:

```
saml_id =>
        uid,
        sarl_url,
        saml_session_id
```

28. For each saml_id get the cookie value.
29. Select the specific cookie value which corresponds to the particular saml_session_id value.
    1. If the cookie value and saml_session_id are equal then connect to the remote Sar DB indicated by the saml_url attribute value.
    2. Query the particular DB for the existence of saml_id record.
    3. If the saml_id is revoked the “Revoked” field gets a True value and a log out process is triggered
        1. assign to the current cookie id the expiration value of -1000 sec
        2. Call a remote script located in the Idp side
        3. The remote script assigns to the corresponding Idp cookie the expiration value of -1000 sec.
        4. Log out the user and redirect her to the log out page.
    4. If the saml_id has not been revoked
        1. the user will continue to have access to the service.