UNIVERSITÀ DI TRENTO

Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer Science

BACHELOR'S THESIS

DESIGN AND IMPLEMENTATION OF AN AUTOMATED TOOL FOR CHECKING SAML SSO VULNERABILITIES AND SPID COMPLIANCE

Supervisor

Silvio Ranise

Co-Supervisor(s)

Andrea Bisegna

Roberto Carbone

Student

Stefano Facchini

Academic Year 2019/2020

Acknowledgements

I dedicate this thesis to my family for the support
demonstrated during this path

I'd also like to thank Andrea and Roberto for their time
and their help in this experience

A special mention is reserved to my former high school professor
M. C. Devoti, whose advice had a significant
impact on my educational career

Contents

Abstract

Since the large-scale diffusion of Internet, getting available not only to companies and organizations but also to privates, a growing number of services has been made accessible through the network to offer a better Quality of Life to the end users. This possibility, however, came along with a set of security issues: the identification of a remote user is more complex than the traditional one due to multiple but intuitive factors.
Several mechanisms arose in the years to assist users in the password management: nowadays a person can have dozen of different credentials, which usually consists of a combination of a username and a password in the typical cases. Single Sign-On (hereafter SSO) is one of the most popular properties in this field: it groups the access to more services with just one set of credentials, offering a better overall experience for the user and lowering the management costs for the organization. However, the security risks are not completely absent: the trade-off is that only one set of credentials needs to be stolen in order to access all the user's services. This makes SSO scenarios to be among the favorite targets for malicious attackers. As the situation between the security experts and cyberattackers is in constant evolution, with the latter one always discovering new flaws to find vulnerabilities, the first step in this work is the analysis of the security situation at the time of the writing. We are focusing on SAML, one of the most popular authentication protocol involved in SSO. Since implementations are error-prone, we want to discover every potential misconfiguration on this side in order to avoid any risk for the user: here comes the need of a tool that is able to discover whether an implementation is misconfigured. As a potentially harmful vulnerability was discovered to be not checked yet by other tools, we proceeded to the implementation of the appropriate test. The tool we developed is a plugin for Burp Suite, a popular software that offer the Proxy capability. Thanks to the Selenium integration, which allows our tool to manage and run the entire login process, the pen-testers now have access to an automated tool which is able to intercept and decode SAML messages and perform both passive and active tests on the desired scenario. Once these have been executed, revealing any possible risk to the user, the results can also be exported in a JSON report to summarize the evaluated situation.
The main contributions of this thesis are the analysis of the current vulnerability situation regarding SAML SSO implementations, the analysis of the message compliance to the Sistema Pubblico d'Identita Digitale (hereafter SPID) format, the design and development of a plugin that can automatically perform a relatively high number of active and passive tests and the tool capability to check the messages compliance against the SPID rules. The testing phase is basically run in three steps: in the first one SAML messages are intercepted, decoded and stored. The second one is the passive analysis phase, where the assertions are parsed and the presence of the selected elements is retrieved and evaluated. The third one is the active testing, where the authentication process is repeated with the tampered values in order to execute more advanced attacks. After each test, the result is printed for the user in the graphical interface.
The developed tool has been tested on several SAML SSO implementations. During the final step, the testing and validation phase, each operation has been double-checked manually to understand if its behavior was the intended one. We can conclude that the results correctly reflected our expectations, confirming the validity of the plugin.

Chapter 1

Introduction

1.1 Context and motivations

Authentication is one of the most frequently used processes worldwide, due to the digitalization of both private and professional life of the people, but also one of the most sensitive ones: it handles credentials and therefore it is the key to private data. In this thesis we are focusing on Security Assertion Markup Language (hereafter SAML) Web Profile 2.0 protocol [16] that supports SSO (hereafter SAML SSO). SSO is a way for users to be authenticated for multiple applications and services at once [6]. If in a first time a user had a set of credentials for each service used, today we have several systems to manage these interactions, designed to reduce the fatigue of the process and provide a better user experience. Thanks to SSO, a user can perform the authentication process once to the Identity Provider (hereafter IdP): if this process is successful, he is granted to access all related Service Providers (hereafter SP) in a "transparent" way.

1.2 Problem

When a user tries to authenticate in a SAML SSO scenario, he communicates with a SP and a IdP, two primary entities in this process. The exchange of information between the SP and the IdP is executed using HTTP messages that contain SAML assertions. The usual scenario consists of the IdP verifying the identity of the user, and one or more SPs that are available to provide the service requested. The communication between the SP and the IdP is made up of SAML messages sent over the Internet.
Besides the authentication result, where it is stated if the user is entitled to access a given resource, also a set of attributes and conditions are reported in order to reduce the possibility of an attack. It is essential that, in the message exchange, SAML authorities both generate the outgoing assertions with all the attributes needed to mitigate the vulnerabilities and that they parse the received message checking that all the attributes are present and valid. Misconfigurations enable malicious users to exploit the authentication flow to perform several attacks, such as Man-in-the-Middle, Denial of Service, Cross Site Scripting and others. Moreover, SPID is based on the SAML framework [9], which allows the creation of a secure federated SSO system. This format is also accepted by the eIDAS regulation [13] that allows several European countries' SSO systems to intercommunicate: it is hence relevant to establish if a SAML SSO implementation that is intended to be SPID-configured satisfies such rules.
Since the implementations are error-prone and, as stated before, a misconfigured entity can represent a risk for its users, it is imperative to have the highest coverage possible in such aspect. The vulnerability situation is constantly evolving and, even if there are already some tools that work in this field, there still are some flaws that are not tested. The necessity of an automated tool able to perform pen-testing on SAML implementations is hence clear, aiming to improve the coverage to assure proper countermeasures to SAML vulnerabilities.

1.3 Contributions

The contributions of this thesis are the following:

- The analysis of currently available penetration testing tools, considering both stand-alone software but also plugins, available for the users to test strengths and weaknesses of SAML implementations. The analysis of the situation is important to understand which are the potentially unchecked flaws of the tested scenarios.

- The design and the implementation of an automated plugin to include in Burp [23], a pen-testing software that offers the Proxy capability, focused on the vulnerabilities not tested by the available tools. Burp was chosen as base in this work due to its several capabilities and the support to third-party plugins.
- The capability to check the compliance against SPID, the Italian SSO system for citizens. Designed by Agenzia per l'Italia Digitale, this system is involved in public services and handles sensitive data: it is mandatory that those messages follow the structure described by AgID [20] to reduce the risk of attacks [9].

1.4 Structure of the thesis

This thesis is divided in the following way:

- Chapter 2: a summary on the technologies involved in the Web-SSO, explaining the basic workflow of SSO and the role taken by SAML in this process.
- Chapter 3: the illustration of the preliminary analysis on available tools that allow checking the integrity and possible flaws of a SAML scenario. The research work is focused on Burp Suite and its plugins, due to its diffusion and its support to third-party plugins.
- Chapter 4: the description of the developed automated tool, covering several aspects such as implementation of the checks, the related risks and the meaning of their results.
- Chapter 5: in this chapter we analyzed three scenarios with the developed plugin: a test environment provided by Istituto Poligrafico e Zecca dello Stato (hereafter IPZS), a SPID scenario and a generic SAML SSO scenario.
- Chapter 6: it consists of some concluding considerations and possible improvements to enhance the work described here.

Chapter 2

Background

In this chapter we are recalling some basic definitions, as well as fundamental concepts, needed to fully understand the context of the provided work.

2.1 Authentication and authorization

Authentication and authorization are the two basic concepts regarding security and protecting sensitive data, both in the physical and digital contexts.
Authentication is the process of proving the identity of someone, for instance, proving that you really are who you claim to be. Although in the everyday life authentication is performed simply with the physical presence, or more formally with the exhibition of ID documents, in the digital context this might be more complex. The authentication process is passed if the needed *authentication factors* are provided by the user: those are divided in three categories based on the type of the factor:

- Knowledge factors: based on something you know. Here we can think about, for instance, passwords and PIN numbers.
- Ownership factors: based on something you have or hold. Security cards and NFC transmitters fall under this category.
- Inherence factors: based on something you are. Biometric parameters, such as fingerprint and retinal pattern, belongs to this section.

Authorization, in contrast, is the process of stating if a defined user is entitled to access a certain resource, item or any other kind of protected service. Once the user requests the resource, there is a specific entity that checks his attributes and permissions and decides whether he is allowed to access or not the requested item. There are several types of Access Controls policies, such as MAC, DAC and RBAC, but we are not covering their description in this paper.

2.2 Single Sign-On

SSO is the authentication process that allows a user to access several resources with a unique login session, i.e. a single set of credentials. There are three entities involved: the *principal*, the SP and the IdP.
The principal, commonly a user, tries to access a specific resource or operate a service, usually through a web browser. The SP is the holder of such resource or the host of the service. Finally the IdP manages the identity of the user and it is responsible for verifying the user authentication and its eligibility to access the requested service or resource.
SSO can be started either SP-side or IdP-side. In the first case it is called *SP-initiated SSO*, while in the latter it is named *IdP-initiated SSO*.
In the SP-initiated SSO (Figure 1), which is the most frequently used method, the user first tries to access a service, then the SP system redirects him to the authentication portal managed by the IdP. Once the user is correctly logged, the SP returns the requested item.

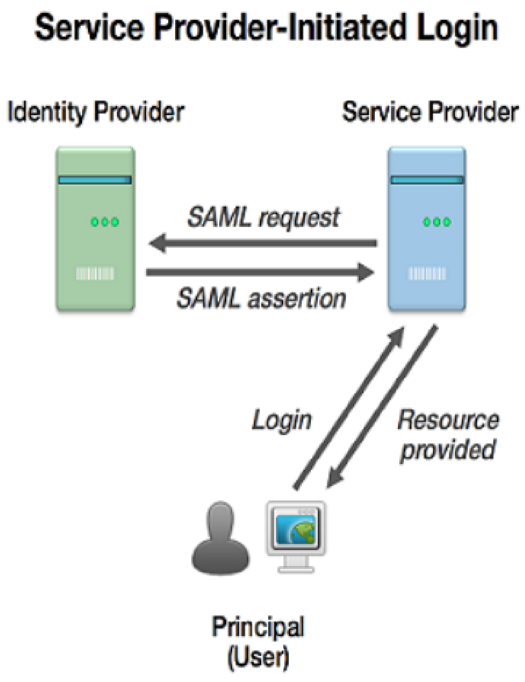


Figure 1: SP-Initiated SSO.

On the contrary, IdP-initiated SSO (Figure 2) consists of the user first authenticating to the IdP. Once the authentication is successful and the permissions have been granted, the user can request the resources to the SP, thanks to the trust relationship between the two SAML authorities.

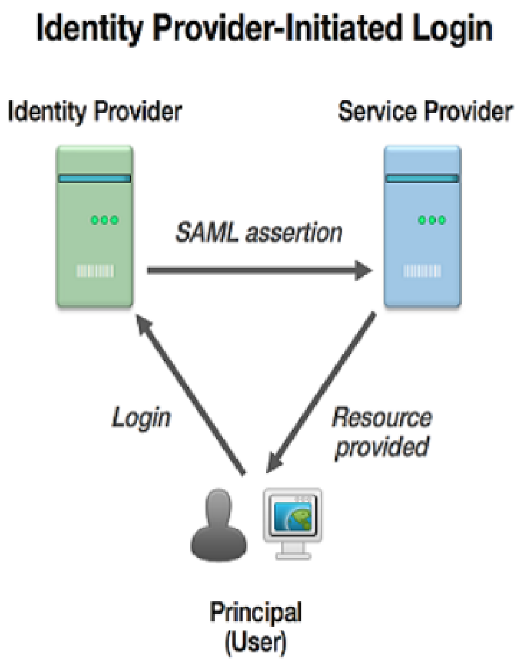


Figure 2: IdP-Initiated SSO.

2.3 Security Assertion Markup Language

SAML is a standard language based on XML for exchanging authentication and authorization data between two entities, in particular, between an IdP and a SP. In order to use SAML messages, the two entities need to

establish a trust relationship over a secure channel (for instance, using SSL/TLS and or a certificate) by the exchange of metadata. The SAML structure consists of 4 layers: *Profiles*, *Bindings*, *Protocols* and *Assertions*.
The Profiles layer defines the use case and specify how Bindings, Protocols and Assertions are combined to support the specified use case. The most used profile, and the one we are focusing on in this thesis, is the Web-Browser SSO Profile [16].
The Bindings layer maps SAML protocols to standard messaging protocols to allow the communication over the network. The two most popular Bindings in the Web-Browser SSO Profile are the HTTP-Redirect binding and the HTTP-POST binding.
The Protocols layer describes how SAML elements are embedded within SAML request and response messages, and provides the processing rules that SAML entities must follow when producing or consuming these items.
Finally, the Assertions layer contains one of the following kind of assertions defined by SAML:

- Authentication: The Principal was authenticated by a particular mean at a particular time
- Attribute: subject is associated with the supplied attributes
- Authorization: a request to allow a subject to access a resource has been granted or denied

The typical authentication and authorization flow runs as shown in Figure 3, with SAML involved in steps 2 to 6.

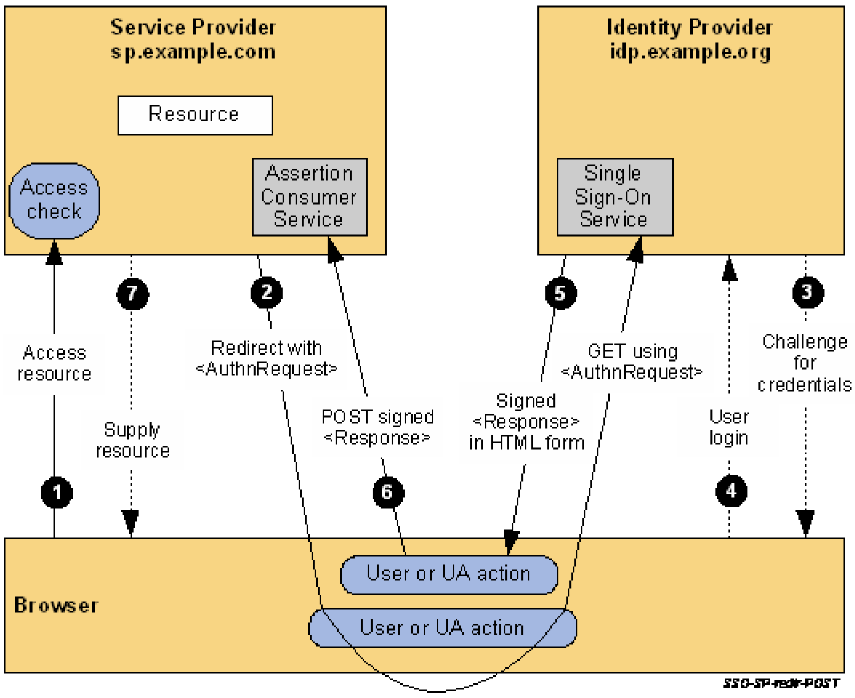


Figure 3: Messages flow in SP-initiated SSO [14].

Chapter 3

Preliminary Analysis of Vulnerabilities in SAML SSO

The first task accomplished during this work is the evaluation of the initial situation. Since the main focus of this work is covering SAML SSO flaws that were not checked yet by other tools, we first collected known vulnerabilities affecting SAML SSO and then we associated those with tools that are testing them.

3.1 Sources

As stated, we started gathering possible vulnerabilities related to the SAML SSO process. In order to do that, I consulted some of the most relevant blogs and publishers in the Security and Vulnerabilities field:

- The Open Web Application Security Project [25]: a nonprofit foundation that works to improve the security of software.
- The Duo Blog [10]: Duo is a user-centric access security platform and its blog contains latest news about vulnerabilities handled by its team.
- The Common Vulnerabilities and Exposures system [7]: The CVE, maintained by The Mitre Corporation, is a list of entries for publicly known cybersecurity vulnerabilities.
- The SSO-Attacks website [21]: created by the Chair for Network and Data Security at Ruhr University Bochum, it aims at delivering the most comprehensive enumeration of all known SSO attacks.
- Some other online resources and blogs such as Epi052's [11] and Anitian's [3] that aim to spread awareness about cybersecurity attacks and explain, for educational purpose, how to execute those.
- The previous thesis works of colleagues that completed their thesis in this field [17, 24, 19].

3.2 Vulnerabilities

After analyzing the sources described in Section 3.1, in the following section the vulnerabilities are reported divided by their type.

3.2.1 XML Signature Wrapping - XSW [27]

The idea behind XSW is to attach a different SAML message into the original one in order to fool the final signature check algorithm. There are 8 variants depending on where the extra message is attached.

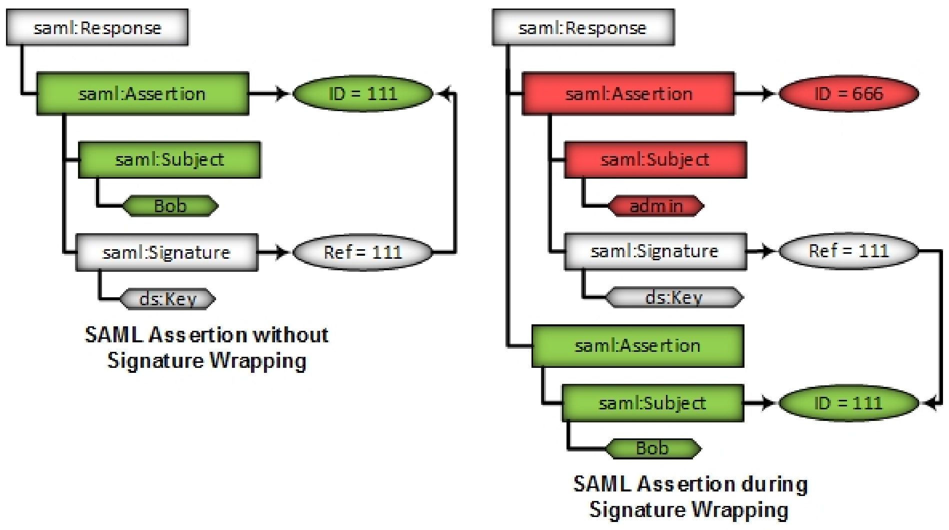


Figure 4: An example of XSW. The green part is the original content, while the red one is the attached one [22].

3.2.2 XML Certificate Faking [11]

Certificate faking is the process of testing whether or not the SP verifies that a trusted IdP signed the SAML Message. The trust relationship between the SP and the IdP is established and should be verified each time a SAML message is received.

3.2.3 XML Signature Exclusion [26]

An alternative to XSW is represented by XML Signature Exclusion. SOAP message validation flow consists of several independent steps: signature verification, certificate validation, business logic invocation, etc. There exists a possibility that one of these steps is omitted, and the message is still considered to be valid.

3.2.4 Missing XML Validation [17]

Missing XML Validation is a class of XML attacks, which has as target the XML parser, that relies on the lack of sanitization of the analyzed XML file. This class of attacks is mainly divided in 2 sets. The first one is XML Entity Expansion (XEE), while the second is called XML External Entity (XXE).
Attacks that fall under the first category aim to fill and then overflow the available memory, resulting in a Denial of Service for the final user: the core idea behind those are that the XML parser, resolving an apparently simple XML document, occupies all the available memory causing a crash of the system. The main specific attacks of this type are the Billion Laughs attack and the Quadratic Blow-up attack.
Example of attacks in the second category are the Classic XXE, Blind XXE and SchemaEntity XXE. The aim of these attacks is to insert a reference to an external file into the XML document in order to cause a disclosure of sensitive data or a Denial of Service.

XEE - Billion Laughs

The classic Billion Laughs attack is illustrated in the XML file represented below.

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lollollollollollollollollol">
<!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

In this example we declare ten different XML entities, from *lol1* to *lol9*. The first entity, *lol* is defined to be the string *"lol"* repeated 10 times. However, each of the other entities are defined to be 10 of the previous entity. The body section of this XML file contains a reference to only one instance of the entity *lol9*. However, when this is being parsed by parser, when *lol9* is encountered, it is expanded into 10 *lol8*s, each of which is expanded into 10 *lol7*s, and so on and so forth. By the time everything is expanded to the text *lol*, there are 1,000,000,000 instances of the string *"lol"*, hence the name of the attack. Intuitively, this quantity of expansions consumes an exponential amount of computational time and resources, causing the DOS.

XEE - Quadratic Blow-up

An alternative to the Billion Laughs is the Quadratic Blow-up. Although the objective is the same, this one aims to avoid some protections that prevent the Billion Laughs to be successful. Such protections consists of limiting the times an entity can expand itself. The quadratic blowup variation causes quadratic growth in storage requirements by simply repeating a large entity over and over again, to avoid countermeasures that detect heavily nested entities.

```
<!DOCTYPE data [
  <!ENTITY lol "lol...100'000 times...lol"
]>
<data>&lol;&lol;...100'000 times...&lol;</data>
```

XXE - Classic XXE

XML External Entity is a class of attacks that exploit the XML parser to load external entities without checking its genuineness. Those entities can be malicious for the system and might be vector of Denial of Service or disclosure of sensitive data. Since the parser might produce different results based on the used encoding, it is worth checking that the vulnerability is not present not only with UTF-8, but also with UTF-7 and UTF-16.

XXE - Blind XXE

Blind XXE, also known as Out-Of-Band XXE, is an alternate version of the Classic XXE. Blind XXE vulnerabilities arise where the application is vulnerable to XXE injection but does not return the values of any defined external entities within its responses. This means that direct retrieval of server-side files is not possible, and so blind XXE is generally harder to exploit than regular XXE vulnerabilities.

XXE - SchemaEntity

A further variation of the Blind XXE is represented by the SchemaEntity attack. Known to come in three variants, it is based on the attribute that refers to the External Entity: the possible ones are schemaLocation, noNamespaceSchemaLocation and XInclude. An example of SchemaEntity attack applied with the noNamespaceSchemaLocation attribute is shown below.

```
<!DOCTYPE data [
<!ENTITY % remote SYSTEM "http://attacker.com/external_entity_attribute.dtd">
%remote; ]>
<data xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
xsi:noNamespaceSchemaLocation="http://192.168.2.31/&internal;"></data>
```

3.2.5 Canonicalization form [10]

The canonicalization is the process of converting the XML document into its normal form, also called canonical XML. This process is used to compare XML documents, since if two XML files have the same canonical form, then these are equivalent by definition. This process is performed applying a set of rules regarding encoding (where UTF-8 becomes mandatory), special characters representation and some others. The security issue is present due to an unexpected behavior of some XML parser such as Python's *lxml* or Ruby's *REXML* [5] regarding the management of comments. The classic canonicalization algorithm can pave the way to an user impersonation attack. The best remediation is either to make sure the XML parser used is not affected by this vulnerability or to use a safe canonicalization algorithm, as explained in Chapter 4.

3.2.6 RelayState tampering [24]

RelayState is a parameter of the SAML protocol that is used to identify the specific resource the user will access after they are signed in and directed to the relying party's federation server. Without the use of any parameters, a user would need to go to the IDP initiated sign on page, log in to the server, choose the relying party, and then be directed to the application. Using RelayState can automate this process by generating a single URL for the user to click and be logged in to the target application without any intervention. It is clear that tampering such value can represent a threat for the end user, that would be redirected to a potentially malicious page after the login.

3.2.7 Session protection [24]

The Session attack consists of replaying the intercepted response without the authentication cookie previously established betweeen SP and IdP. This cookie is needed to avoid Cross Site Request Forgery that can result in user impersonation.

3.2.8 Incorrect generation [24]

Incorrect generation is a class of vulnerabilities that might occur when the SAML authority that is generating the assertion is not properly configured. The attributes that we consider to be threatening for the security of the SAML environment are the following:

- For the SAML Request:
  - ID
  - ProviderName
  - Issuer
  - The presence of the signature
- For the SAML Response:
  - References to the SP
  - InResponseTo
  - The used canonicalization form
  - Audience
  - Recipient
  - OneTimeUse
  - NotOnOrAfter

3.3 Software and plugins

Given that SAML is widely used in the SSO scenarios, there already are some tools that perform security tests. Since our aim was to execute checks that were not previously available, we analyzed available software to understand which were left to implement: in this phase, those programs have been installed to understand their features. This evaluation was focused on Burp Suite and its plugins, bearing in mind that it is one of the most used products in Network Security field. The studied tools are:

- Burp Suite [23].
- SAML Raider [4]: SAML Raider is a Burp Suite plugin for testing SAML infrastructures. It contains two core features - a SAML message editor and an X.509 certificate manager.
- EsPReSSO [12]: this Burp plugin processes and recognizes SSO protocols and integrates WS-Attacker, DTD-Attacker and XML-Encryption-Attacker while intercepting SAML messages.
- SAML ReQuest [2]: this plugin enables security testers to view, decode, and modify SAML Authentication requests and test IdP's response to their manipulations. Note that this plugin does not perform any test and has not been included in further analysis, but it is reported in the current list for completeness.
- MXV Detector [17]: unlike the previous programs, this plugin is compatible with OWASP Zap, an alternative to Burp. Developed by a colleague in a previous internship experience, it is mainly focused on XXE and XEE vulnerabilities.

As result of the analysis made in Section 3.2 and Section 3.3, a resume table (Table 3.1) was compiled to match each vulnerability with the tool that performs its check.

Table 3.1: The table shows, for each vulnerability, which tools cover it. The red ones are tested by our plugin, called MIG - SAML SSO.

Vulnerability name	Burp Suite	EsPReSSO	SAML Raider	MIG - SAML SSO
XML Signature Wrapping #1-#8			X	
XML Signature Exclusion		X		
XML Certificate Faking		X	X	
XEE - Billion laughs		X		
XEE - Quadratic blowup		X		
Canonicalization form				X
Classic XXE - UTF-8		X		
Classic XXE - UTF-16		X		
Classic XXE - UTF-7		X		
Blind XXE (Out of band)	X			
Bypassing XXE restrictions		X		
SchemaEntity via schemaLocation		X		
SchemaEntity via noNamespaceSchemaLocation		X		
SchemaEntity via XInclude		X		
RelayState tampering				X
Session attack				X
Response containing SP references				X
Missing recipient attribute				X
Missing Audience element				X
Missing OneTimeUse attribute				X

Missing NotOnOrAfter attribute	X
Missing InResponseTo attribute	X

Chapter 4

Description of MIG - SAML SSO

This chapter contains a brief presentation of Burp, the software used as base component for this work. Also, the designed solution will be introduced and described to understand the operations it can perform and the meaning of the results displayed. The overall idea is that, given the analysis described in the previous chapter, we considered the vulnerabilities that were left unchecked and developed an appropriate test to cover each of these.

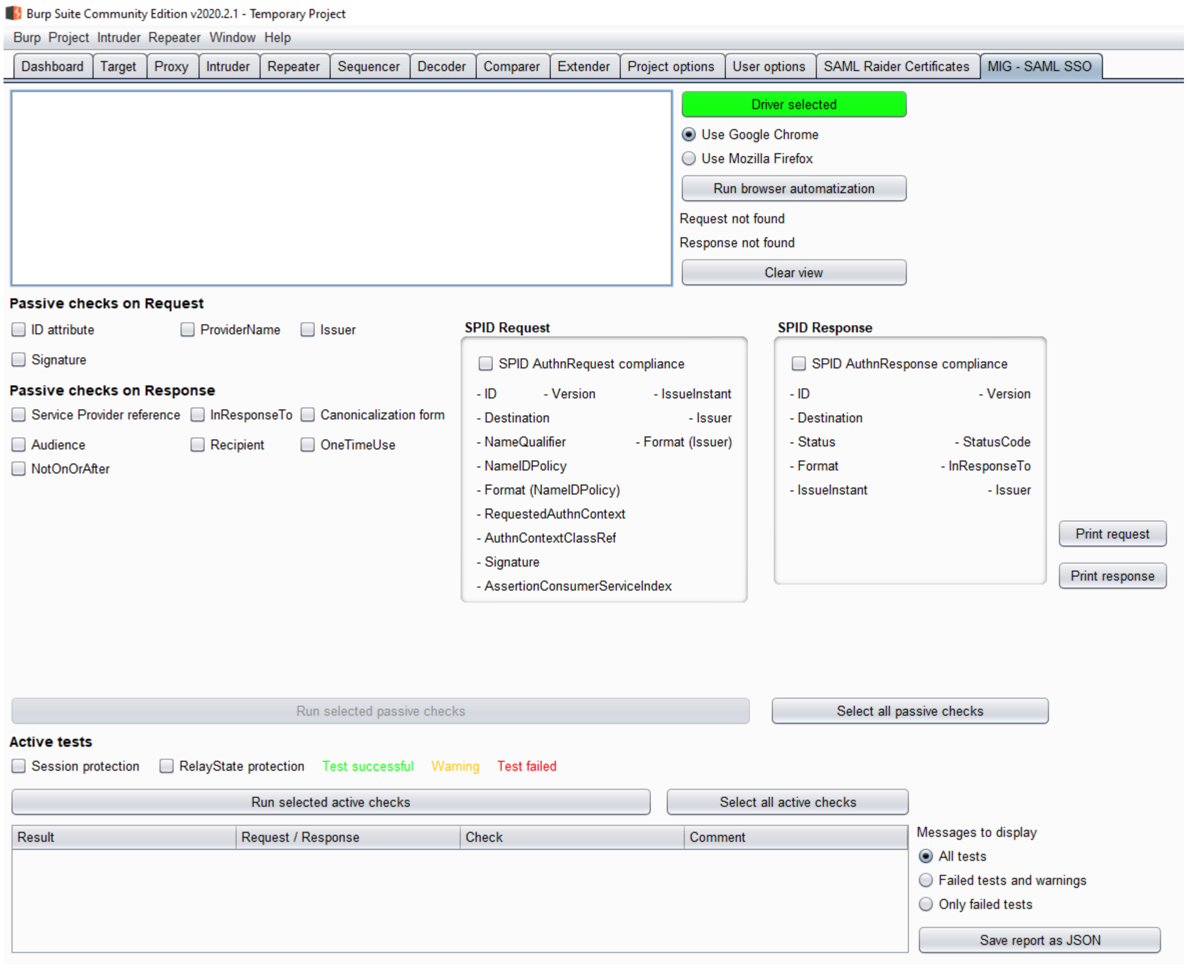
4.1 Burp Suite

Burp Suite [23], developed by PortSwigger, is a platform for performing security testing of web applications. It provides a set of interesting built-in functionalities: among these we can find the Scanner, the Proxy and the Intruder. First, the scanner identifies and notifies the user regarding security-related issues and generates a report of encountered ones. Second, the Proxy works as middle point between the client and the server, through which requests and response are intercepted, shown and eventually modified. Third, the Intruder allows us to execute personalized attacks, adding custom payloads created by the user. In addition, Burp Suite is open to third-party plugins, providing several interfaces that permits to work with Burp objects and tools. Its company encourages developers to work with Burp APIs, supporting them with documentation, snippets of example code and even an official forum.

4.2 MIG - SAML SSO

The developed tool, called MIG - SAML SSO, consists of a Burp Suite plugin that is able to recognize SAML messages, properly decoding the assertion depending on the used binding, to perform penetration testing and to presents the results both graphically and in a tabular way to the user. The user interface, depicted in Figure 5, is visually divided in three sections:

- The upper one is the setup area, where the user is allowed to insert the authentication trace and choose the WebDriver file: these steps allow the system to automatically perform the login process.
- The middle one is the configuration area. Once the authentication process is finished and our plugin has successfully intercepted the SAMLRequest/SAMLResponse messages exchange, the user can choose which checks to perform selecting the corresponding option here. More details about each test are reported in Section 4.3.
- At the bottom we can find the Summary area: once tests are performed, the results are shown here in the table. Moreover, the results can also be exported as JSON through the apposite button on the right side of the screen.



Driver selected

[✓ Use Google Chrome]

Request not found

Response not found

Messages to display: [✓ All tests]

Figure 5: The graphical interface of MIG - SAML SSO.

4.2.1 Usage

In our testing we first generated the authentication trace using the appropriate software (such as Katalon [15] or UI.Vision [1]), which allows Selenium to automatically perform the login process once the proper WebDriver was selected.
The tool can be easily used in the following way: after generating the trace with the aforementioned software and having it executed, we can notice that the SAML messages have been intercepted. It is then possible to select the desired passive checks and perform those with the apposite button: they will be colored in a green, yellow or red color depending of the result. Finally, we can proceed to execute the active checks in the same way, selecting the ones in which we are interested and starting the process with the button. Also those one will be colored depending of the tests results.

4.3 Performed tests

The following section contains the list of implemented checks, including their description and the meaning of their result. There are two categories of tests, depending on how these are performed: the first one are the passive checks, while the second one are the active checks. In the former, we analyze the exchanged messages as they are: using the proxy functionality the SAML messages are intercepted and inspected. In the latter, we have to intercept the ongoing message, perform an effective action over it (for instance, the rewriting of a parameter, etc) and then forward the message to the desired destination. Each test is either marked as *Mandatory* or *Optional*. Mandatory means that the related vulnerability is required to pass, otherwise it represents a high risk for the user. On the contrary, Optional means that the test should pass to grant the user security, but the failure does not necessarily lead to a risk for the user. Moreover, the possible results are three: *Successful*, *Warning* and *Failed*. For the majority of the checks, Successful means that the test was passed, Warning states that an optional test was failed or certain conditions were not satisfied while Failed indicates that the test did not pass.

4.3.1 Testing on SAML Request

ID Attribute [18]

Mandatory
Retrieve the presence of the `ID` attribute of the AuthnRequest element, which is the root of the assertion. This attribute is needed to uniquely identify the authentication flow between the SP and the IdP. Its omission should invalidate the process.

Test successful: The `ID` attribute was found and its value is not empty.
Warning: Not available for this test.
Test failed: The `ID` attribute could not be found or its value is empty.

Provider name [18]

Optional
The `ProviderName` attribute specifies a friendly name for the SP. It should identify the same entity of the Issuer element.

Test successful: The `ProviderName` attribute was found and its value is not empty.
Warning: The `ProviderName` attribute could not be found or its value is empty.
Test failed: Not available for this test.

Issuer [18]

Mandatory
The `Issuer` attribute specifies the formal name of the Service Provider. This node must be child of the AuthnRequest element. This element is mandatory and the authentication flow should be invalidated if it is not present.

Test successful: The `Issuer` element was found and its value is not empty.
Warning: Not available for this test.
Test failed: The `Issuer` element could not be found or its value is empty.

Signature [18]

Optional
Check the presence of the signature in the AuthRequest. Since this plugin supports both HTTP-Redirect and HTTP-POST bindings, the Signature is searched in the appropriate place depending which binding is detected.

Test successful: The signature was found and its value is not empty.
Warning: The signature could not be found or its value is empty.
Test failed: Not available for this test.

SPID AuthnRequest Compliance [9]

Optional
The plugin checks if the Request is compliant to the SPID format. The compliance is optional in the majority of applications, but it is mandatory if it is a SPID scenario. The test is run against the set of technical rules

described on the official website [9].

Test successful: The request is compliant to the SPID standard.
Warning: Not available for this test.
Test failed: The request does not follow the SPID guidelines.

4.3.2 Testing on SAML Response

Service Provider references [24]

Mandatory
The plugin checks if the either the value of the Issuer element or the value of the ProviderName attribute of the Request is contained into the Response, more precisely in the Audience node. Although the omission of the Issuer element must be considered a flaw, this check will also consider the ProviderName attribute for this test in case the Issuer is missing.

Test successful: References of the SP were found in the response.
Warning: Not available for this test.
Test failed: No references about the SP could be found in the response.

InResponseTo [24]

Mandatory
The plugin checks if the `InResponseTo` attribute of the Response matches the ID attribute of the Request. If the values are not equals, the authentication flow shall be invalidated. `InResponseTo` must be found in both the following locations:

- As attribute of the root element AuthnResponse of the SAML response
- As attribute of the SubjectConfirmationData element, that can be found at the following path: AuthnResponse/Assertion/Subject/SubjectConfirmation

Test successful: The `InResponseTo` element was successfully found in both SubjectConfirmationData and Conditions nodes.
Warning: The `InResponseTo` element was found, but only in one of the two allowed locations.
Test failed: The `InResponseTo` element could not be found neither in SubjectConfirmationData nor in Conditions.

Canonicalization form [5]

Mandatory
Checks the canonicalization algorithm that has been used to transform the request, if any. An unsafe algorithm can be exploited and any Auth process involving unsafe canonicalization must be dropped.

Test successful: The safe algorithm, where comments are included, was used in the canonicalization process.
Warning: The plugin cannot decide if the algorithm used was safe or unsafe.
Test failed: The unsafe algorithm was used, comments might be exploited.

Audience [24]

Mandatory
The `Audience` element contains the name of the Service Provider that is waiting for this response. It usually is the domain of the SP, but other values are accepted conventionally. The Audience element is child of the

AudienceRestriction element, that must be found at the path: AuthnResponse/Assertion/Condition.

Test successful: The `Audience` element was found and it is not empty.
Warning: Not available for this test.
Test failed: The `Audience` element was not found or its value is empty.

Recipient [24]

Mandatory
The plugin checks the presence of the Recipient attribute of the SubjectConfirmationData element, which shall be located at the path AuthnResponse/Assertion/Subject/SubjectConfirmation.

Test successful: The Recipient element was found and it is not empty.
Warning: Not available for this test.
Test failed: The Recipient element was not found or its value is empty.

OneTimeUse [24]

Mandatory
The tool checks whether the `OneTimeUse` attribute is present in the current response or not. It must be found as attribute of the element Condition, which is a child of the root element AuthnResponse. Note that this attribute is mandatory, but some providers might implement additional mechanisms that prevent assertions to be reused. In those cases this attribute is ignored and thus its omission has no consequences. However, it is good practice to specify it anyway to avoid any risk.

Test successful: The `OneTimeUse` attribute was found and it is not empty.
Warning: Not available for this test.
Test failed: The `OneTimeUse` attribute was not found or its value is empty.

NotOnOrAfter [24]

Mandatory
Checking if the `NotOnOrAfter` attribute is present in the response. It is recommended that it is set to expire in a short period of time, which we assume to be around 5 minutes. It can be found in either one or both of the following element:

- In SubjectConfirmationData element, which can be found at the following path AuthResponse/Assertion/Subject/SubjectConfirmation
- In Conditions element, which is a child of the root element AuthnResponse.

Test successful: The `NotOnOrAfter` element was found in both paths and it is set to a short time.
Warning: The `NotOnOrAfter` attribute was found, but one of the following conditions happened:

- The attribute is present in only one of the two elements
- The value of the attribute is set to a higher value that the recommended.
- The value of those two attributes does not match.

Test failed: The `NotOnOrAfter` attribute was not found.

SPID AuthnResponse Compliance [9]

Optional
The plugin checks if the Response is compliant to the SPID format. The compliance is optional in the majority of applications, but it is mandatory if it is a SPID service. The test is run against the set of technical rules described on the official website [9].

Test successful: The response is compliant to the SPID standard.
Warning: Not available for this test.
Test failed: The response does not follow the SPID guidelines.

4.3.3 Active tests

Session [24]

This test checks that the Session of the principal cannot be reused, based on the fact that Client and Server should use an authentication cookie. If this is not the case, a malicious user can perform a replay attack with no cookie in the request and successfully log as the legitimate user.

Test successful: Proper protection is enabled and the Session test was successful.
Warning: Not available for this test.
Test failed: No protections were involved and the Session could be reused.

RelayState [24]

This test tampers the RelayState parameter in the Request, so the user is going to be redirected to a potentially harmful page after the login phase. The test checks if the parts involved keep protected the original value, generating an error in case it is modified.

Test successful: Proper protection is enabled and the RelayState test was successful.
Warning: Not available for this test.
Test failed: No protections were involved and the RelayState could be tampered with no security checks.

Chapter 5

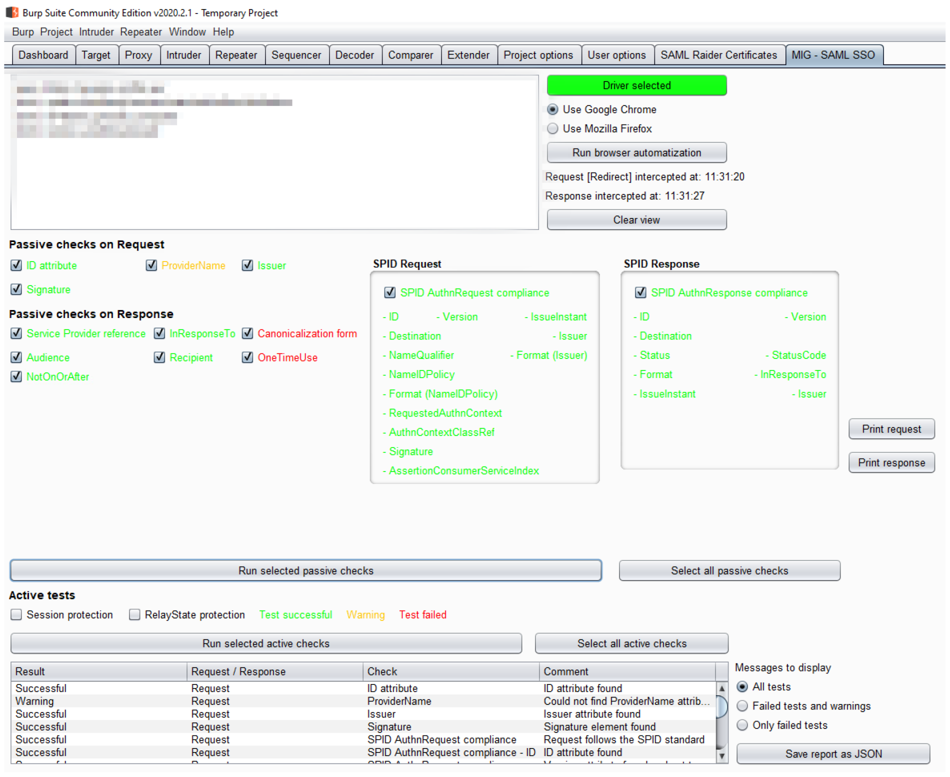
Experimental Analysis

In this chapter three use cases will be presented. The target case is the scenario made available by IPZS, while for validation purposes we also tested some others SPID and SAML scenarios with MIG - SAML SSO. In those cases personal accounts have been used to perform the authentication flow. For disclosure reasons, the names of the SPID scenario and the SAML SSO scenario will not be specified.

5.1 IPZS use case

The main target, thanks to the industrial collaboration between IPZS and the S&T research unit of Fondazione Bruno Kessler, is a fully working scenario composed of an IdP and a SP. More precisely, the developed tool has been run on a test environment, reserved for developing purposes. Since one of the authentication options is represented by the Italian Carta d'Identità Elettronica (CIE) [8], we used a test account loaded on one of these which was read with an NFC reader.
It is worth to mention that in the case of the IPZS scenario we cannot think about a direct SPID compliance. Those messages are more precisely compliant to the eIDAS regulation [13]. We can, on the contrary, state that if exchanged messages in this scenario are SPID compliant, then they are eIDAS compliant, since we know that SPID is certified to be eIDAS compliant.
As the results reported in Figure 6 are the expected ones, there still are a few notes that can be made:

- ProviderName: a warning is displayed, since this attribute is missing but it is optional. However, it works as a friendly name for the SP and it might be used for some secondary reasons by the SAML authorities, hence it is good practice to include it.
- Canonicalization form: the detected canonicalization algorithm has been observed to be unsafe. More advanced analysis should be done in order to understand if the scenario suffers this vulnerability or hidden protection mechanisms are working to offer the proper security.
- OneTimeUse: the OneTimeUse element is missing. This element explicitly tells the receiving SAML authority that this message shall be consumed only the first time it is received. Although most scenarios implicitly apply this rule, there might be a few where this is not true, hence this element should always be included.


Burp Suite Community Edition v2020.2.1 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options SAML Raider Certificates MIG - SAML SSO
Driver selected
Run browser automatization
Clear view
Passive checks on Request
Passive checks on Response
SPID Request
SPID Response
Print request
Print response
Run selected passive checks
Select all passive checks
Active tests
Session protection
RelayState protection
Test successful
Warning
Test failed
Run selected active checks
Select all active checks
Messages to display
Save report as JSON

[redacted]

[✓ Use Google Chrome] Use Mozilla Firefox

Request [Redirect] intercepted at: 11:31:20
Response intercepted at: 11:31:27

Passive checks on Request: [✓ ID attribute] [✓ ProviderName] [✓ Issuer] [✓ Signature]

Passive checks on Response: [✓ Service Provider reference] [✓ InResponseTo] [✓ Canonicalization form] [✓ Audience] [✓ Recipient] [✓ OneTimeUse] [✓ NotOnOrAfter]

SPID Request: [✓ SPID AuthnRequest compliance] - ID - Version - IssueInstant - Destination - Issuer - NameQualifier - Format (Issuer) - NameIDPolicy - Format (NameIDPolicy) - RequestedAuthnContext - AuthnContextClassRef - Signature - AssertionConsumerServiceIndex

SPID Response: [✓ SPID AuthnResponse compliance] - ID - Version - Destination - Status - StatusCode - Format - InResponseTo - IssueInstant - Issuer

Active tests: Session protection, RelayState protection

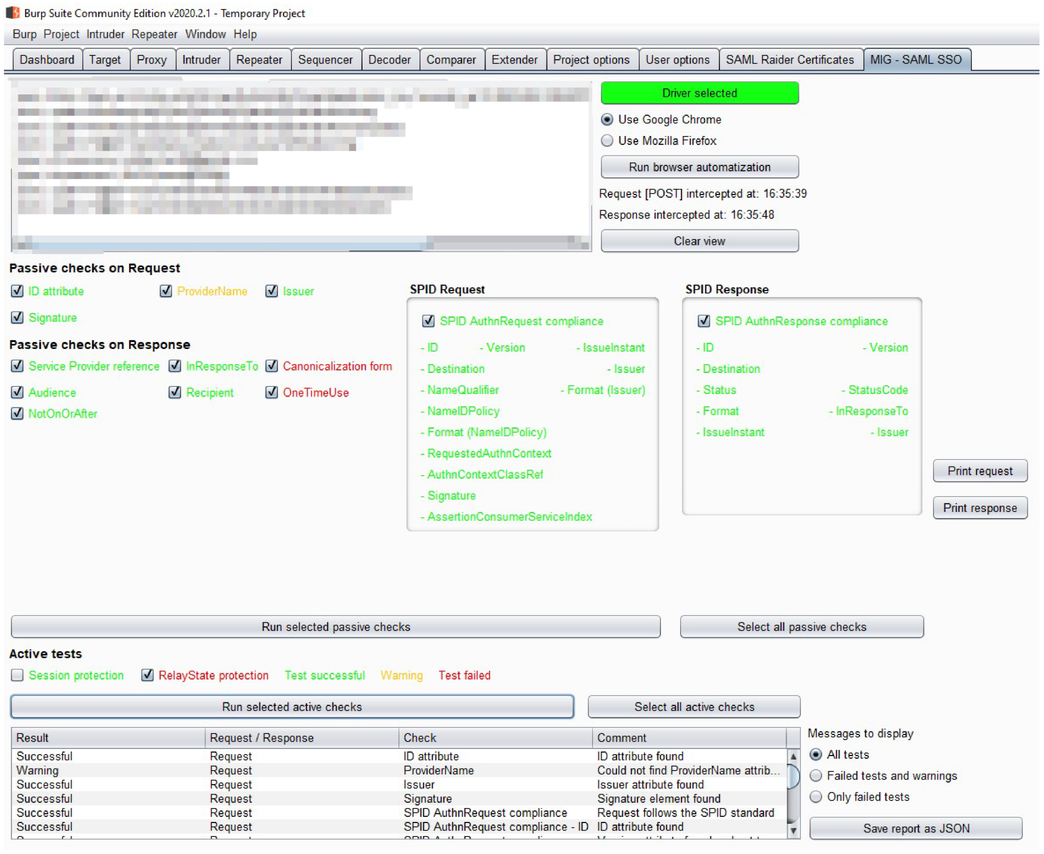
Messages to display: [✓ All tests] Failed tests and warnings Only failed tests

Result	Request / Response	Check	Comment
Successful	Request	ID attribute	ID attribute found
Warning	Request	ProviderName	Could not find ProviderName attrib...
Successful	Request	Issuer	Issuer attribute found
Successful	Request	Signature	Signature element found
Successful	Request	SPID AuthnRequest compliance	Request follows the SPID standard
Successful	Request	SPID AuthnRequest compliance - ID	ID attribute found

Figure 6: The results view on IPZS use case (test environment).

5.2 SPID use case

In this case the tool has been used to analyze the messages exchange in a login process using, as Service Provider, a public service that supports SPID authentication. For this purpose a personal account has been used.


Burp Suite Community Edition v2020.2.1 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options SAML Raider Certificates MIG - SAML SSO
Driver selected
Run browser automatization
Clear view
Passive checks on Request
Passive checks on Response
SPID Request
SPID Response
Print request
Print response
Run selected passive checks
Select all passive checks
Active tests
Session protection
RelayState protection
Test successful
Warning
Test failed
Run selected active checks
Select all active checks
Messages to display
Save report as JSON

[redacted]

[✓ Use Google Chrome] Use Mozilla Firefox

Request [POST] intercepted at: 16:35:39
Response intercepted at: 16:35:48

Passive checks on Request: [✓ ID attribute] [✓ ProviderName] [✓ Issuer] [✓ Signature]

Passive checks on Response: [✓ Service Provider reference] [✓ InResponseTo] [✓ Canonicalization form] [✓ Audience] [✓ Recipient] [✓ OneTimeUse] [✓ NotOnOrAfter]

SPID Request: [✓ SPID AuthnRequest compliance] - ID - Version - IssueInstant - Destination - Issuer - NameQualifier - Format (Issuer) - NameIDPolicy - Format (NameIDPolicy) - RequestedAuthnContext - AuthnContextClassRef - Signature - AssertionConsumerServiceIndex

SPID Response: [✓ SPID AuthnResponse compliance] - ID - Version - Destination - Status - StatusCode - Format - InResponseTo - IssueInstant - Issuer

Active tests: Session protection [✓ RelayState protection]

Messages to display: [✓ All tests] Failed tests and warnings Only failed tests

Result	Request / Response	Check	Comment
Successful	Request	ID attribute	ID attribute found
Warning	Request	ProviderName	Could not find ProviderName attrib...
Successful	Request	Issuer	Issuer attribute found
Successful	Request	Signature	Signature element found
Successful	Request	SPID AuthnRequest compliance	Request follows the SPID standard
Successful	Request	SPID AuthnRequest compliance - ID	ID attribute found

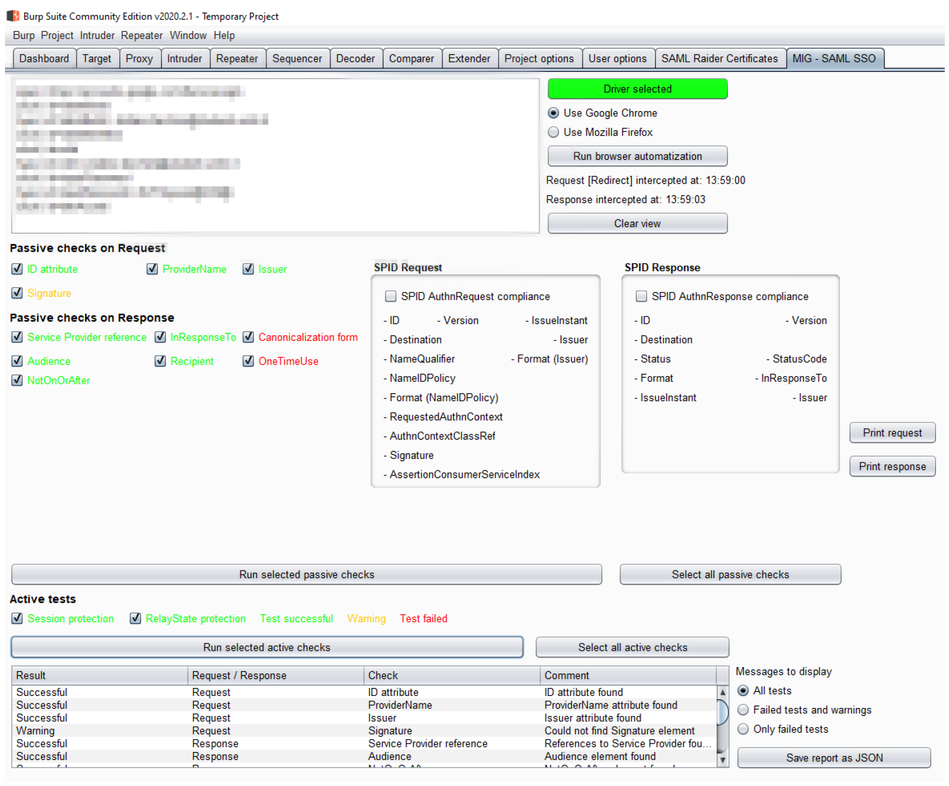
Figure 7: tool results on a SPID use case.

We can note, from this use case (Figure 7), that:

- ProviderName: a warning is displayed, since this attribute is missing but it is optional. However, it works as a friendly name for the SP and it might be used for some secondary reasons by the SAML authorities, hence it is good practice to include it.
- Canonicalization form: the detected canonicalization algorithm has been observed to be unsafe. More advanced analysis should be done in order to understand if the scenario suffers this vulnerability or hidden protection mechanisms are working to offer the proper security.
- OneTimeUse: the OneTimeUse element is missing. This element explicitly tells the receiving SAML authority that this message shall be consumed only the first time it is received. Although most scenarios implicitly apply this rule, there might be a few where this is not true, hence this element should always be included.
- RelayState: the value of the RelayState attribute can be tampered with no security warnings. In our case, we were able to edit the RelayState value of the request: the response was surprisingly containing the tampered value, with no error or warning of any type. However the authentication flow proceeded through the safe and original path, meaning that further protection are involved to ignore the tampered value.

5.3 SAML SSO use case

We also tested our plugin on a generic SAML SSO implementation. After inserting the apposite authentication trace and having it executed, we can run the passive and active checks. Note that in this case we do not check for the SPID compliance, since we know that this environment is not SPID-related.



Burp Suite Community Edition v2020.2.1 - Temporary Project

Burp Project Intruder Repeater Window Help

Dashboard | Target | Proxy | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | SAML Raider Certificates | MIG - SAML SSO

[redacted]

Driver selected

[✓ Use Google Chrome]
Use Mozilla Firefox

Run browser automatization

Request [Redirect] intercepted at: 13:59:00

Response intercepted at: 13:59:03

Clear view

Passive checks on Request

✓ ID attribute ✓ ProviderName ✓ Issuer

✓ Signature

Passive checks on Response

✓ Service Provider reference ✓ InResponseTo ✓ Canonicalization form

✓ Audience ✓ Recipient ✓ OneTimeUse

✓ NotOnOrAfter

SPID Request

☐ SPID AuthnRequest compliance

- ID - Version - IssueInstant
- Destination - Issuer
- NameQualifier - Format (Issuer)
- NameIDPolicy
- Format (NameIDPolicy)
- RequestedAuthnContext
- AuthnContextClassRef
- Signature
- AssertionConsumerServiceIndex

SPID Response

☐ SPID AuthnResponse compliance

- ID - Version
- Destination
- Status - StatusCode
- Format - InResponseTo
- IssueInstant - Issuer

Print request

Print response

Run selected passive checks

Select all passive checks

Active tests

✓ Session protection ✓ RelayState protection Test successful Warning Test failed

Run selected active checks

Select all active checks

Result	Request / Response	Check	Comment
Successful	Request	ID attribute	ID attribute found
Successful	Request	ProviderName	ProviderName attribute found
Successful	Request	Issuer	Issuer attribute found
Warning	Request	Signature	Could not find Signature element
Successful	Response	Service Provider reference	References to Service Provider fou...
Successful	Response	Audience	Audience element found

Messages to display

[✓ All tests]
Failed tests and warnings
Only failed tests

Save report as JSON

Figure 8: tool results on a SAML use case.

All checks pass successfully, except for those ones (Figure 8):

- Signature: we observe that the Signature check returns a warning, meaning that the request signature could not be found. Checking the intercepted messages we can actually see that there is no signature. However, we recall that it is not mandatory, explaining the yellow warning instead of the red failure.

- Canonicalization form: the detected canonicalization algorithm has been observed to be unsafe. More advanced analysis should be done in order to understand if the scenario suffers this vulnerability or hidden protection mechanisms are working to offer the proper security.

- OneTimeUse: the OneTimeUse element is missing. This element explicitly tells the receiving SAML authority that this message shall be consumed only the first time it is received. Although most scenarios implicitly apply this rule, there might be a few where this is not true, hence this element should always be included.

Chapter 6

Conclusions and Future Work

Given the high number of services available online, the authentication through the network has become more and more popular over the years. To simplify this process, several mechanisms that allow one user to access more platforms with only one set of credentials arose: those are called SSO services. In this thesis we focused on one of the most popular ones, SAML. Our choice to focus on this one comes from the high number of scenarios that rely on this protocol: suitable mainly for enterprise contexts, it is applicable where one account can identify the very same user on several related services. It is, for instance, used in Italy to access Public Administration services. Moreover, SAML is also involved in eIDAS regulation, which allows different nations' SSO to intercommunicate to identify a foreign person in another country.
After analyzing the current situation, we defined which vulnerabilities where already tested by other tools for security testing and which ones were instead left unchecked. Although SAML SSO is a popular scenario for already mentioned purposes, the number of threats is still high, potentially compromising confidentiality, availability and integrity of the related data and paving the way for various attacks such as impersonation and Cross Site Request Forgery.
The proposed solution performs the appropriate checks to reduce the possibility of security flaws, executing passive and active tests on the intercepted SAML messages. Once one of those is completed, the result will be shown to the user that now has a way to understand if there is any problem and which elements are causing it.
After the design and implementation of the plugin, we proceeded to the testing and validation phase. After analyzing several cases, during which each test has been double checked to verify its genuineness, we can state that the results were the expected ones.
The developed solution can be used by cybersecurity experts that want to test their implementation of a SAML SSO environment, as well as by pen-testers who are interested in investigating a third-party scenario and want to analyze its characteristics and possible flaws.
As future work, the developed plugin can be improved implementing new checks and refining the present ones. For instance, the RelayState test works assuming that, after applying the desired tampering, the server returns an HTTP message with an error status code. A possible enhancement is to check also other behaviors such as an error page or the redirect to a fixed URL. Another aspect that can be improved is the User Interface: its restyling can be considered to offer a better user experience and more natural usability.

Bibliography

[1] a9t9. UI.Vision RPA. https://ui.vision/.

[2] Ahmad Abolhadid. SamlReQuest. https://portswigger.net/bappstore/ec94f32a5b1f467bbb1459398818156f.

[3] Anitian. Anitian's Blog. https://www.anitian.com/all-posts/.

[4] Roland Bischofberger and Emanuel Duss. SAML Raider. https://portswigger.net/bappstore/c61cfa893bb14db4b01775554f7b802e.

[5] The Duo Blog. Duo Finds SAML Vulnerabilities Affecting Multiple Implementations. https://duo.com/blog/duo-finds-saml-vulnerabilities-affecting-multiple-implementations.

[6] Cloudflare. What is SAML? — How SAML authentication works. https://www.cloudflare.com/learning/access-management/what-is-saml/.

[7] The MITRE Corporation. Common Vulnerabilities and Exposures. https://cve.mitre.org/.

[8] Ministero dell'Interno. Carta d'Identita Elettronica. https://www.cartaidentita.interno.gov.it/.

[9] Agenzia Per L'Italia Digitale. SPID - Regole Tecniche. https://docs.italia.it/italia/spid/spid-regole-tecniche/it/stabile/index.html.

[10] Duo. The Duo Blog. https://duo.com/blog.

[11] Epi052. Epi052's blog. https://epi052.gitlab.io/notes-to-self/blog/.

[12] Tim Guenther et al. EsPReSSO. https://portswigger.net/bappstore/e1d08d4ab1ea4c17be3431d7d2b20b30.

[13] Europa. eIDAS SAML Message Format - europa EU. https://ec.europa.eu/cefdigital/wiki/download/attachments/82773108/eidas_message_format_v1.0.pdf?version=1&modificationDate=1497252920416&api=v2.

[14] Jamsheert. Difference between IDP initiated SSO and SP initiated SSO. http://jamsheert.blogspot.com/2015/08/difference-between-idp-initiated-sso.html.

[15] Katalon. Katalon Studio. https://www.katalon.com/homepage/.

[16] OASIS. Security Assertion Markup Language (SAML) v2.0 Technical Overview. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html#5.1.Web%20Browser%20SSO%20Profile%7Coutline.

[17] Valentina Odorizzi. *Progettazione e sviluppo di uno strumento per l'analisi automatica di vulnerabilita Missing XML Validation in SAML SSO*. Bachelor's thesis, DISI, Università di Trento, 2018.

[18] OneLogin. Examples/AuthNRequest. https://developers.onelogin.com/saml/examples/authnrequest.

[19] Giulio Pellizzari. *Design and implementation of a tool to detect Login Cross-Site Request Forgery in SAML SSO: G Suite case study*. Bachelor's thesis, DISI, Università di Trento, 2019.

[20] Agenzia per l'Italia Digitale. SPID: il sistema pubblico di identità digitale. https://www.agid.gov.it/it/spid-come-ottenere-identit%C3%A0-digitale.

[21] SSO-Attacks. Ruhr university bochum. https://www.sso-attacks.org/Main_Page.

[22] sso attacks.org. XML Signature Wrapping/Description. https://www.sso-attacks.org/XML_Signature_Wrapping.

[23] Burp Suite. PortSwigger. https://portswigger.net/burp.

[24] Lorenzo Tait. *A CUSTOMIZED THREAT MODELING FOR SECURE DEPLOYMENT AND PENTESTING OF SAML SSO SOLUTIONS*. Bachelor's thesis, DISI, Università di Trento, 2019.

[25] Open Web Application Security Project. OWASP Foundation. https://owasp.org/.

[26] ws attacks.org. XML Signature Exclusion. https://ws-attacks.org/XML_Signature_Exclusion.

[27] ws attacks.org. XML Signature Wrapping. https://www.ws-attacks.org/XML_Signature_Wrapping.