Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer Science

FINAL DISSERTATION

Development of SAML Testsuite to facilitate Automatic Pentesting

Supervisor

Silvio Ranise

Student

Sofia Zanrosso

Co-Supervisor(s)

Andrea Bisegna

Roberto Carbone

Academic year 2020/2021

Acknowledgements

*I would like to dedicate this thesis to my family
for their precious help and support, for their
unconditional love and for the sacrifices
they made for me.*

*I sincerely want to thank Andrea and Roberto
for their priceless advice.
After awkward years, for the first time I felt
my contribution was important.
Being a "woman in tech" has never been easy,
but their passion and support helped me in
following my dreams.*

Contents

Abstract

As the process of digitalisation is increasing, the amount of managed private or sensitive information has become a crucial aspect. In particular, the adoption of appropriate technologies to securely authenticate and authorise users plays a fundamental role. One of the mechanisms that assists users in password management is Single Sign-On, an authentication service that provides a way to access different web applications with just a set of credentials.

Since misconfigurations in password management systems enable malicious users to expoloit authentication flows to perform several attacks, the purpose of this thesis is to provide a facilitate way, even for those who are not directly involved in the security field, to perform automatic pentesting w.r.t. the Security Assertion Markup language, an XML standard used to control the exchange of messages regarding users' authentication and authorisation.

The first step is a Cyber Threat Intelligence phase focusing on vulnerabilities and attack and, as potentially harmful vulnerabilities are presented and described, a mapping between attacks and existing pentesting tools is carried out. These tools allow an experimental analysis in real world use cases on SPID scenarios which leads to the discovery of important flaws. The goal of the thesis is reached thanks to the development of a proper tool (by a colleague) capable of performing a json testsuite containing the results of the Cyber Threat Intelligence phase.

The main contribution of this thesis are the research and analysis of the current vulnerability situation regarding SAML SSO implementations and existing pentesting tools covering such vulnerabilities and attacks. Then the experimental analysis on use case scenarios, followed by the development of the json testsuite covering almost all the attacks, together with a constant monitoring on the problems and errors of the developed plugin in order to solve them. Also a comparison phase between the existing tools and the new one which results confirmed the validity of our choices.

Glossary

AgID	Agenzia per l'Italia digitale
CSRF	Cross Site Request Forgery
eIDAS	Electronic Identification, Authentication and trust Services
IdP	Identity Provider
SAML	Security Assertion Markup Language
SP	Service Provider
SPID	Servizio Pubblico Identità Digitale
SSO	Single Sign-On
SSRF	Server Side Request Forgery
XEA	XML Encryption Attack
XML	Extensible Markup Language
XSE	XML Signature Exclusion
XSS	Cross Site Scripting
XSW	XML Signature Wrapping
XXE	XML External Entity

Keywords: Cyber Threat Intelligence, Pentesting Tools, Testsuite.

1 Introduction

1.1 Context and motivations

The process of digitalisation has allowed companies to simplify, speed up and make each process more accurate. This has resulted in a clear improvement in efficiency and performances, with the possibility of guaranteeing products and services of great quality [24]. Also in public administration, digitalisation aims to improve the access to goods and services both for citizens and businesses, but also aims to take advantage of the potential of ICT technologies to promote innovation, sustainability, economic growth and progress. Everyday, public administrations, but also privates, manage sensitive information, in particular credentials which are the key factor to protect private data.

This innovative way to relate with users, however, deals with the growing number of cyber-threats. For this reason, the security requirements should be considered essential when public administrations design and implement online services for citizens. Many cyber-attacks focus on digital identities and, if violated, the harm might be crucial for users and organizations.

In this thesis we are focusing on Security Assertion Markup Language (SAML) that supports Single Sign-On (SSO). SSO allows users to be authenticated for multiple applications at once in order to reduce the fatigue of the process and provide a better user experience. This technological solution adopted by many public administrations is a good trade-off between security and usability.

Relevant examples of the use of SAML SSO are SPID [19] for the Italian government and eIDAS [18] as an European regulation. Sistema Pubblico Identità Digitale (SPID) is the access key to public administrations' digital services. A unique set of credentials which represent the personal digital identity of each citizen, with which it is recognized in order to make personalized and secure use of digital services [20]. The electronic IDentification Authentication and Signature (eIDAS) regulation aims at providing a common normative basis for secure electronic interactions between citizens, businesses and public administrations and at increasing the security and effectiveness of electronic services and e-business and e-commerce transactions in the European Union [18].

1.2 Problem

Thanks to SSO, a user can perform the authentication process once to the IdP: if this process is successful, he/she is granted to access all related SP in a transparent way. The communication between the SP and IdP is made up of SAML messages sent over the Internet [7]. Misconfigurations in SAML SSO scenarios enable malicious users to exploit authentication flows to perform several attacks such as Denial of Service [16], Cross Site Scripting [15] and others.

Even though there already are some pentesting tools that work in this field, the amount of vulnerabilities is constantly growing and some untested flaws still remain. Organisations may be using tools that detect some but leave other vulnerabilities undetected. Since this is critical to a meaningful definition of a security test plan, an exhaustive and complete testsuite should be created as a strategy to ensure that SAML SSO solutions are tested to combat unfolding security threats and vulnerabilities.

In order to address this problem, we proceeded improving the threat modeling for SAML SSO through a careful research into new vulnerabilities and attacks. We also provided an examination of the existing tools for automatic pentesting that results in false positives causing a bad estimation of the security of real world use cases. A more accurate analysis was necessary in order to identify the arisen problems.

The issues, emerged from the incorrectness of analysed pentesting tools, enlighten the absolute necessity of having a unique plugin capable of indentifying SAML SSO vulnerabilities in a facilitated, automatic and more accurate way. So our main aim was to provide for the creation of a secure methodology allowing cyber-security experts but also beginners to perform automatic pentesting in a easier manner.

1.3 Contributions

The contributions of this thesis are the following:

- The aim of this thesis is the development of a SAML testsuite in order to facilitate automatic pentesting w.r.t. SAML. The first phase consisted in collecting vulnerabilities and attacks in order to test in an automatic way if a particular SP or IdP is vulnerable to the given attacks. The sources used for collecting information are scientific reports about eIDAS vulnerabilities and attacks [23], and some colleagues' bachelor and master theses about SAML inspection [25, 3], and plugins [7]. *Cyber Threat Intelligence for SAML SSO* plays a fundamental role in this phase as the methodology for which the detection of additional vulnerabilities and attacks is possible. Since the number of risks is constantly growing, in order to achieve our goal, we considered as necessary a web search of the latest state-of-the-art novelties in the mentioned field. The result of Cyber Threat Intelligence is the increase in coverage of the known threats.

- The Cyber Threat Intelligence phase allowed us to map the found attacks with existing tools. We focused, due to its diffusion, on Burp Suite [26] plugins with the purpose of providing actual tests, later used to perform an experimental analysis on real world use cases. The chosen plugins are SAML Raider [2], EsPReSSO [1] and Micro-Id Gym [21, 7].

- By mapping these series of tests with the corresponding tools, we performed an *Experimental Analysis on Real World Use Ceses* based on SPID scenarios in order to prove the correctness of the existing tools: the results revealed some false positives. Also serious vulnerabilities were detected and, in order to communicate the found secuirity flaws, confidential security reports were drafted for a proper responsible disclosure [27].

- The latter, but most improtant, phase of this thesis was the creation of a json testsuite in order to unify vulnerabilities, attacks and to correct tools' flaws. Thanks to the development of a proper plugin [4], I managed to improve this tool, communicating problems and comparing it with other existing tools. Almost all the collected vulnerabilities and attacks of the Cyber Threat Intelligence phase were converted for the creation of the json testsuite, increasing the coverage of the feasible tests. The results in terms of effectiveness correctly reflected our expectations thus allowing to facilitate automatic pentesting and to provide a support tool to allow developers to test first-hand their implementations.

1.4 Structure of the thesis

The thesis is divided in the following way:

- *Chapter 2*: Summary of the background concept of Single Sign-On, SAML SSO with the relative login flow, and SPID and eIDAS use cases.

- *Chapter 3*: Illustration of analysis on latest news related to vulnerabilities and attacks (Cyber Threat Intelligence for SAML SSO) and description of the found novelties.

- *Chapter 4*: Description of the existing tools that allow checking the integrity and possible flaws of a SAML scenario. Selection of tests in order to analyse existing solutions related to SAML SSO and SPID use cases.

- *Chapter 5*: Experimental analysis on real world use cases, comparison with the existing tools, found false positives, found vulnerabilities, responsible disclosure and implementation of a test-suite to facilitate automatic pentesting.

- *Chapter 6*: Final considerations and possible improvements to enhance the work described here.

2 Background

In this chapter some basic definitions are reported, as well as fundamental concepts in order to fully understand the context of the provided work. Here we mention Single Sign-On, the Security Assertion Markup Language and its login flow, and some use cases.

2.1 Single Sign-On

Single Sign-On (SSO) is an authentication scheme that allows a user to log in with a single set of credentials to any of several systems. It allows the user to log in once and access services without re-entering authentication factors.

Authentication is the process of identifying a user to provide access to a system. In this process, user credentials are validated within three categories of authentication factors. For a positive authentication, elements from at least two factors should be verified. The three factors are:

- *Knowledge factors*: something the user knows, like password or personal identification number.
- *Ownership factors*: something the user posses, such as ID card or cellphone.
- *Inference factors*: something the user is or does, such as fingerprint or retinal pattern.

The process of authentication is different from authorisation. Whereas authentication is the process of verifying that *"you are who you say you are"*, authorisation is the process of verifying that *"you are permitted to do what you are trying to do"*. While authorisation often happens immediately after authentication, this does not mean authorisation requires authentication.

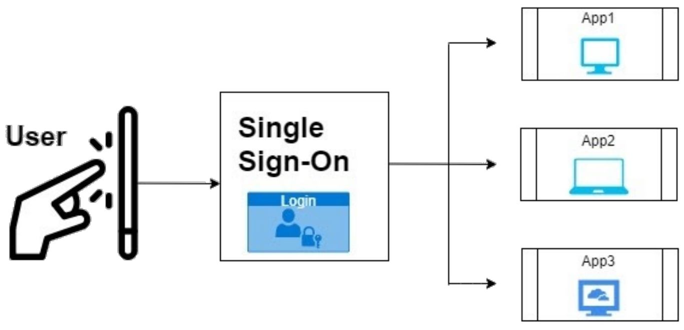


Figure 2.1: SSO [11]

2.2 SAML SSO

Security Assertion Markup Language (better known as its acronym, SAML) [9] is one of the most widely used open standard for authentication and authorisation between multiple parties. It is one of the protocols that give users the Single Sign-On (SSO) experience for applications. SAML is a mean by which is possible to exchange authorisation and authentication information between services. It is frequently used to implement SSO solutions where the user can seamlessly connect to a set of services that act as the single source of identity without creating a multitude of accounts. Without SSO on the user side there is the necessity to remember several logins/passwords or, even worse, using the same ones.

In SSO protocols, three types of entities can be distinguished:

- *Principal*: it is the user trying to authenticate. You can think of this as the actual human behind the screen.

- *Identity Provider (IdP)*: it is the service that serves as the source of identity information and authentication decision. Identity providers authenticate clients and return identity information to Service Providers.

- *Service Provider (SP)*: are the services that are requesting authentication and identity information about the client. SPs take authentication responses received from identity providers and use that information to create and configure sessions. In other words, it is an application that offers a SSO mechanism for its users to login and access its resources.

2.2.1 SSO login flow

SSO can be started either SP-side (Figure 2.2) and IdP-side (Figure 2.3). In the first case the user goes to the SP (for example on *sp.example.com*), if he/she has no active session, the SP redirects him/her to the IdP which will ask the user to connect. After entering his/her correct login credentials, the user will be redirected to the SP with a valid session.

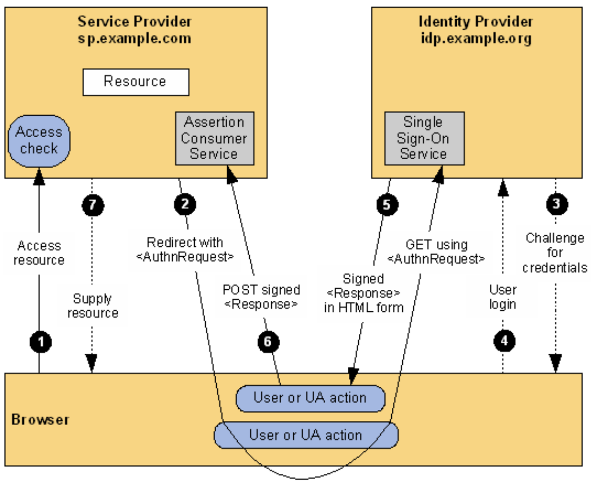


Figure 2.2: SP-side [28]

In an IdP-initiated use case, the IdP is configured with specialized links that refer to the desired SP. So instead of visiting the SP directly, the user accesses the IdP site and clicks on one of the links to gain access to the remote SP, he/she enters his/her credentials, and if the user is a regular one is now authenticated and can be redirected to the SP [28].

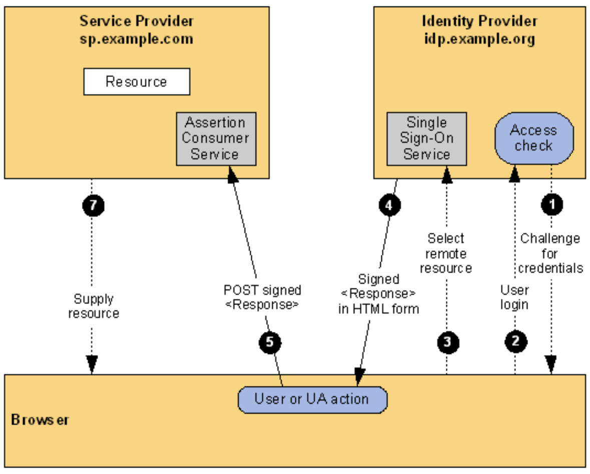


Figure 2.3: IdP-side [28]

2.3 Use cases

In this section a brief description of the common use case scenarios is presented: SPID, the Italian public system related to digital identity and eIDAS, the European regulation on electronic identification.

2.3.1 SPID

The Public Digital Identity System (SPID) is a digital identity consisting of a pair of personal credentials, with which it is possible to access online services of the public administration and private members. It is possible to use SPID from any device whenever the *"Enter with SPID"* button is found [19]. SPID implements SSO service in order to allow citizens to use only one set of credentials to access any public administration's online service. In the SPID authentication process, users authenticate to SPs, which are public or private organizations providing a service to authorised SPID users. In relation to SAML SSO profile, SPID implements the SP-initiated SSO.

2.3.2 eIDAS

The eIDAS regulation aims at providing a common normative basis for secure electronic interactions between citizens, businesses and public administrations and at increasing the security and effectiveness of electronic services and e-business and e-commerce transactions in the European Union [18]. European countries developed strong authentication schemes based on electronic identification (eID) cards. The main goal is to provide different services, called eID services, for which access should be provided for citizens and organizations by using information already available on eID cards. European countries have worked on developing their own authentication schemes based on different technologies. This made authentication between eID services of European countries impossible so, in 2014 an eIDAS regulation was released by the European Commission, defining a high level overview of cross-country eIDAS authentication. It does not provide a standalone SSO solution, but rather specifies a SAML based compatibility layer between different eID implementations. The components facilitating the cross-border information exchange are called eIDAS-Nodes [23]. As SAML SSO is utilized as a standard on the nodes, eIDAS was considered in this work because the purpose of this thesis is also applicable to an European reality.

3 Cyber Threat Intelligence for SAML SSO

Cyber Threat Intelligence consists in collecting, analysing and disseminating information related to a company's operation in cyberspace and physical space. The analysis' main purpose is to help keep situational awareness about current and arising threats. It is designed to inform all levels of decision makers [10]. In this thesis, Cyber Threat Intelligence for SAML SSO plays an important role in finding and analysing new vulnerabilities and attacks in order to increase the amount of known cases.

3.1 Sources

The goal of this analysis phase is to offer a list of threats starting from a colleague's previous work [25]. The sources that have been analysed for this phase are the following:

- Scientific report about the security analysis of eIDAS that enlighten the tests performed on eIDAS services and so on SAML SSO protocol [23].
- Bachelor's thesis based on the implementation of an automated tool for checking SAML vulnerabilities and SPID compliance [7].
- Master's thesis regarding a security test plan for SAML SSO mainly focused on the SPID use case [3].
- Web searching on the newest found vulnerabilities as the cyber threats is rapidly evolving and as the number of attacks is increasing every year. The main sources are [12, 5, 13].

3.2 Vulnerabilities and Attacks

A increased level of coverage has been achieved after analysing the resources reported in Section 3.1. The found novelties (attacks and relative vulnerabilities), compared to the analysis carried out by previous colleagues [25, 7, 3], are reported in the following sections.

3.2.1 XML Encryption Attack

XML encryption defines a standard for how to encrypt XML documents with high granularity, starting from encrypting the entire XML document or just one element [29]. *XML Encryption Attack (XEA)* consists in modifying the SAML message with a different encryption key. After the XEA is performed, the assertion related to data is fully encrypted with the new key. Successful attacks against XML Encryption would undermine the confidentiality goals of the encrypted SAML assertions [23].

3.2.2 Certificate Faking

Certificate Faking is the process of replacing the `<Signature>` element of an XML message with a self-signed certificate. If Certificate Faking is performed, the trust relationship between the SP and the IdP is tested. This relationship should be verified each time a SAML message is received. The vulnerability related to this attack consists in validating the signature even if an untrusted key is used. If a single SAML service is vulnerable to Certificate Faking, the authentication scheme is broken because a malicious user is able to forge arbitrary messages and identities [23].

3.2.3 Authentication Bypass Attack

Authentication Bypass Attack refers to a critical security vulnerability affecting SAML Certificate management across a range of Palo Alto Networks devices. This is not a general kind of attack but it refers to a specific implementation, it impactes customers using Palo Alto devices with IdPs that rely on the SAML protocol and who are using self-signed certificates. Any customer with a vulnerable platform, using a self-signed certificate faces the risk that an attacker may be able to bypass their authentication and access sensitive resources [22].

3.2.4 Session Fixation Attack

Session Fixation Attack is a class of Session Hijacking, which steals the established session between the client and the web server after the user logs in. After the log in to the web application using the provided session ID, the attacker can use the valid session ID to gain access to the victim's account. There are several techniques to execute the attack, it depends on how the Web application deals with session tokens. Some of the most common techniques are [17]:

- Session token in the URL argument for whom the session ID is sent to the victim in a hyperlink and the victim accesses the site through the malicious URL.
- Session token in a hidden form field for which the victim is tricked to authenticate in the target web server, using a login form developed for the attacker.

The vulnerability related consists in insufficient web application security and bad coding practices associated with session management, in particular by assigning the session ID before the user is logged in. Some of the ways to avoid Session Fixation attacks consist in invalidating session IDs after a timeout or changing the session ID right after the user logs in [14].

3.2.5 Clickjacking

Clickjacking (or *User-Interface Redressing*) consists in tricking the user into unknowingly execute actions of the attacker's choice. In order to perform this attack, features like transparent `iFrames` are involved, and are often combined with some incentive for the user to induce the intended action. Successful Clickjacking attacks may result in data exposure or an account to become compromised. It can be used to lure the victim into unknowingly authenticating at the IdP and authorising the access to sensitive resources. This vulnerability can be mitigated by adding a `X-Frame-Options` header to relevant pages.

4 Mapping Attacks to Tools

The Cyber Threat Intelligence research phase carried out in Chapter 3 allowed us to collect vulnerabilities and attacks in order to extend the collection of known existing threats. In this chapter, we focused on mapping the known vulnerabilities and attacks with existing pentesting tools. A brief description of such tools is reported and, in order to move from theory to practice, we focused on providing, through the cited tools, an automatic way to test if IdPs and SPs are vulnerable to the mentioned attacks.

4.1 Software and Plugins

Since our aim was to increase the coverage of tested vulnerabilities and attacks, we consider a few SAML SSO security tools with the intention of understanding their features. In this phase we considered Burp Suite [26] and its plugins.
The examined tools are the following:

- *Burp Suite*: A platform for performing security testing of web applications. Its various tools work to support the entire testing process, from initial mapping and analysis of an application's attack surface, to finding and exploiting security vulnerabilities [26].
- *Micro-Id-Gym*: A tool for pentesting Identity Management (IdM) Protocols which supports two main activities, namely the creation of sandboxes with an IdM protocol deployment and the pentesting of IdM protocol deployments. It offers on the one hand an easy way to configure the preoduction environment where pentesters can experience IdM solutions, performing attacks; and on the other hand a set of tools for the automatic security analysis of IdM protocols. It leads to perform passive and active tests on SAML Request and SAML Response to automatically detect security issues [21, 7].
- *EsPReSSO*: This Burp Suite extension processes and recognizes SSO protocols. It is incorporated with WS-Attacker, DTD-Attacker and XML-Encryption-Attacker integration while intercepting SAML messages [1].
- *SAML Raider*: A Burp Suite extension for testing SAML infrastructures. It contains two core features: a SAML message editor and an X.509 certificate manager [2].

4.2 Performed Tests

Thanks to the tools presented above, it was possible to target vulnerabilities and attacks mentioned in Chapter 3 through specific tests. In particular, taking cues from [21], we made a distinction between passive and active tests: the first type refers to vulnerabilities such as missing checks or incorrect generations of SAML messages; the latter instead refers to the reproduction of attacks.

4.2.1 Passive Tests

The following passive tests refers to vulnerabilities such as *(i)* Missing XML validation caused by an improper or a missing implementation of controls to diseable the acceptance of unrequested SAML messages; *(ii)* Missing protections against Cross Site Request Forgery attacks; *(iii)* Incorrect checks of SAML messages caused by a missing implementation of controls; and *(iv)* Incorrect generation related to an improper generation of elements or parameters in SAML messages.

In Table 4.1, a summary of the passive tests is reported with the relative SAML Message and tool with whom the tests can be performed.

All the passive tests were performed with the Micro-Id-Gym tool [21, 7]. They are intended to control the presence of elements within the SAML Request or SAML Response. The consequences of

Table 4.1: Collection of security passive tests.

Name	**SAML Message**	**Tool(s)**
`ID` attribute	Request	MIG
`ProviderName` attribute	Request	MIG
`Issuer` attribute	Request	MIG
`Signature` element	Request	MIG
SPID compliance	Request/Response	MIG
Service Provider references	Response	MIG
`Recipient` element	Response	MIG
`Audience` element	Response	MIG
`Destination` element	Response	MIG
`InResponseTo` element	Response	MIG
Canonicalization form	Response	MIG
`OneTimeUse` element	Response	MIG
`NotOnOrAfter` attribute	Response	MIG
`SubjectConfirmationData` element	Response	MIG

missing attributes without any further protection are the possibility of performing dangerous penetrations in the vulnerable system. A more exhaustive table of passive tests is reported in Appendix A.

4.2.2 Active Tests

In Table 4.2, a summary of the active tests is reported with the relative SAML Message and tool with whom the tests can be performed.

In order to better understand some of these tests and the correlated risks due to their exploits, a brief description is reported [25].

XML Signature Wrapping Attack. The attacker obtains a SAML assertion with a valid signature. An original element of the SAML message is moved within the document to a position unknown to the application logic. This location allows the element not to be processed by application logic, while the signature verification logic can still verify the element. There can be different permutations of the attack based on the location of the original element, the faked element and the signature. It is possible to perform 8 different variants of the attack exploiting several elements and positions of the SAML message.

Sensitive File Disclosure via Classic XXE using UTF-16 and UTF-7 variants. This attack is based on changing the XML encoding from UTF-8 to UTF-16 or UTF-7 in order to embed data which are located outside the main file into the XML document.

File-not-found exception Advanced XXE attack. The data extraction is performed by forcing an error on the server through the addition of an invalid URI to the request.

Exponential entity expansion attack (Billion Laugh). It consists of nesting entities within an XML `doctype` declaration, which can be easily exploited to construct a memory bomb. Similarly, the *Quadratic blowup attack* consists in defining one single large entity and to refer to it many times. Basically, an entity `&a;` with a significant length of characters, say 100,000 long, refers to `&a;` 100,000 times inside the root element.

Canonicalization attack. The attack consists in inserting comments into the value of the `NameID` element of the tester's assertion so that the XML document has still the same signature but the victim is going to be authenticated.

A more exhaustive table of active tests is reported in Appendix B.

Table 4.2: Collection of security active tests.

Name	**SAML Message**	**Tool(s)**
Integrity protection of `RelayState`	Request	MIG
Session protection	Request	MIG
Certificate Faking	Request/Response	SAML Raider
XSE	Request/Response	SAML Raider
XSW #1 - #8	Response	SAML Raider
Authentication Bypass attack	Response	SAML Raider
Exponential entity expansion	Request/Response	EsPReSSO
DOS via `Xinclude`	Request/Response	EsPReSSO
Classic XXE variant 1	Request/Response	EsPReSSO
Classic XXE attack using netdoc	Request/Response	EsPReSSO
Classic XXE using UTF-16	Request/Response	EsPReSSO
Classic XXE using UTF-7	Request/Response	EsPReSSO
Bypassing XXE restriction variant 1	Request/Response	EsPReSSO
Bypassing XXE restriction variant 2	Request/Response	EsPReSSO
XXE by abusing attribute values	Request/Response	EsPReSSO
XXE Out of Band variant 1	Request/Response	EsPReSSO
XXE Out of Band variant 2	Request/Response	EsPReSSO
XXE Out of Band variant 3	Request/Response	EsPReSSO
File-not-found exception variant	Request/Response	EsPReSSO
SchemaEntity via `schemaLocation`	Request/Response	EsPReSSO
SchemaEntity via `noNamespaceSchemaLocation`	Request/Response	EsPReSSO
SchemaEntity via `XInclude`	Request/Response	EsPReSSO
SSRF via XXE External general entities	Request/Response	EsPReSSO
SSRF via XXE External parameter entities	Request/Response	EsPReSSO
SSRF via XXE `schemaLocation`	Request/Response	EsPReSSO
SSRF via XXE `noNamespaceSchemaLocation`	Request/Response	EsPReSSO
SSRF via XXE `XInclude`	Request/Response	EsPReSSO
TCP scans	Request/Response	EsPReSSO
Local file read/write	Request/Response	EsPReSSO
Exponential entity expansion attack	Request/Response	EsPReSSO
Quadratic blowup attack	Request/Response	EsPReSSO
XEA	Request/Response	EsPReSSO
Delivery of unrequested resource	Request/Response	Burp Suite
CSRF attack	Request/Response	Burp Suite
Login CSRF	Request/Response	Burp Suite
XSS attack	Request/Response	Burp Suite
XSS for User Impersonation	Request/Response	Burp Suite
Canonicalization attack	Response	Burp Suite
Expired SAMLResponse Replay attack	Response	Burp Suite
Level of Assurance downgrade attack	Response	Burp Suite

5 New Plugin and Testsuite

Given the amount of available tests reported in Chapter 4, an experimental analysis in real world use cases has been performed. Its purpose was to test the security provided by the chosen SPs and IdPs, but also to compare the existing pentesting tools' efficiency.

In this chapter, the last phase of this thesis consisting in the creation of a complete json testsuite is reported. This was possible thanks to the development of a proper Burp Suite's extension by a colleague [4]. The aim was to unify vulnerabilities, attacks and to correct tools' problems found during the experimental analysis phase, but most of all to provide a facilitate way to developers to perform automatic pentesting on their implementations.

5.1 Experimental analysis in real world use cases and comparison

As mentioned before, one of the purposes of listing and cataloging the attacks and their vulnerabilities, is to run and test them in some specifically selected SPs and IdPs in order to compare the results obtained by the different tools. During the testing phase, some important findings arise. This paragraph collects the vulnerable scenarios, the found false positives, how the false positives have been exploited as a comparison between the tools, and the adopted solutions for the vulnerable situations. The experimental analysis was carried out in real world SPID use cases scenarios.

5.1.1 False positives

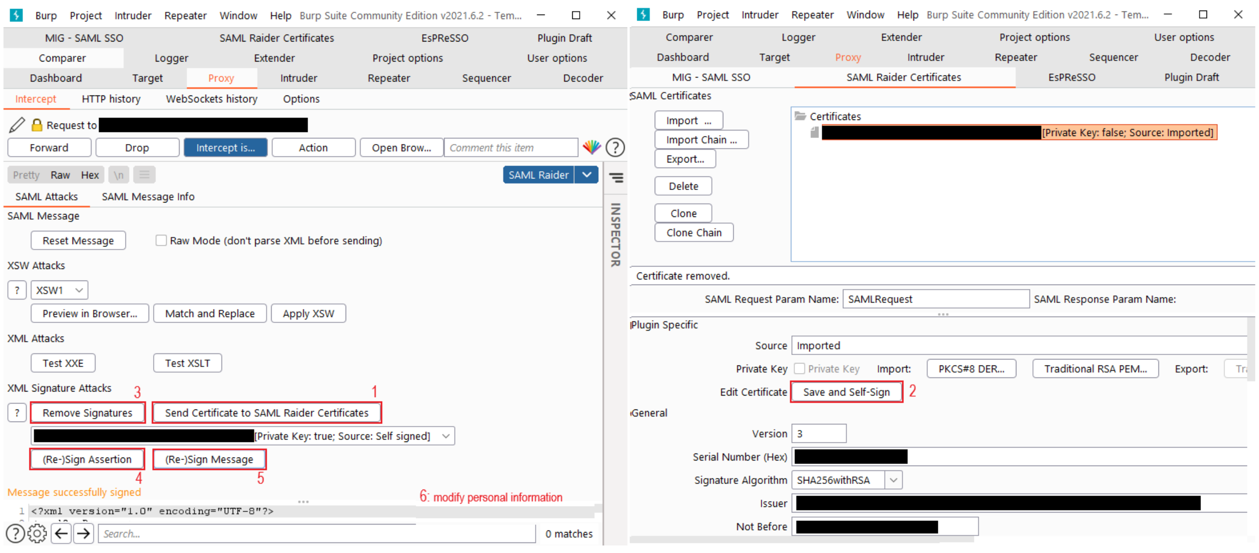
The main problems of this testing procedure were based on some false positives. They were all correlated to the Micro-Id Gym tool, in particular:

- *Canonicalization form*: every test results vulnerable to a canonicalization attack because of the unsafe canonicalization algorithm. However, a more accurate manual analysis with Burp Suite has been carried out in order to determine that the vulnerability was not actually relevant as further protections were involved.
- *NotOnOrAfter*: in a particular use case scenario the NotOnOrAfter test failed even if, examining the message, the attribute was set to the recommended value and was found both in the `SubjectConfirmationData` element and in the `Conditions` element, so no risk of Replay Attack has been detected.
- *OneTimeUse*: every test results vulnerable to a Replay Attack because of the failed OneTimeUse test. However, the Session Protection's active test showed that other protections for this vulnerability were involved.
- *RelayState protection*: a particular use case results vulnerable to RelayState tampering for the missing `RelayState` element. In our case, in checking this vulnerability, we were able to edit the `RelayState` value of the request. The response contained the tampered value with no error. However, the authentication flow proceeded through the original path, meaning that further protections were involved to ignore the tampered value [7].
- *Signature*: the SAML standard requires that all messages passed through insecure channels, such as the user's browser, must be signed. However, messages that pass through secure channels, such as an SSL/TLS, do not have to be. To prove this, a manually XSE attack was performed and the use case scenario results not vulnerable.

5.1.2 Found vulnerabilities

The success of this experimental analysis has led to important results. However, they were repeatedly checked and some of the them tested again manually with Burp Suite. In this paragraph, particular attention is given to the description of found vulnerabilities.

Certificate Faking attack resulting in Impersonation attack This attack consists in removing the `Signature` element in the SAML Response, modifying some user's personal information, signing the assertion again with a locally generated private key and applying a self-signed certificate derived from the original. The modified SAML Response is then sent by the browser to the SP (Figure 5.1). If the SP does not carry out the checks described above, the attacker is authenticated as the victim on the SP. A SP that behaves in this way does not verify correctly the digital signature that guarantees the integrity and authenticity of the SAML Response, therefore being vulnerable to an attack of Impersonation type. In relation to the severity and the type of breaches that can be made by changing personal information, a potential *very high* risk is estimated.



Request to [redacted]

[redacted] [Private Key: true; Source: Self signed]

Message successfully signed

```
<?xml version="1.0" encoding="UTF-8"?>
```

[redacted] [Private Key: false; Source: Imported]

Certificate removed.

SAML Request Param Name: SAMLRequest SAML Response Param Name:

Source: Imported

Private Key: ☐ Private Key

Version: 3

Serial Number (Hex): [redacted]

Signature Algorithm: SHA256withRSA

Issuer: [redacted]

Not Before: [redacted]

Figure 5.1: Certificate Faking with SAML Raider

Cross Site Request Forgery This attack consists in intercepting the SAML Response with the relative assertion, copying it on a text file, dropping the request and closing the browser. The next steps are opening a new session, creating another request, intercepting and replacing it with the replayed message previously intercepted. So the tester is logged into his/her account in a new session, unrelated with the original (Figure 5.2). It is important to notice that the replayed message can be forwarded with or without cookies, the result is the same. In a real world scenario, the victim believes that he/she is using his/her account whereas he/she is using the attacker's.

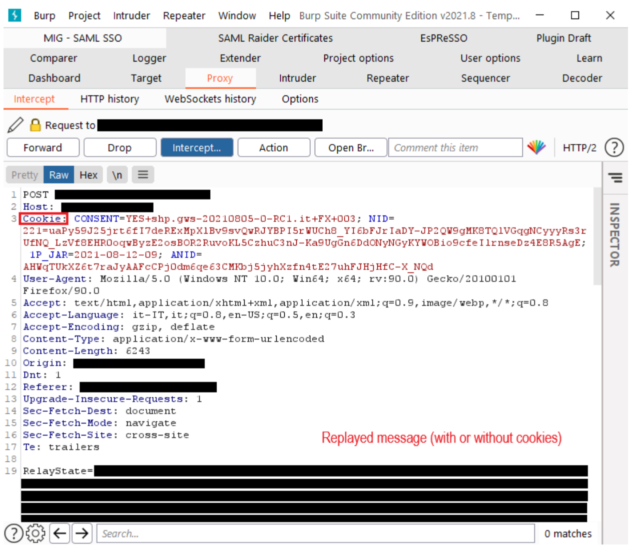
5.2 How the problems have been resolved

The amount of issues reported in the previous paragraph was important from a security point of view, but also for the necessity of finding a compact way for a successful experimental analysis. In this section, solutions to arised problems are specified: the important task of responsible disclosure on found vulnerabilities and how difficulty of this process in Italy; our practical way to solve false positives and to facilitate automatic pentesting.

5.2.1 Responsible disclosure

Responsible disclosure is a vulnerability disclosure model whereby a security researcher discreetly alerts a hardware or software developer to a security flaw in its most recent product release. The researcher then provides the vendor with an opportunity to mitigate the vulnerability before disclosing its existence to the general public [27].

Since some important vulnerabilities were found during the experimental analysis, confidential vulnerability reports were promptly sent to the relative vulnerable SP/IdP. In these reports also a guided


Request to [redacted]

```
POST [redacted]
Host: [redacted]
Cookie: CONSENT=YES+shp.gws-20210805-0-RC1.it+FX+003; NID=
221=uaPy59J25jrt6fI7deRExMpX1Bv9svQwRJYBPI5rWUCh8_YI6bFJrIaDY-JP2QW9gMK8TQ1VGqgNCyyyRs3r
UfNQ_LzVf8EHROoqwByzE2osBOR2RuvoKL5CzhuC3nJ-Ka9UgGn6DdONyNGyKYWOBio9cfeIlrnseDz4E8R5AgE;
 1P_JAR=2021-08-12-09; ANID=
AHWqTUkXZ6t7raJyAAFcCPjOdm6qe63CMKbj5jyhXzfn4tE27uhFJHjHfC-X_NQd
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101
Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 6243
Origin: [redacted]
Dnt: 1
Referer: [redacted]
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Te: trailers

RelayState=[redacted]
```

Figure 5.2: Cross Site Request Forgery with Burp Suite

description of the test reproduction and the risk of a possible attack were provided. For safety reason, the names of the vulnerable scenarios are not reported.

Italian laws on responsible disclosure. Unlike overseas realities, although reporting vulnerability is considered a good rule, Italian laws do not include responsible disclosure, indeed it is often ethical hackers who are condemned for abusive access to computer systems. The commitment of American government companies in the field of vulnerability research (also thanks to *Bug Bounty Programs*), demonstrates the importance of keeping high attention on cyber-security in all areas that are increasingly subject to attacks [6].

The *Italian Digital Team* firmly believes in responsible disclosure as the primary tool for communicating with ethical hackers, especially when the security, privacy, and personal information of citizens is at stake. A program of responsible disclosure can also facilitate the quick resolution of security problems with the aim of minimizing risks. Resolving flaws in a timely manner is crucial to reducing the exposure to malicious attackers [8].

Even if AgID reserved an important place to the cataloging of IT vulnerabilities through the implementation of the *National Vulnerability Database* in the *Triennial Plan for IT 2019-2021*, in Italy much progress is yet to be made in this area, primarily by developing a critical and result-oriented approach and secondly by regularizing a national responsible disclosure policy.

5.2.2 New plugin and testsuite

Despite these false positives, comparison between tools was successful, meaning that the existing ones are a viable resource for pentesters even though they usually target specific vulnerabilities.

As the collection of vulnerabilities is one of the aims of the previously mentioned *Triennial Plan for IT 2019-2021*, similarly our aim was not only to collect test definitions but also to create a complete testsuite based on existing tools, to facilitate automatic pentesting even for those who are novice with the disclosure of vulnerabilities.

In order to achieve this goal, a new Burp Suite's extension has been specially developed by a colleague [4]. This tool consists of a plugin able to recognize SAML messages, properly decoding the assertion depending on the used binding, performing penetration testing based on a json testsuite and to presents the results in a tabular way. The user interface, depicted in Figure 5.3, is visually divided in two sections:

- The upper one is the setup area where the authentication trace is inserted and the WebDriver file and browser is selected: this steps allow the system to automatically perform the login process.
- At the bottom we can find the area where the testsuite is described: once tests are performed, the results are shown in the *Test Suite Result* tab.

The plugin allows to perform passive tests in the order of a few seconds and all active tests within an hour. In this way, the previously employed transition times between tools have been greatly reduced, thus making it possible to analyse almost completely the vulnerabilities of the tested subjects.

My personal contribution to this phase consisted in converting almost all of the passive and active tests mentioned in Chapter 4 into a json testsuite in addition to a constant monitoring on the tool's possible errors with the purpose of communicating them directly to the developer. I also took care of the comparison between this plugin and the existing pentesting tool.

In order to perform the json testsuite I had to generate and save the authentication trace of every tested SP or IdP for whom secrecy must be maintained for a proper responsible disclosure 5.2.1. In addition a suitable driver had to be selected which lead to the track execution. The next steps consisted in importing and reading the json testsuite with a consequent execution of the testsuite.

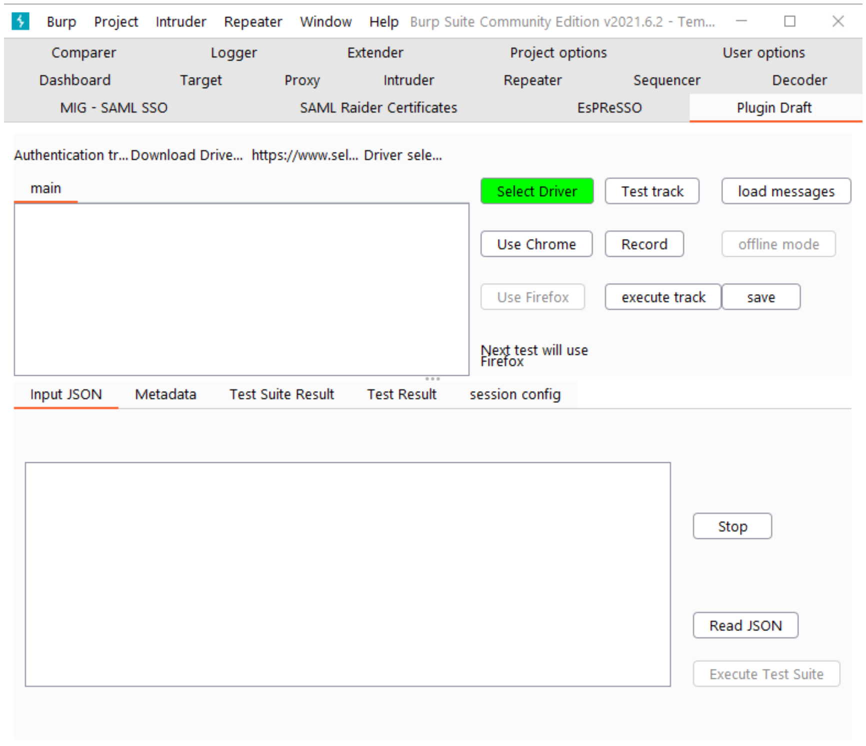


Figure 5.3: New plugin [4]

Practical examples of json testsuite. Each test suite shares a preamble (Listing 5.1) including *name* and *description*, followed by the list of tests included in the testsuite.

Listing 5.1: "Preamble of the testsuite"

```
{
  "test suite": {
    "name": "Example testsuite.",
    "description": "Performing some tests.",
    "filter messages": true
  }
  "tests": [
    ...
  ]
}
```

An example of a json testsuite with passive test is shown in Listing 5.2. Here we specify the *name*, *description*, *type* and the *operations* to perform, which refers the decodification of the parameter `SAMLRequest` and the checks in the url of the parameter `SAMLRequest` containing the `ProviderName` element.

Listing 5.2: “Testsuite of passive test - Check on ProviderName in Request”

```
{
  "test": {
    "name": "Check on ProviderName in Request",
    "description": "Checks whether the ProviderName element is present in
       the SAMLRequest.",
    "type": "passive",
    "operations": [
      {
        "message type": "saml request",
        "decode param": "SAMLRequest",
        "encoding":[
          "url",
          "base64",
          "deflate"
        ],
        "checks": [
          {
            "in": "url",
            "check param": "SAMLRequest",
            "contains": "ProviderName"
          }
        ],
        "message section": "url"
      }
    ]
  }
}
```

More complicated operations have to be done in order to perform an active test (Listing 5.3). It is necessary to set a session which is intercepted for editing (in this specific test) in the `SAMLResponse` parameter, the `saml2:NameID` tag adding comments to the value.

The *result* tag is used in active tests to specify the oracle to be used, which are the criteria to which the test is evaluated.

Listing 5.3: “Testsuite of active test - Canonicalization attack”

```
{
  "test": {
    "name": "Canonicalization attack (Attack)",
    "description": "This attack consists in inserting comments into the
       value of NameID element of the attacker's assertion so that the XML
       document still has the same signature but another user (the victim)
       is going to be authenticated.",
    "type": "active",
    "sessions": [
      "s1"
    ],
    "operations": [
      {
        "session": "s1",
        "action": "start"
      },
      {
        "action": "intercept",
        "from session": "s1",
        "then": "forward",
        "message type": "saml response",
        "message operations": [
          {
            "from": "body",
            "decode param": "SAMLResponse",
            "encoding": [
              "url",
              "base64",
              "deflate"
            ],
            "type": "xml",
            "edit tag": "saml2:NameID",
            "value": "nome.cognome<!-- newvalue -->@gmail.com"
          }
        ]
      }
    ],
    "result": "incorrect flow s1"
  }
}
```

6 Conclusions and Future Work

The high number of services made available to citizens has made online authentication an increasingly used process. As a result, the number of mechanisms needed for a user to access multiple platforms with a single set of credentials, in order to improve the user experience, has increased. In this thesis we focused on this aspect, on the services of SAML SSO and on their application in real cases such as SPID for Italy or eIDAS regulation at European level.

Given the performed analysis on the current threats related to SAML SSO, we improved the number of known test cases and associated the vulnerabilities that were already tested with the relative tools. The experimental analysis presented some issues and considering the amount of known vulnerabilities, a unique tool became the proper solution. This choice was also made in order to facilitate the pentesting processes. 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 a test is completed, the result is shown to the user that now knows if there is a possible problem and the element to which it is related.

The developed solution can be used by cyber-security experts that want to test their implementation of a SAML SSO environment, but also by pentesters who are interested in investigating and analysing a third-party scenario.

The goal of this thesis was to improve the list of known vulnerabilities and attacks, and to build a proper testsuite in order to create an automated and easier way for pentesting processes. The limitations encountered in the process of achieving the goal include the impossibility to provide a complete testsuite as the developed plugin still does not support certain types of testing methodologies.

As future work, the developed plugin and testsuite can be improved by completing the implementation of the missing active tests and by adding a more complete way of specify an oracle in the "result" element for each tested vulnerability.

Bibliography

[1] EsPReSSO. https://github.com/RUB-NDS/BurpSSOExtension.

[2] Saml Raider - SAML2 Burp Extension. https://github.com/CompassSecurity/SAMLRaider.

[3] Mercy Viola Akuleut. *Security Test Plan for SAML SSO 2.0 Implementations: The SPID Use Case.* Master's thesis, DISI, Università di Trento.

[4] Matteo Bitussi. *New plugin for Automatic Pentesting.* DISI, Università di Trento, 2021.

[5] Catalin Cimpanu. Major vulnerability patched in the EU's eIDAS authentication system. https://www.zdnet.com/article/major-vulnerability-patched-in-the-eus-eidas-authentication-system/, 2019.

[6] Agenda Digitale. Sicurezza PA, tempo di responsible disclosure e bounty program anche in Italia. https://www.agendadigitale.eu/sicurezza/sicurezza-pa-tempo-di-responsible-disclosure-e-bounty-program-anche-in-italia/, 2019.

[7] Stefano Facchini. *Design and Implementation of an automated Tool for checking SAML SSO Vulnerabilities and SPID compliance.* 2020. Bachelor's thesis, DISI, Università di Trento.

[8] Gianluca Varisco Giovanni Bajo. Perchè la sicurezza informatica non è una questione di bianco e nero. https://medium.com/team-per-la-trasformazione-digitale/sicurezza-informatica-policy-responsible-disclosure-hacker-etici-52a174d44c49, 2016.

[9] Russell Jones. How SAML 2.0 Authentication Works? https://goteleport.com/blog/how-saml-authentication-works/, 2019.

[10] James Chappell Krassimir Tzvetanov, Hendrik Adrian. Introduction to CTI as a General topic. https://www.first.org/global/sigs/cti/curriculum/cti-introduction.

[11] miniOrange. Single Sign-On (SSO). https://www.miniorange.com/single-sign-on-sso.

[12] Palo Alto Networks. Palo alto networks security advisories. https://security.paloaltonetworks.com.

[13] Tomasz Andrzej Nidecki. Session Hijacking and Other Session Attacks. https://www.acunetix.com/blog/web-security-zone/session-hijacking/.

[14] Tomasz Andrzej Nidecki. What is Session Fixation. https://www.acunetix.com/blog/web-security-zone/what-is-session-fixation/, 2019.

[15] OWASP. Cross Site Scripting (XSS). https://owasp.org/www-community/attacks/xss/.

[16] OWASP. Denial of Service. https://owasp.org/www-community/attacks/Denial_of_Service.

[17] OWASP. Session Fixation. https://owasp.org/www-community/attacks/Session_fixation.

[18] Agenzia per l'Italia Digitale. The eIDAS regulation. https://www.agid.gov.it/index.php/en/platforms/eidas.

[19] Agenzia per l'Italia Digitale. SPID - Public Digital Identity System. https://www.agid.gov.it/index.php/en/platforms/spid.

[20] Agenzia per l'Italia Digitale. What is SPID. https://www.spid.gov.it/en/what-is-spid/.

[21] A. Bisegna R. Carbone G. Pellizzari S. Ranise. *Micro-Id-Gym: a Flexible Tool for Pentesting Identity Management Protocols in the Wild and in the Laboratory.*

[22] Marc Rogers. Palo Alto Networks SAML Vulnerability. https://sec.okta.com/articles/2020/06/palo-alto-networks-saml-vulnerability, 2020.

[23] N. Engelbertz N. Erinola D. Herring J. Somorovsky V. Mladenov J. Schwenk. *Security Analysis of eIDAS. The Cross-Country Authentication Scheme in Europe.* 2018.

[24] Lie Solutions. Digitalizzazione: pro e contro di un fenomeno inarrestabile. https://www.lie-solutions.com/digitalizzazione/.

[25] Lorenzo Tait. *A customized Threat modeling for secure deployment and pentesting of SAML SSO solutions.* 2019. Bachelor's thesis, DISI, Università di Trento.

[26] Kali Tools. Burp Suite Package Description. https://tools.kali.org/web-applications/burpsuite.

[27] VerSprite. What is Responsible Disclosure? https://versprite.com/security-testing/what-is-responsible-disclosure/, 2020.

[28] Jamsheer's Views. Difference between IDP initiated SSO and SP initiated SSO. http://jamsheert.blogspot.com/2015/08/difference-between-idp-initiated-sso.html, 2015.

[29] Rizki Wicaksono. XML Encryption Attack. http://www.ilmuhacking.com/cryptography/xml-encryption-attack/.

Appendix A List of passive tests

Table A.1: SAML passive tests.

Name	Description	Refs	Msg	Tool(s)	New
Check the presence of the `ID` attribute	This attribute is need to uniquely identify the authentication flow between the SP and the IdP.	[25, 7]	Req.	MIG	-
Check the presence of the `ProviderName` attribute	This attribute specifies a friendly name for the SP, it should identify the same entity of the `Issuer` element.	[25, 7]	Req.	MIG	-
Check the presence of the `Issuer` attribute	This node must be child of the `AuthnRequest` element.	[25, 7]	Req.	MIG	-
Check the presence of the `Signature` element	The `Signature` is searched in the appropriate place depending on the HTTP-Redirect and HTTP-POST bindings.	[25, 7]	Req.	MIG	-
Check the SPID compliance	Checks if the Request and the Response are compliant to the SPID format.	[7]	Req., Resp.	MIG	-
Check the presence of the Service Provider references	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.	[25, 7]	Resp.	MIG	-
Check the presence of the `Recipient` element in `Assertion`	Checks the presence of the `Recipient` attribute of the `SubjectConfirmati- onData` element, which shall be located at the path `AuthnResponse/ Assertion/Subject/ SubjectConfirmation`.	[25, 7]	Resp.	MIG	-
Check the presence of the `Audience` element in `Assertion`	The `Audience` element is the child of the `AudienceRestriction` element, that must be found at the path `AuthnResponse/Assertion/Condition`.	[25, 7]	Resp.	MIG	-
Check the presence of the `Destination` element in `Assertion`	The implementation of a verification mechanism must be implemented in order to verify that the value of the `Destination` element represents the location at which the Response was sent and if that condition does not hold, the Response must be discarded.	[25]	Resp.	MIG	-
Check the `InResponseTo` element	Checks if the `InResponseTo` attribute of the Response matches the `ID` attribute of the Request.	[25, 7]	Resp.	MIG	-

Check the Canonicalization form	Checks if the `Canonicalization Algorithm` is the one with comments.	[7]	Resp. MIG	-
Check the presence of the `OneTimeUse` element	It must be found as attribute of the element `Condition`, which is a child of the root element `AuthnResponse`.	[25, 7]	Resp. MIG	-
Check the presence of the `NotOnOrAfter` attribute	Checks whether the attribute is present in the `Condition` element and in the `SubjectConfirmation Data` element, if the two attributes match and if they are set to a recommended value.	[25, 7]	Resp. MIG	-

Appendix B List of active tests

Table B.1: SAML active tests.

Name	Description	Refs	Msg	Tool(s)	New
Integrity protection of `RelayState`	Checks whether the IdP implements sanitization mechanism on the value of the `RelayState` parameter. This test tampers the parameter so the user is going to be redirected to a potentially harmful page and checks if the original value is kept protected, generating an error if it is modified.	[7]	Req.	MIG	-
Session protection	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.	[7]	Req.	MIG	-
Certificate Faking	Replace the `<Signature>` element of an XML message with a self-generated signature and key. The test make the access to arbitrary accounts feasible, since it is possible to generate and sign valid tokens containing the identities of other users.	[23]	Req., Resp.	SAML Raider	X
XML Signature Exclusion Attack	Consists in intercepting a SAML message, removing all the signatures. It can tamper the security elements or attributes.	[25]	Req., Resp.	SAML Raider	-

XML Signature Wrapping Attack #1 - #8	The attacker obtains a SAML assertion with a valid Signature. An original element of the SAML message is moved within the document to a position unknown to the application logic. This location allows the element to not be processed by application logic, while the signature verification logic can still verify the element. There can be different permutations of the attack based on the location of the original element, the faked element and the signature. It is possible to perform 8 different variants of the attack exploiting several elements and positions of the SAML message.	[25]	Resp.	SAML Raider	-
Authentication Bypass attack	When SAML authentication is enabled and the *Validate Identity Provider Certificate* option is disabled (unchecked), improper verification of signatures in SAML authentication enables an unauthenticated attacker to access protected resources.	[12]	Resp.	SAML Raider	X
Exponential entity expansion using parameter entities	In this attack an external entity is declared and a reference is made to a maliciously large file on a network resource or locally.	[25]	Req., Resp.	EsPReSSO	-
DOS via `Xinclude`	This attack uses an element containing the reference namespace for `XInclude`, an element with that namespace and an href attribute pointing to a malicious large local file.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Classic XXE variant 1	It leverages external entity references in order to embed the content of different file into the XML document.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Classic XXE attack using netdoc	It uses Netdoc as a protocol to embed content from different file into the XML document. Netdoc is a URI scheme and it's based on Oracle's Java virtual machine.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Classic XXE using UTF-16	The attack variant is based on changing the XML encoding from UTF-8 to UTF-16 in order to embed data which are located outside the main file into the XML document.	[25]	Req., Resp.	EsPReSSO	-

Sensitive File Disclosure via Classic XXE using UTF-7	The attack variant is based on changing the XML encoding from UTF-8 to UTF-7 in order to embed data which are located outside the main file into the XML document.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - Bypassing XXE restriction variant 1	It utilizes parameter entities to wrap content in a `CDATA` escape so the wrapped external entity includes the content of the target document. Another external entity points to an external `.dtd` file which contains only references to the parameters entities wrapping the content.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - Bypassing XXE restriction variant 2	In this attack the external entity and the contained parameter entities are defined in a `payload.dtd` file hosted on the attacker's server. An entity reference of the main XML file allows the inclusion of external document's content through those external entities.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - abusing attribute values	It consists in having an element `root` that contains an attribute whose value is a reference to a general entity which is declared by the value of a parameter entity on another `.dtd` file. The value of the general entity is a parameter entity reference to another parameter entity whose value is the target file.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - Out of Band variant 1	The DTD file forces the XML parser to echo the content of a target file and to assign it to an entity. Then it will create a reference to that entity containing the content of the target file.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - Out of Band variant 2	The attack is executed by referencing the hosted DTD file and creating a connection to load that external DTD file. The hosted DTD file then uses parameter entities to wrap the contents of the `/etc/passwd` file into another HTTP request to `tester.com`. It is now possible to extract the contents of `/etc/passwd` file by reading incoming traffic on `tester.com`.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - Out of Band variant 3	This attack variant utilizes a ftp request in order to perform the same attack as the variant 2.	[25]	Req., Resp.	EsPReSSO	-

Sensitive File Disclosure via Advanced XXE attack - File-not-found exception variant	In this attack the data extraction is performed by forcing an error on the server through the addition of an invalid URI to the request.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - SchemaEntity via `schemaLocation`	It consists in a declaration of an external parameter entity `remote` with a URL resource as literal entity value, a reference of the external parameter entity `remote` and a declaration of an element `data` with a namespace xsi for XML-Schema and the attribute `xsi:schemaLocation`. The value of `xsi:schemaLoation` points to a server and the URL contains a reference to an internal general entity. The parser invokes a URL request and the contents of the file are transmitted to the attacker.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - SchemaEntity via `noNamespace SchemaLocation`	The attack differs from the previous one in the declaration of the element data which contains the attribute `xsi:noNamespace SchemaLocation`.	[25]	Req., Resp.	EsPReSSO	-
Sensitive File Disclosure via Advanced XXE attack - SchemaEntity via `XInclude`	The attack differs from the previous one in the declaration of the element data which contains the attribute `xsi:XInclude`.	[25]	Req., Resp.	EsPReSSO	-
Server Side Request Forgery (SSRF) via XXE - External general entities variant	This attack is performed by creating inside a `DOCTYPE` an external entity whose value is a path to a target file and with a reference to that external entity in order to include its content in the XML document.	[25]	Req., Resp.	EsPReSSO	-
Server Side Request Forgery (SSRF) via XXE - External parameter entities variant	Given that the parameter entities are not subjected to constraints of well-formedness, they are used to enable SSRF attacks by creating requests to internal network from web-server parsing XML document in such a way to bypass the perimeter protection.	[25]	Req., Resp.	EsPReSSO	-
Server Side Request Forgery (SSRF) via XXE - `schemaLocation` variant	the `schemaLocation` enables SSRF attacks by creating requests to internal network from web-server parsing XML document using the namespace SchemaLocation in such a way to bypass the perimeter protection.	[25]	Req., Resp.	EsPReSSO	-

Server Side Request Forgery (SSRF) via XXE - `noNamespaceSchemaLocation` variant	The `noNamespaceSchema Location` attribute enables SSRF attacks by creating requests to internal network from web-server parsing XML document in such a way to bypass the perimeter protection.	[25]	Req., Resp.	EsPReSSO	-
Server Side Request Forgery (SSRF) via XXE - XInclude variant	the attack consists in inserting an internal URL related to a file as a value of the attribute href in the `xi:include` element inside of a related element.	[25]	Req., Resp.	EsPReSSO	-
TCP scans	The attack consists in inserting an internal URL related to a file as a value of the attribute href in the `xi:include` element.	[25]	Req., Resp.	EsPReSSO	-
Local file read/write	The XSLT transformation (Extensible Stylesheet Language Transformation) is executed before verification of the digital signature because it is allowed by the XML Signature standard. The attack consists in preparing a SAML token `t` and creating an XML Signature for it which is not necessary to be correctly computed but inserted in a well-formed XML document. Then a Transform element is added to the XML Signature and a XSLT Payload is placed in it. This payload has the path of the targeted file as an attribute of the element `<xsl:value-of>`.	[25]	Req., Resp.	EsPReSSO	-
Exponential entity expansion attack (Billion Laugh)	A lot of nested entities are defined within an XML `DOCTYPE` declaration, which can be easily exploited to construct a memory bomb.	[25]	Req., Resp.	EsPReSSO	-
Quadratic blowup attack	The attack consists in defining one single large entity and to refer to it many times. Basically, an entity "&a;" with a significant length of characters, say 100,000 long, refers to "&a;" 100,000 times inside the root element.	[25]	Req., Resp.	EsPReSSO	-

XML Encryption Attack	This attack consists in modifying the SAML message with a different from the original encryption key. After the XEA is performed, the assertion related to data is fully encrypted with the new key. Successful attacks against XML Encryption would undermine the confidentiality goals of the encrypted SAML assertions.	[23]	Req., Resp.	EsPReSSO	X
Malicious resource provision by Intruder SP - Delivery of unrequested resource variant	The client initiates the protocol by requesting a bad resource at a malicious SP. the malicious SP, pretending to be the client, requests a different resource URI at another SP. SP generates an Authentication Request which is returned to the malicious SP. It replies to the client by sending a response to IdP containing an Authentication Request generated by SP, instead of the Request with the fields related to original request of the user. The IdP authenticates the user though an authentication challenge. The attack makes the client consume a resource from SP, while the client originally asked for a different resource at the malicious. It is performed by modifying the `RelayState` element in order to let the user consume a malicious resource.	[25]	Req., Resp.	Burp Suite	-
Malicious resource provision by Intruder SP - CSRF attack variant	Similar to the previous one, differs in making the client consume a URL-encoded command that performs an undesired action. It is performed by modifying the `RelayState` element in order to let the user consume the malicious action.	[25]	Req., Resp.	Burp Suite	-
Malicious resource provision by Intruder SP - Login CSRF attack variant	The attacker authenticates himself with his credentials through the SAML authentication phase for a specific resource. Then, he/she makes the victim send a HTTP request containing its credentials to the authentication end point of a web site. Receiving the forged request, the website will authenticate the victim as the attacker.	[25]	Req., Resp.	Burp Suite	-

Malicious resource provision by Intruder SP - XSS attack variant	Executing the same procedure as the previous attacks, at the end it makes the client execute malicious code as XXS attack. It is performed by modifying the `RelayState` element by inserting a malicious XSS code.	[25]	Req., Resp.	Burp Suite	-
Malicious resource provision by Intruder SP - XSS for User Impersonation variant	The procedure is the same of the previous one, but the attack makes the client consume the URI `javascript:window.open(’uri(i)’+document.cookie)` that will make the client victim of theft of its cookies. It is performed by modifying the `RelayState` element with the given URI.	[25]	Req., Resp.	Burp Suite	-
Canonicalization Attack	This attack consists in inserting comments into the value of `NameID` element of the attacker’s assertion so that the XML document still has the same signature but the victim is going to be authenticated.	[25]	Req., Resp.	Burp Suite	-
Expired SAMLResponse Replay Attack	As a registered user, the attacker sends a valid Response to the SP. Once the attacker has received the requested resource, it sends the Response or the Assertion again within the valid lifetime to the same SP. The attack is performed successfully whenever the SP provides the requested resource for the second time.	[25]	Req., Resp.	Burp Suite	-
Level of Assurance downgrade attack	This attack consists in intercepting the SAML Response tampering the Level of Assurance present in the `AuthnContextClassRef` element contained as a sub-element of `AuthContext` of the `AuthStatement`.	[25]	Req., Resp.	Burp Suite	-