Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer Science

FINAL DISSERTATION

Enlarging the Pen-Test Coverage of SAML Single Sign-On Solutions with Cyber Threat Intelligence

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.

I would also like to thank Prof. Ranise for the
internship opportunity at Fondazione Bruno Kessler,
his kindness and encouragement.

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. Digitalisation has positively affected businesses as well as public administrations and has enabled easier access to services for both private and public administrations.

Digital identities, the set of information that allows to uniquely identify an individual, are the key factors to approach the process of digitalisation. One of the mechanisms that assist users in identity management is Single Sign-On (SSO), an authentication service that provides an easy way to access different web applications with just a set of credentials. However, this way of accessing information has to deal with a continuous increase of vulnerabilities: these facts prove the necessity to increase the level of security in the implementation of these services. This can be achieved by leveraging pentesting tools capable to spot security issues in the implemented services.

There are numerous available pentesting tools that assess the security of SSO services but they cover only some types of vulnerabilities, leaving others uncovered. This leads to an incomplete awareness of the current risks with which users have to deal.

This thesis is focusing on the Security Assertion Markup Language Web Profile 2.0 SSO (SAML SSO): a standard for an authentication protocol providing a SSO experience for web applications. Vulnerabilities in the implementations of this protocol are constantly disclosed, even though it has been known for years. The first contribution of this thesis is the analysis of the state-of-the-art vulnerabilities regarding SAML SSO implementations through a Cyber Threat Intelligence phase starting from previous colleagues' works, with an in-depth analysis of scientific reports, to find the latest novelties in the field.

We considered the state-of-the-art automatic pentesting tools for SAML SSO in order to assess their coverage, understanding which kind of vulnerabilities they can detect choosing between the results of the Cyber Threat Intelligence phase. They concern only target specific vulnerabilities, but also their accuracy is challenged by the number of false positives they are prone to. We also managed to perform an experimental analysis on Sistema Pubblico Identità Digitale (SPID) scenarios, based on SAML SSO, to assess the security of these implementations and the effectiveness of the existing tools, resulting in important vulnerabilities, but also false positives.

Therefore, in the final part of the thesis, we provide a JSON test suite by specifying all the tests as an input (JSON file) of a novel cutting-edge pentesting tool. Then, we managed to check the correctness of the existing tools with a comparison analysis with the new one. We used as systems under tests Micro-Id-Gym backends, EsPReSSO and SAML Raider. The results consist of an increment in terms of coverage and effectiveness.

The solution provides an easy way to perform an analysis on the known tests in an automatic way, not only for cyber-security experts but also for beginners.

Acronyms

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

Key words

SAML SSO, Cyber Threat Intelligence, Pentesting Tools.

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 performance, with the possibility of guaranteeing products and services of great quality [18]. Also in public administration, digitalisation aims to improve, thanks to digital identities, 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. Every day, public administrations, but also privates, manage sensitive information, in particular credentials which are the key factor to protect private data.

However, this way to relate to users 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, therefore it is increasingly crucial for users and organisations to have a secure system for digital identities.

In this thesis, we are focusing on the Security Assertion Markup Language Web Profile 2.0 SSO protocol (SAML SSO) that supports Single Sign-On (SSO). SSO allows users to be authenticated for multiple applications at once 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, but there is, therefore, the need to bring the security of SAML SSO implementations to a higher level to avoid potential damage that vulnerabilities could cause, which is the aim of our work.

SAML SSO is used on Sistema Pubblico Identità Digitale (SPID) [5] provided by the Italian government, and on the electronic IDentification Authentication and Signature (eIDAS) [4] as a European regulation. 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 recognised to make personalised and secure use of digital services [6]. 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" [4].

1.2 Problem

Thanks to SSO, a user can perform the authentication process once to the Identity Provider (IdP), a system that provides authentication services to relying applications within a federation or distributed network. If this process is successful, the user is granted to access all related Service Providers (SP), an organisation that provides services to the public, in a transparent way [14]. The case we analyse is based on SAML SSO thus we consider the communication between the SP and the IdP made of SAML messages. Misconfigurations in SAML SSO implementations enable malicious users to exploit authentication flaws to perform several attacks such as Denial of Service [22] and Cross Site Scripting [21], just to mention a few. All this leads to the need of having tools to support developers.

There already are some existing pentesting tools that work in this field, but their work focuses only on specific target vulnerabilities. Also, the amount of vulnerabilities is constantly growing and those tools have a critical number of untested flaws and provide false positive results, so 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 test suite, together with a new tool, should be created as a strategy to ensure that SAML SSO implementations are tested to spot unfolding security threats and vulnerabilities.

In order to address this problem, we proceeded to improve the threat modelling for SAML SSO

through careful research into new vulnerabilities and attacks. We also provide an experimental analysis on real world scenarios for an examination of the existing tools, which brought false positive results causing a bad estimation of the security of the use cases.

The issues that emerged from the incorrectness of analysed pentesting tools enlighten the absolute necessity of having a unique plugin capable of identifying SAML SSO vulnerabilities in a semi-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 an easier manner.

1.3 Contributions

The contributions of this thesis are the following:

- *Cyber Threat Intelligence*: One contribution of this thesis is the development of a SAML SSO test suite to facilitate automatic pentesting w.r.t. SAML SSO. The first phase consisted in collecting vulnerabilities and attacks in order to test automatically if a SP or IdP is vulnerable to the given attacks. The sources used for collecting information are scientific reports about eIDAS vulnerabilities and attacks [13], and some colleagues' Bachelor and Master's theses about SAML SSO security assessment [28, 7], and plugins [14]. 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.

- *Tools for Automatic Pentesting of SAML SSO*: The Cyber Threat Intelligence phase allowed us to map the found attacks with existing tools. We focused, due to its diffusion, on Burp Suite's [17] 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 [9, 14]. Their selection allowed us to understand their capabilities of analysing vulnerabilities and attacks.

- *Specification of JSON Test Suite for a Comprehensive Tool*: The most important phase of this thesis was the creation of a JSON test suite by specifying all the tests as an input (JSON file) of a novel pentesting tool. Thanks to the development of a new plugin [10], we managed to convert almost all the collected vulnerabilities and attacks, increasing the coverage of the feasible tests. This new plugin covers the tests supported by the existing pentesting tools previously reported. The results in terms of effectiveness correctly reflected our expectations, thus allowing us to facilitate automatic pentesting and to provide a support tool which allows developers to test their implementations.

- *Experimental Analysis on SAML SSO*: By mapping these series of tests with the corresponding tools, we performed an experimental analysis on real world use cases based on SPID scenarios in order to prove the correctness of the existing tools: the results spotted some false positives which were fixed in the new plugin. Also, serious vulnerabilities were detected and, in order to communicate the found security flaws, confidential security reports were drafted for an appropriate responsible disclosure [30].

1.4 Structure of the thesis

The thesis is divided in the following way:

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

- *Chapter 3 - Cyber Threat Intelligence*: Details of analysis on the latest news related to vulnerabilities and attacks and description of the found novelties.

- *Chapter 4 - Tools for Automatic Pentesting of SAML SSO*: Mapping between attacks found in the Cyber Threat Intelligence phase and existing pentesting tools that allow us to check the integrity and possible flaws of a SAML SSO implementation.
- *Chapter 5 - Experimental Analysis and New Plugin*: Experimental analysis in real world use cases based on SPID scenarios, comparison between the existing tools, found flaws and vulnerabilities, responsible disclosure and implementation of a JSON test suite to facilitate automatic pentesting.
- *Chapter 6 - Conclusions and Future Work*: 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 SSO, SAML SSO, and the most relevant use cases.

2.1 Single Sign-On

SSO is an authentication schema that allows a user to log in with a single set of credentials to several services. In Figure 2.1 we report the authentication schema of SSO which 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 successful authentication, elements from at least two factors should be verified. The three factors are:

- *Knowledge factors*: something the user knows, like a password or personal identification number.
- *Ownership factors*: something the user possesses, such as an 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.

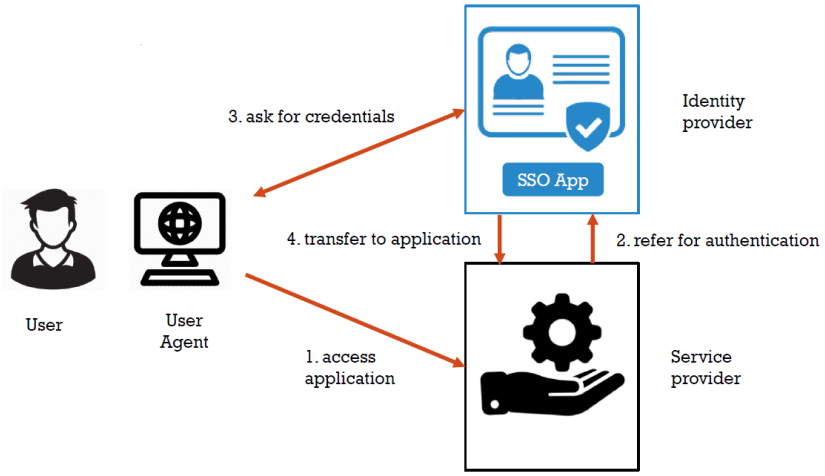

Figure 2.1: SSO authentication schema [26]

2.2 SAML SSO

"SAML is one of the most widely used open standard for authentication and authorisation between multiple parties" [16]. SAML SSO is one of the protocols that give users the SSO experience to access services. Without SSO on the user side there is the necessity to remember several logins/passwords or, even worse, using the same ones.

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

- *Principal/User Agent*: 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.

SAML SSO can be started either SP-side and IdP-side. In this thesis we are focusing only on the SP-side, reported in Figure 2.2, in which the user goes to the SP (for example on *sp.example.com*), and if the user has no active session, the SP redirects the user to the IdP which will ask the user to connect. After entering the user's correct login credentials, the user will be redirected to the SP with a valid session.

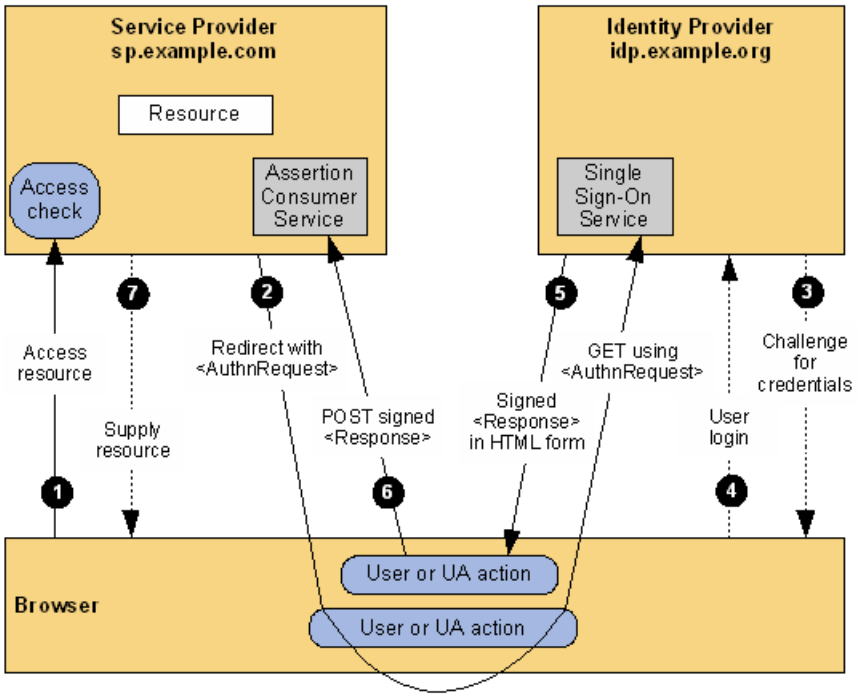

Figure 2.2: SAML SSO SP-side [15]

The exchanged messages in a SAML SSO scenario are SAML Requests and SAML Responses which might contain zero or more SAML Assertions, namely packages of information. The SAML Requests are the authentication requests, generated by the SP with the purpose of requesting authentication for the user. The SAML Responses are generated by the IdP containing the actual Assertions to asset the authentication statement of the user.

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

SPID is a digital identity consisting of a set of personal credentials, with which it is possible to access online services of the private and public administration. In relation to the SAML SSO profile, SPID implements the SP-initiated SSO: in the authentication process, users authenticate to SPs, which are public or private organisations providing a service to authorised SPID users [5]. Thanks to it, it is possible to access various services using the same credentials, which must be requested from one of the many available IdPs (or certifying bodies). The system was born to facilitate the dissemination and use of online services, it is free for all citizens and is managed by the Agency for Digital Italy (AgID) [12]. It "ensures to citizens and enterprises to be uniquely recognised by identifiers issued by certified IdPs" [25].

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 [4]. 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 organisations 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 SSO based compatibility layer between different eID implementations. The components facilitating the cross-border information exchange are called eIDAS-Nodes [13]. As SAML SSO is utilised as a standard on the nodes, eIDAS was considered in this work because the purpose of this thesis is also applicable to a European reality.

3 Cyber Threat Intelligence for SAML SSO

Cyber Threat Intelligence consists of 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 rising threats. It is designed to inform all levels of decision makers [29]. 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 number of known threats.

3.1 Sources

The goal of this analysis phase is to offer a list of threats starting from a colleague's previous work [28]. 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 [13].
- Bachelor's thesis based on the implementation of an automated tool for checking SAML SSO vulnerabilities and SPID compliance [14].
- Master's thesis regarding a security test plan for SAML SSO mainly focused on the SPID use case [7].
- Web searches 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 [24, 11, 19].

3.2 Vulnerabilities and Attacks

An 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 [28, 14, 7], 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 [31]. *XML Encryption Attack (XEA)* consists in modifying HTTP messages involved in a SAML SSO scenario 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 [13].

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 SSO service is vulnerable to Certificate Faking, the authentication scheme is broken because a malicious user is able to forge arbitrary messages and identities [13].

3.2.3 Authentication Bypass Attack

Authentication Bypass Attack refers to a critical security vulnerability affecting SAML SSO 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 impacts customers using Palo Alto devices with IdPs that rely on the SAML SSO 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 [27].

3.2.4 Session Fixation Attack

Session Fixation Attack is the possibility for an attacker to hijack a valid user session. It takes advantage of the limitations with which ID sessions are handled. When a user is authenticated, a new session ID is not assigned, making it possible to use an existing session ID. The attack consists of getting a valid session ID, inducing a user to authenticate with the same session ID and then hijacking the user-validated session by the knowledge of the used session ID [23]. Therefore consists in fixing an established session on the victim's browser.

The related vulnerability consists in insufficient web application security and bad coding practices associated with session management, in particular with 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 [20].

3.2.5 Clickjacking

Clickjacking (or *User-Interface Redressing*) consists in tricking the user into unknowingly executing 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 becoming compromised. It can be used to lure the victim into unknowingly authenticating at the IdP and authorising access to sensitive resources. This vulnerability can be mitigated by adding a `X-Frame-Options` header to relevant pages.

4 Tools for Automatic Pentesting of SAML SSO

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 we aimed to increase the coverage of tested vulnerabilities and attacks, we consider a few SAML security tools intending to understand which kind of vulnerability they test. In this phase we considered Burp Suite (Burp) [17] 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 [17].
- *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 production 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 (respectively referred to vulnerabilities and attacks) on SAML Request and SAML Response to automatically detect security issues [9, 14].
- *EsPReSSO*: A Burp plugin that processes and recognises SSO protocols. It is incorporated with WS-Attacker, DTD-Attacker and XML-Encryption-Attacker integration while intercepting SAML messages [1].
- *SAML Raider*: A Burp plugin for testing SAML SSO 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 [9], 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 disable the acceptance of unsolicited SAML messages; *(ii)* Missing protections against CSRF 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.

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

All the passive tests were performed with the Micro-Id-Gym tool [9, 14]. They are intended to control the presence of elements within the SAML Request or SAML Response. The consequences of 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 –related to SAML Request and SAML Response– and tool with whom the tests can be performed.

In order to better understand these tests and the correlated risks due to their exploits, some of the most relevant examples, together with a brief description, are reported [28].

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.

Session Protection. The Session attack is accomplished by replaying the intercepted SAML Response without the authentication cookie previously settled between SP and IdP. The authentication cookie is needed to avoid CSRF that can result in user impersonation [14].

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 of inserting a comment inside the value of the `NameID` element of the assertion in order to impersonate the victim. The usage of an unsafe canonicalization

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

algorithm can cause an user impersonation attack since, after the edit, the XML document has still the same signature.

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

5 Experimental Analysis and New Plugin

We decided to consider all the available tests, reported in Chapter 4, in order to perform an experimental analysis in real world use cases. The purpose of this phase is to test the security provided by the chosen SPs and IdPs, but also to compare the existing pentesting tools' efficiency.

In this chapter, the results of the experimental analysis on SPID scenarios are reported. We also specify a JSON test suite containing the SAML SSO tests to be used inside a new tool, a Burp plugin developed by a colleague [10]. The aim is to unify vulnerabilities and attacks under a comprehensive tool which provide a facilitated 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 cataloguing the attacks and their vulnerabilities is to run and test them in some selected SPs and IdPs in order to compare the results obtained by the different tools. In this section, a description of the experimental analysis phase's results is reported. In particular, we will describe the found flaws: in Subsection 5.1.1 the false positives, their exploitation and the comparison between existing tools; in Subsection 5.1.2 the found vulnerabilities and the adopted solutions for the vulnerable situations. The testing phase is based on 6 scenarios, each including 85 tests, for a total amount of 510 performed tests.

5.1.1 False positives

The results of the experimental analysis reported false positives which were all correlated to the Micro-Id-Gym tool, in particular:

- *Canonicalization form*: every test results vulnerable to a canonicalization attack because of the usage of an unsafe canonicalization algorithm. However, a more accurate manual analysis with Burp 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 (Table 4.2, Line 2) 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 [14].
- *Signature*: indications contained in the SAML SSO core document require 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 active test (Table 4.2, Line 4) was performed and the scenario results not vulnerable.

5.1.2 Discovered vulnerabilities

The experimental analysis has led to important results. All the tests have been performed several times with the relative tools and manually with Burp to ensure the results were correct. In this paragraph, particular attention is given to the description of found vulnerabilities.

Certificate Faking attack resulting in Impersonation attack This attack, performed with SAML Raider, 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 as reported in 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.

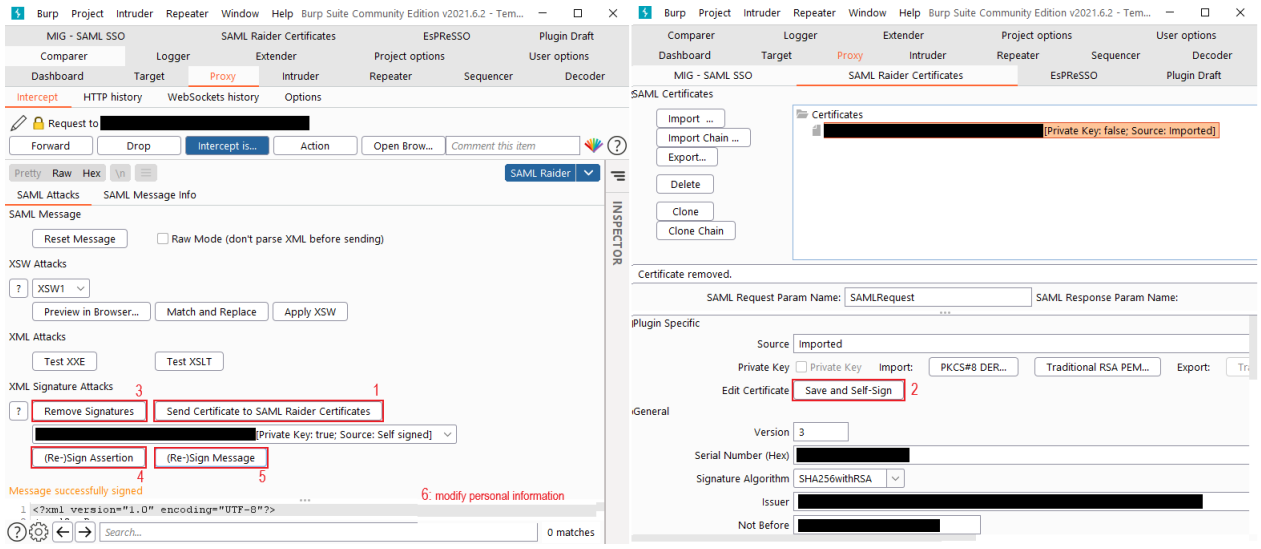

Figure 5.1: Certificate Faking with SAML Raider

Cross Site Request Forgery This attack consists in intercepting and copying the SAML Response, dropping the request and closing the browser. In a new session we have to create another request and replace it with the replayed message previously intercepted. So the tester is logged into his/her account in a new session, unrelated to the original. In Figure 5.2 we report the replayed SAML Response. 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 issues have been fixed

The number of issues reported in Subsections 5.1.1 and 5.1.2 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 arisen problems are specified: the important task of responsible disclosure on found vulnerabilities and the 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

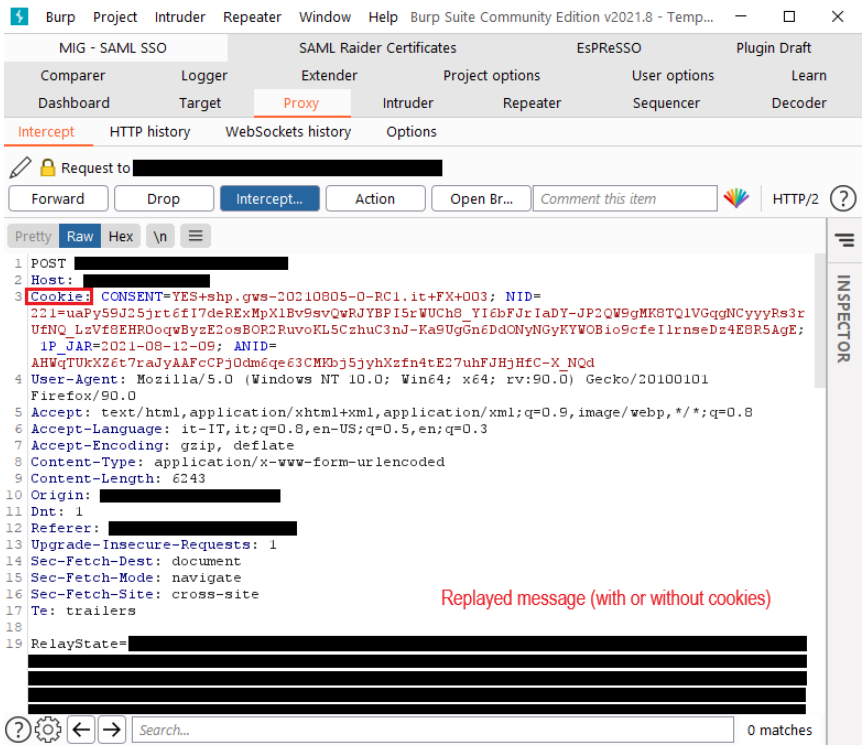

Figure 5.2: Replayed SAML Response for Cross Site Request Forgery with Burp Suite

then provides the vendor with an opportunity to mitigate the vulnerability before disclosing its existence to the general public [30].

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 description of the test reproduction and the risk of a possible attack were provided. For safety reasons, 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 [3].

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 minimising 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 cataloguing 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 regularising a national responsible disclosure policy.

5.2.2 Created new JSON Test Suite and Plugin

Despite the false positives reported in Subsection 5.1.1, the 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

test suite based on existing tools, to facilitate automatic pentesting even for those who are novices with the disclosure of vulnerabilities.

In order to achieve this goal, a new Burp plugin has been developed by a colleague [10]. This tool consists of a plugin able to recognise SAML messages, properly decode the assertion depending on the used binding, perform penetration testing based on a JSON test suite and presents the results in a tabular way. The user interface is depicted in Figure 5.3 and is visually divided into two sections:

- The upper one represents what pentesters use in order to perform the authentication flow, namely the authentication trace –a list of user actions to execute the authentication flow in the system under test–, the Selenium Web Driver file[1] and the selected browser.
- At the bottom we can find the area where the test suite is described: once tests are performed, the results are shown in the *Test Suite Result* tab.

The plugin allows performing passive tests in the order of a few seconds and all active tests within an hour. The time needed to perform all the tests with the new tool is, therefore, significantly reduced compared to the use of the existing tools.

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

In order to correctly execute the tests, we generated the authentication trace of every tested SP or IdP for whom secrecy must be maintained for a proper responsible disclosure as reported in Subsection 5.2.1. In addition, a suitable driver had to be selected which led to the track execution. The next steps consisted in importing and reading the JSON test suite with a consequent execution of the test suite.

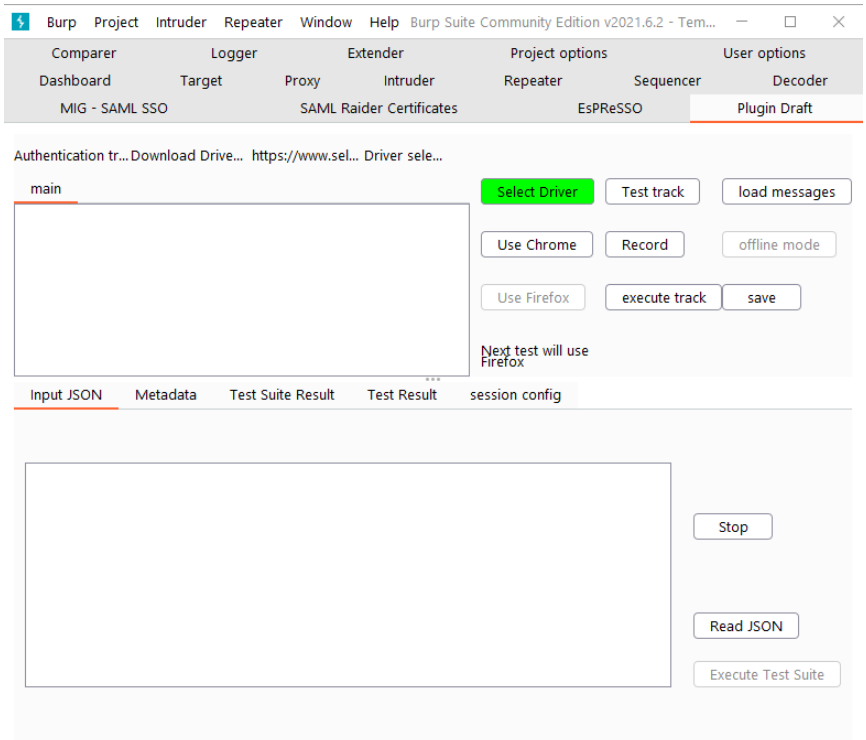

Figure 5.3: User interface of the new plugin [10]

Practical examples of JSON test suite. Each test suite shares a preamble, as shown in Listing 5.1, including `name` and `description`, followed by the list of tests included in the test suite.

[1] `https://www.selenium.dev/downloads/`

Listing 5.1: “Preamble of the test suite”

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

An example of a JSON 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: “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: “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 the European level.

Considering the performed Cyber Threat Intelligence phase w.r.t. SAML SSO, we improved the number of known threats and mapped vulnerabilities and attacks with existing pentesting tools. Moreover, we performed an experimental analysis in real world SPID scenarios which led to the discovery of issues related to false positives, but also to security flaws in the analysed SPs and IdPs. Given the fact that the existing tools cover only target specific vulnerabilities, we considered the need to unify vulnerabilities and attacks' exploitation under an innovative tool capable of performing passive and active tests on the intercepted SAML messages. Therefore we managed to create a JSON test suite capable of gathering all the mentioned tests. Once a test has finished, the test result is displayed, highlighting the element to which is related to and the possible flaw.

The developed tool can be used by cyber-security experts that want to test SPs and IdPs on a SAML SSO implementation, but also by pentesters who are interested in investigating and analysing a scenario.

The goal of this thesis was therefore to improve the list of known vulnerabilities and attacks, to build a proper test suite in order to create an automated and easier way for pentesting processes, and to perform an experimental analysis. The limitations encountered in the process of achieving the goal included the impossibility to provide a complete test suite as the developed tool still does not support specific types of tests.

As future work, this innovative tool and test suite can be improved by completing the implementation of the missing active tests and by adding a more complete way of specifying the oracle for each test.

Bibliography

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

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

[3] 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.

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

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

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

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

[8] Giovanni Bajo and Gianluca Varisco. 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] Andrea Bisegna, Roberto Carbone, Giulio Pellizzari, and Silvio Ranise. Micro-Id-Gym: a Flexible Tool for Pentesting Identity Management Protocols in the Wild and in the Laboratory. 3rd International Workshop on Emerging Technologies for Authorization and Authentication (ETAA2020).

[10] Matteo Bitussi. *Declarative Specification of Pentesting Strategies for Browser-based Security Protocols: the Case Studies of SAML and OAuth/OIDC.* Bachelor's thesis, Università di Trento, 2021.

[11] Catalin Cimpanu. Major vulnerability patched in the EU's eIDAS authentication system. Technical report, 2019.

[12] Centro Assistenza di Register.it. Cos'è spid e a cosa serve. https://www.register.it/help/cose-spid-e-a-cosa-serve/.

[13] N. Engelbertz, N. Erinola, D. Herring, J. Somorovsky, V. Mladenov, and J. Schwenk. Security Analysis of eIDAS. The Cross-Country Authentication Scheme in Europe. Chair for Network and Data Security, Horst Gortz Institute for IT-Security, Ruhr University Bochum, 2018.

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

[15] 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.

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

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

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

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

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

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

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

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

[24] Palo Alto Networks. Palo Alto Networks Security Advisories. https://security.paloaltonetworks.com.

[25] Alberto Polzonetti and Alessandro Bettacchi. Cloud Services Implementing Security: a case study. Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, 2017.

[26] Silvio Ranise. Introduction to Computer & Network Security course's slides, 2020.

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

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

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

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

[31] 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.	[28, 14]	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.	[28, 14]	Req.	MIG	-
Check the presence of the `Issuer` attribute	This node must be child of the `AuthnRequest` element.	[28, 14]	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.	[28, 14]	Req.	MIG	-
Check the SPID compliance	Checks if the Request and the Response are compliant to the SPID format.	[14]	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.	[28, 14]	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`.	[28, 14]	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`.	[28, 14]	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.	[28]	Resp.	MIG	-
Check the `InResponseTo` element	Checks if the `InResponseTo` attribute of the Response matches the `ID` attribute of the Request.	[28, 14]	Resp.	MIG	-

Check the Canonicalization form	Checks if the `Canonicalization Algorithm` is the one with comments.	[14]	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`.	[28, 14]	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.	[28, 14]	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.	[14]	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.	[14]	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.	[13]	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.	[28]	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.	[28]	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.	[24]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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`.	[28]	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.	[28]	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.	[28]	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.	[28]	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`.	[28]	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`.	[28]	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.	[28]	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.	[28]	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.	[28]	Req., Resp.	EsPReSSO -

Server Side Request Forgery (SSRF) via XXE - `noNamespaceSchemaLocation` variant	The `noNamespaceSchemaLocation` 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.	[28]	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.	[28]	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.	[28]	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>`.	[28]	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.	[28]	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.	[28]	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.	[13]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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.	[28]	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 subelement of `AuthContext` of the `AuthStatement`.	[28]	Req., Resp.	Burp Suite -