



POLITECNICO DI TORINO

Master's degree in Computer Engineering

Master Degree Thesis

**Design and development of WiFi access  
with eIDAS for cross border  
authentication**

**Supervisors**

prof. Antonio Lioy  
ing. Diana Berbecaru

**Candidate**

Muhammad Ali ANJUM

DECEMBER 2019



*To my family, who helped  
me during this journey*

# Summary

There is a significant increase of devices with wireless LAN (Local Area Network) capabilities from the start of 21<sup>st</sup> century and the use of these devices has become a requirement in every profession. This increases the need for wireless connectivity and at the same time security of wireless LAN has become more important. Which led to many national and international initiatives for secure WLAN connectivity. That includes *eduroam* and *govroam*, that provides roaming services for educational and government sectors respectively. But there are some limitations to these projects, they don't support complex authorisation mechanism and are based on RADIUS servers infrastructure, which needs to be maintained in the entire federated hierarchy.

In this thesis we provide a solution to access WiFi connectivity for citizens of European countries with eIDAS (electronic identification and trust services) infrastructure. eIDAS is EU Regulation (EU)910/2014 on electronic identification and trusted services for cross border electronic transaction, which is adopted by all European countries. It provides mutual recognition of electronic identification between member states by establishing interoperability between existing national eID infrastructures for cross border authentication using their own national eID credentials. We developed and tested two solutions, first one using software based approach and second one using eIDAS code and hardware (wireless) infrastructure deployed in Politecnico di Torino university. In first solution we used Zeroshell, which is a Linux based distribution, specifically designed to provide routers and firewall services. We created a Captive Portal and modified Shibboleth authentication to send and receive messages in eIDAS compatible format to provide authentication. The eIDAS framework for authentication is consists of eIDAS Nodes (specific for each country) and Identity Providers (IdP). In second solution we integrated eIDAS-SP code with Politecnico di Torino wireless infrastructure. Authentication is provided by Wifi-Auth eIDAS-SP application with eIDAS framework. Whereas wireless infrastructure includes Cisco WLC (Wireless LAN Controller), Cisco AP (Access Point) and Fortigate-60D firewall, which is responsible for providing Captive Portal, management of authenticated users and network management. We have tested our solution successfully using Italy-SPID (Public System for Digital Identity), Portugal-Chave Móvel Digital and Spain-DNIe (Documento Nacional de Identidad electrónico).



# Acknowledgements

I would like to thank my supervisor, prof. Antonio Liroy, whose extensive knowledge was invaluable. I would also like to thank my supervisor, ing. Diana Berbecaru, the completion of my thesis would not have been possible without her unparalleled knowledge of the field, continuous guidance and advice, specially during the writing phase.

A special thank goes to my father, who has always supported me. I also want to thank my brother and sisters for their support, which helped me to get through the ups and downs of my research.

My deepest appreciation goes to Cesare Cameroni, who has always been tolerant and supportive throughout the thesis. I would also like to thank Ignazio Pedone and Marco De Benedictis, PhD students inside the TORSEC research group. A general thanks goes to all the members of the TORSEC research group, which were always available to help me when needed.

Finally, I would like to thank the Higher Education Commission of Pakistan and Politecnico di Torino for providing the funding and resources to undertake this research opportunity.

# Contents

<b>List of Figures</b>	10
<b>1 Introduction</b>	12
<b>2 Background</b>	14
2.1 User authentication . . . . .	14
2.2 Identity management . . . . .	15
2.2.1 Digital identity . . . . .	15
2.2.2 Identity Management System . . . . .	15
2.3 SAML . . . . .	16
2.3.1 Introduction . . . . .	16
2.3.2 Overview . . . . .	16
2.3.3 Drivers of SAML adoption . . . . .	17
2.3.4 SAML participants . . . . .	17
2.3.5 Basic concepts . . . . .	18
2.3.6 SAML components . . . . .	18
2.3.7 Privacy in SAML . . . . .	22
2.3.8 Security in SAML . . . . .	23
2.4 Shibboleth . . . . .	23
2.5 eIDAS . . . . .	23
2.5.1 Main problems addressed by eIDAS . . . . .	24
2.5.2 eIDAS specification . . . . .	24
2.5.3 eIDAS cryptography requirements for trust between eIDAS entities . . . . .	25
2.5.4 eIDAS protocol . . . . .	27
2.5.5 Attributes . . . . .	28
2.6 Related work . . . . .	29
2.6.1 eduroam . . . . .	29
2.6.2 govroam . . . . .	33
2.7 Review of possible networking tools for implementing Captive Portal with SAML . . . . .	34
2.7.1 PacketFence . . . . .	34
2.7.2 NoDogSplash . . . . .	35
2.7.3 Zeroshell . . . . .	36

<b>3</b>	<b>Design and implementation of WiFi access with eIDAS through Zeroshell</b>	<b>37</b>
3.1	Introduction . . . . .	37
3.2	Authentication flow . . . . .	38
3.3	Zeroshell setup . . . . .	40
3.3.1	Virtual Box initialisation . . . . .	40
3.3.2	Accessing web interface . . . . .	41
3.3.3	Profile creation . . . . .	41
3.3.4	Network configuration . . . . .	42
3.3.5	Captive Portal . . . . .	43
3.3.6	Shibboleth authentication . . . . .	43
3.3.7	Shibboleth configuration files . . . . .	44
3.4	Configuring Shibboleth authentication with eIDAS . . . . .	44
3.4.1	EntityID of the SP . . . . .	45
3.4.2	ApplicationDefaults element . . . . .	45
3.4.3	MetadataProvider . . . . .	45
3.4.4	Cryptography certificates . . . . .	45
3.4.5	Node-Country selection . . . . .	46
3.4.6	SessionInitiator AuthnRequest element . . . . .	47
3.4.7	Attributes configuration . . . . .	47
3.4.8	White-listing . . . . .	47
3.4.9	Problems encountered . . . . .	48
3.5	Authentication cycle . . . . .	48
<b>4</b>	<b>Design of WiFi access with eIDAS through WiFi-Auth eIDAS-SP and Polito wireless infrastructure</b>	<b>53</b>
4.1	Introduction . . . . .	53
4.2	Authentication process . . . . .	54
4.2.1	Wifi-Auth eIDAS-SP set-up . . . . .	54
4.3	Authentication flow . . . . .	54
4.3.1	Authentication flow Italian scenario . . . . .	55
4.3.2	Authentication flow detail: request part . . . . .	57
4.3.3	Authentication flow detail: response part . . . . .	60
4.4	Wifi-Auth eIDAS-SP implementation . . . . .	63
4.4.1	SSH library . . . . .	63
4.4.2	Guest user properties . . . . .	63
4.4.3	Function createUser . . . . .	64
4.4.4	Function createCMD . . . . .	64
4.4.5	Functions randomString and randomPassword . . . . .	64
4.4.6	Login form . . . . .	64
4.4.7	Cryptography certificates . . . . .	65

4.5	Configuration of network elements . . . . .	66
4.5.1	Fortigate-60D introduction . . . . .	66
4.5.2	Cisco WLC 2504 . . . . .	68
4.6	Script for creating ACL rules . . . . .	69
4.7	Testing authentication cycle using TestCafe . . . . .	70
4.8	Authentication cycle . . . . .	71
<b>5</b>	<b>Installation and configuration of WiFi access with eIDAS-SP and Polito wire-</b>	
	<b>less infrastructure</b> . . . . .	<b>77</b>
5.1	Installation of Wifi-Auth eIDAS-SP application . . . . .	77
5.1.1	Architecture . . . . .	77
5.1.2	Docker and Docker-compose installation . . . . .	78
5.1.3	Source code . . . . .	78
5.2	Fortigate-60D . . . . .	79
5.2.1	Firewall interfaces . . . . .	79
5.2.2	Firewall policy . . . . .	79
5.2.3	Virtual IP . . . . .	80
5.3	Cisco WLC 2504 . . . . .	80
5.3.1	WLAN . . . . .	81
5.3.2	Creating WLAN . . . . .	82
5.3.3	ACL (Access Control List) . . . . .	82
5.3.4	WebAuth SecureWeb . . . . .	84
<b>6</b>	<b>Results</b> . . . . .	<b>85</b>
<b>7</b>	<b>Conclusion</b> . . . . .	<b>88</b>
	<b>Bibliography</b> . . . . .	<b>89</b>
<b>A</b>	<b>SAML message flow</b> . . . . .	<b>92</b>
A.1	SAML message flow example with Zeroshell . . . . .	92
A.1.1	Zeroshell to eIDAS-Connector . . . . .	92
A.1.2	eIDAS-Connector to eIDAS-Service . . . . .	93
A.1.3	eIDAS-Service to IdP-Proxy . . . . .	95
A.1.4	IdP-Proxy to IdP . . . . .	97
A.1.5	IdP to IdP-Proxy . . . . .	98
A.1.6	IdP-Proxy to eIDAS-Service . . . . .	100
A.1.7	eIDAS-Service to eIDAS-Connector . . . . .	102
A.1.8	eIDAS-Connector to Zeroshell . . . . .	103
A.2	SAML message flow example with Wifi-Auth eIDAS-SP . . . . .	105
A.2.1	Wifi-Auth eIDAS-SP to eIDAS-Connector . . . . .	105
A.2.2	eIDAS-Connector to eIDAS-Service . . . . .	107

A.2.3	eIDAS-Service to IdP-Proxy . . . . .	108
A.2.4	IdP-Proxy to IdP . . . . .	110
A.2.5	IdP to IdP-Proxy . . . . .	111
A.2.6	IdP-Proxy to eIDAS-Service . . . . .	113
A.2.7	eIDAS-Service to eIDAS-Connector . . . . .	115
A.2.8	eIDAS-Connector to Wifi-Auth eIDAS-SP . . . . .	117

# List of Figures

1.1	Example of eIDAS cross border authentication . . . . .	12
2.1	Relationship between SAML core components (source: cse.wustl.edu). . . . .	18
2.2	eIDAS basic architecture (source: IEEE) . . . . .	24
2.3	eIDAS architecture for cross border authentication (source: IEEE access [32]) . . . . .	25
2.4	eID Schemes notified in Europe (source: EUROSMART [34]) . . . . .	26
2.5	TLS recommended cipher suites (source: eIDAS [37]) . . . . .	26
2.6	eIDAS authentication Level of Assurance (source: SCRIVE) . . . . .	29
2.7	eduroam confederation structure (source: GEANT). . . . .	31
2.8	Tunnelled authentication (source: Alfa&Ariss). . . . .	31
2.9	eduroam basic authentication flow (source: Belnet). . . . .	32
2.10	PacketFence components (source: PacketFence). . . . .	34
2.11	PacketFence architecture (source: PacketFence). . . . .	35
3.1	Zeroshell Captive Portal infrastructure in test environment . . . . .	37
3.2	Zeroshell authentication flow . . . . .	38
3.3	Zeroshell web interface . . . . .	40
3.4	Network interfaces of Zeroshell . . . . .	41
3.5	Zeroshell command line interface . . . . .	41
3.6	Zeroshell profile creation . . . . .	42
3.7	Zeroshell profile creation parameters . . . . .	42
3.8	Zeroshell network configuration . . . . .	43
3.9	Enabling Captive Portal on Zeroshell . . . . .	43
3.10	Enabling Captive Portal on Zeroshell . . . . .	44
3.11	Configuration of Shibboleth files . . . . .	44
3.12	Captive Portal White-listing for eIDAS-nodes/IDP . . . . .	48
3.13	Zeroshell Captive Portal authentication Step-1 . . . . .	49
3.14	Zeroshell Captive Portal authentication Step-2 . . . . .	49
3.15	Zeroshell Captive Portal authentication Step-3 . . . . .	49
3.16	Zeroshell Captive Portal authentication Step-4 . . . . .	50
3.17	Zeroshell Captive Portal authentication Step-5 . . . . .	50
3.18	Zeroshell Captive Portal authentication Step-6 . . . . .	50
3.19	Zeroshell Captive Portal authentication Step-7 . . . . .	51

3.20	Zeroshell Captive Portal authentication Step-8	51
3.21	Zeroshell Captive Portal authentication Step-9	51
3.22	Zeroshell Captive Portal authentication Step-10	52
4.1	WiFi access test-bed setup with Polito wireless infrastructure	53
4.2	WiFi access test-bed setup with Wifi-Auth eIDAS-SP and Polito wireless infrastructure	54
4.3	Overview of authentication flow for generic scenario	55
4.4	Overview of authentication flow for Italian scenario	56
4.5	Detailed authentication flow request for Italian scenario	59
4.6	Detailed authentication flow response for Italian scenario	62
4.7	jsch library for user creation.	63
4.8	Attributes configuration file.	64
4.9	Function for creating user on WLC.	65
4.10	Function for user creation command for WLC.	66
4.11	Function for generating string for username and password.	66
4.12	Fortigate firewall interfaces	67
4.13	Fortigate firewall policies	67
4.14	Virtual IP for Wifi-Auth SP	68
4.15	Addresses and Address Group for policies	68
4.16	Wifi-Auth eIDAS-SP Captive Portal authentication Step-1	71
4.17	Wifi-Auth eIDAS-SP Captive Portal authentication Step-2	72
4.18	Wifi-Auth eIDAS-SP Captive Portal authentication Step-3	72
4.19	Wifi-Auth eIDAS-SP Captive Portal authentication Step-4	73
4.20	Wifi-Auth eIDAS-SP Captive Portal authentication Step-5	73
4.21	Wifi-Auth eIDAS-SP Captive Portal authentication Step-6	74
4.22	Wifi-Auth eIDAS-SP Captive Portal authentication Step-7	74
4.23	Wifi-Auth eIDAS-SP Captive Portal authentication Step-8	75
4.24	Wifi-Auth eIDAS-SP Captive Portal authentication Step-9	75
4.25	Wifi-Auth eIDAS-SP Captive Portal authentication Step-10	76
5.1	Fortigate-60D	79
5.2	Fortigate-60D GUI interface configuration	79
5.3	Fortigate-60D GUI policy configuration	80
5.4	Virtual IP configuration GUI	81
5.5	Cisco WLC web interface	81
5.6	Attributes configuration file.	82
5.7	ACL part 1	83
5.8	ACL part 2	83
6.1	IdP used by number of users for authentication	86
6.2	Operating system/browser used by users for authentication	86
6.3	Survey Answer: After testing this pilot, I would like the inclusion of this initiative in the academic services of other European universities.	87

# Chapter 1

## Introduction

Wireless LANs (WLAN) provides network communication to devices within a limited area such as home, restaurant, organisation etc. WLAN also referred as WiFi networks have become a basic requirement for it's enhanced mobility and increased number of devices supporting wireless connectivity. To provide a seamless connectivity within a large area such as in a university using multiple Access Points (AP) is called roaming. It is a common practice to provide roaming between different campuses of university or offices of an organisation.

Demand for secure wireless roaming between organisation have increased due to partnerships between different organisations. As well as programs like Erasmus, which provide students the facility to study or gain experience in a different European country. Therefore various initiatives have been stated to provide national and international roaming services including eduroam and govroam.

eduroam [1] is an international roaming services which provides WiFi access connectivity to users in educational sector. It provides secure network connectivity to students, teachers and researchers in institutions other than their own. govroam is an other roaming service which provides roaming services to users from different professions. It provides secure network authentication to individuals from different organisations. These services are widely used but are specific to an individual sector e.g education, government.

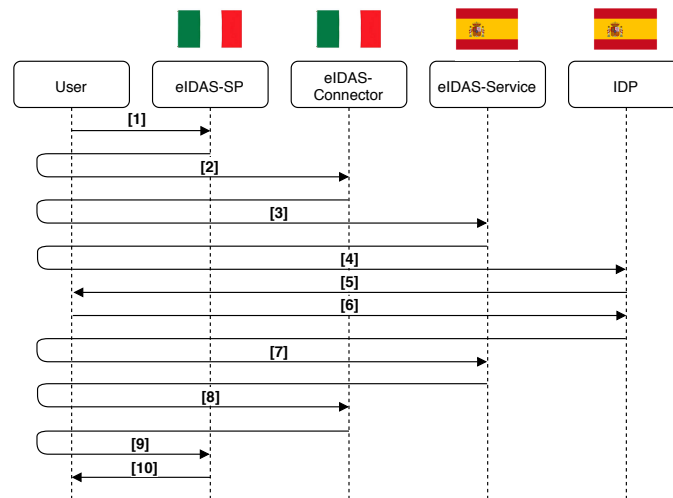


Figure 1.1: Example of eIDAS cross border authentication

In this thesis we provide a WiFi access service for network connectivity. This service is a part of European eID4U [2] project, which aims to use national electronic identities to provide advanced cross border services for European environment by integrating new attributes [3] in the eIDAS network. eIDAS provides a framework for secure cross border electronic identification between



member states using interoperability between existing national eID infrastructure. eIDAS framework is composed of eIDAS-Nodes, Service Provider (SP) and IdP (Identity Provider). Member states are responsible for implementing eIDAS nodes to securely exchange identity information between IdP and SP present in another state using eIDAS protocol. eIDAS-Connector node and eIDAS-Service node are responsible for cross border identification in the eIDAS framework. A part from these it can also include proxy nodes (specific to a country ) such as SP-Proxy which will create a bridge between SP and eIDAS-Connector or IdP-Proxy which will create a bridge between eIDAS-Service and IdP. Member states have to implement an eIDAS-Connector node which will be responsible for the communication between SP/SP-Proxy and eIDAS-Service node. Whereas eIDAS-Service node is responsible for communication between eIDAS-Connector node and IdP/IdP-Proxy.

An overview of authentication for SP in Italy using Spanish eID is shown in Figure 1.1. The authentication cycle starts with a user connecting to a eIDAS-SP. In the first step user will select Spain in the citizen country and start authentication (step 1). eIDAS-SP will generate an eIDAS *AuthnRequest* and send it to Italian eIDAS-Connector using user's browser (step 2). eIDAS-Connector will create an eIDAS *AuthnRequest* and send it to Spanish eIDAS-Service using user's browser (step 3). eIDAS-Service will create an eIDAS *AuthnRequest* for Spanish IdP and send it using user's browser (step 4). IdP will authenticate user using its authentication mechanism (credential-based/mobile-based/certificate-based) (step 5-6). IdP will then create an eIDAS *Response* for Spanish eIDAS-SP and send it using user's browser (step 7). eIDAS-SP will create an eIDAS *Response* for Italian eIDAS-Connector and send it using user's browser (step 8). eIDAS-Connector will create an eIDAS *Response* for eIDAS-SP and send it using user's browser (step 9). eIDAS-SP will get the identification of the user and will allow access to the internet (step 10).

Our goal was to create a solution for WiFi access for European environment using eIDAS framework. For which we created wireless network and setup a captive portal to only allow network access to authenticated users. We will provide two solutions, a software based and a hardware based solution for the development of eIDAS SP. First we discuss the software based solution using Zeroshell Linux distribution, which provides router and firewall services. It has the ability to create a captive portal using SAML 2.0, a service provider (SP) for authentication using Shibboleth, a DHCP server for dynamic allocation of IP address and an access control list (ACL) for allowing to access eIDAS-Nodes/IdP during authentication. Secondly we discuss the hardware based solution by using the eIDAS code and the wireless infrastructure of Politecnico di Torino for wireless access. We created a eIDAS-SP service which provides authentication using eIDAS framework for the Captive Portal. Wireless infrastructure is consisted of Cisco WLC, Cisco AP (Access Point) and Fortigate-60D, that are responsible for implementation of Captive Portal, management of users, ACL and network management.

# Chapter 2

## Background

This chapter contains concepts and underlying technologies important for the understanding Federated Identity Management.

### 2.1 User authentication

In the last couple of decades there is a massive increase in the number of organisations providing online service, with each service there is a necessity of authenticating user's identity to allow access to private resources. Recognition paradigms provides the authentication of an identity by confirming the credentials belongs to the user. Authentication protocols are the basics of user identification, because they provide the verification of the claims used in every step of the communication afterwards. Multiple approaches can be used for user authentication depending on the security constraints. Authentication can be provided using something user knows (password, key), something user is (face, fingerprint) or something user has (card) [4].

The study of exchanging secret for authentication and confidentiality of messages was an important subject in last decade. In the context of online services, secret can be exchanged in multiple ways: directly in the form Basic HTTP Authentication [5], encrypted in the form of EKE (Encrypted Key Exchange) [6] or with a challenge/response exchange of messages. The last one is very popular because it doesn't require transferring of secret on the insecure network. It provides a challenge (e.g. pseudo-random string) which is created by the private key and can only be processed if the receiver has the key. In a symmetric challenge/response exchange of messages both sender and receiver needs to know the private key to complete the verification. Whereas in an asymmetric challenge/response exchange both sender and receiver has a private key and a public key. The sender encrypts the message using its private key and receiver verifies the message using public key of the sender.

In both of these cases, symmetric or asymmetric the mechanism for securely distribution of key is very important. It can be symmetric or asymmetric with addition of a third party for distribution. Symmetric distribution can be done using **Needham-Schroeder** Symmetric Key Authentication [7] protocol, which allows to create a session key between sender and receiver without exchange of the private key. This requires a third party which will provide authenticity of the parties included by verifying the private keys. Another protocol that do not requires a third party for symmetric key distribution is ISO One-pass Symmetric Key Unilateral Authentication Protocol [8].

Asymmetric key distribution can be provided using digital certificates of identity such as X.509 [9] or using cryptography algorithm such as **Diffie and Hellman**. [10]. The digital certificates are provided by a trusted entity called Certification Authority (CA), it provides at least identifier of the subject, its public key, cryptography algorithm such as **RSA** [11] and digital signature of the Certification Authority. This used Public Key Infrastructure (PKI) [12] which guarantees identification of the subject of certificate without previous knowledge through a trusted infrastructure. The Certificate Authority issues digital certificates to the user or other intermediate

Certification Authority. The digital certificate provides trust by a chain of ordered list of certificates from user certificate to the root certificate. By trusting the root certificate means to trust in all certificates issued by the root and also by intermediate certification authorities. A part from this, Diffie and Hellman cryptography algorithm can be used to negotiate a shared session key between two parties without inclusion of a third party. Both parties shares a secret key by means of Diffie-Hellman and use it to create a private key to encrypt the communication. This private key allow them to communicate using symmetric cryptography algorithms, which by its nature can be made computationally fast by using hardware accelerator and by encrypting whole blocks.

## 2.2 Identity management

In this section we will focus on Identity Management, that refers to process for managing identities in the administrative environment, controlling access to the private resources and Identity Management Systems (IDM) which are tools and technologies used to provide identity management.

### 2.2.1 Digital identity

Each entity in Information and Communication Technology (ICT) is represented by a digital identity [13]. It contains information used to describe an internal or external entity. An entity can be user, organisation or a device. The entity requires to prove its identity by providing a secret known by the ICT system such as user credentials (username, password).

### 2.2.2 Identity Management System

In each Identity Management System there are three entities involved in the authentication cycle: user, Identity Provider and Service Provider. A user is the entity which wants to consume a service and is registered in one or more Identity Provider. Identity Provider is the entity which manages identities of a user and their respective credentials. It provides authentication and authorisation services for the user to access various services. The Service Provider is the entity which relies on Identity Provider for authentication of the user and provide services to the authenticated user.

An Identity Management System on the basis of data storage and entity roles can be classified into Isolated, Centralised and Federated Model [14].

#### Isolated model

In this model Service Provider and Identity Provider are combined together on the same server. This is the most simple approach in which both authentication and authorisation are carried out at the same point but there are several problems with this approach. For each service user wants to use, it requires to create a separate identity. If identities are created properly (separate credentials for each service), with the increase of online services it will not be possible for user to remember all those credentials and If identities are not created properly (same credentials for all services), there will be a lack of security.

#### Centralised model

In this model there is a single central Identity Provider, which provides authentication to users of multiple Service Providers. Identity Provider store identifiers/information of the user and Service Provider authenticate user using Identity Provider. A most common use of Centralised Model is single sign-on, which allows user to use multiple web services using single identification instance. In model user only need to manage one credential to use all the service but this is also a problem: as there is a single point of failure.

## Federated model

This model is an improvement of the centralised model and solves the problem of a single point of failure. In this model, entities involved in the Identity Management System have established an agreement on how the entities are going to be refereed and on configuration parameters of the entities involved in authentication. Using this model, a user can authenticate in multiple services within an organisation or across other organisations. A Service Provider in one enterprise can allow access to its services by exchanging identity, attribute, authorisation and authentication with an Identity Provider in another domain. In this way, a group of Service Providers can identify users from other Service Providers within a federated domain using a common federated Identity [22].

Entities in the Federated Model define the Federated Identity Architecture (FIA), in which Identity Providers and Service Providers exchange digital identities/information of the user preserving their privacy. The IdP and SP across enterprises in the federated domain exchange agreements and configuration in the form of metadata. Metadata provides agreement between the system entities, bindings, endpoints, certificates, keys, cryptography algorithms, security and privacy policies. Discovery and exchange of metadata make it easy to establish trust and determine policies for obtaining services.

The federated approach makes it easy to map identities across organisations and reduces the need to handle many credentials from the user perspective.

## 2.3 SAML

### 2.3.1 Introduction

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security information between online business partners. It was developed by the Security Services Technical Committee (SSTC) of the standards organisation OASIS (the Organisation for the Advancement of Structured Information Standards). This document provides a technical description of SAML V2.0.

### 2.3.2 Overview

SAML defines a XML framework for describing and exchanging cross domain authentication and authorisation information about users, usually between a Service Provider and Identity Provider. It exchanges messages in the form of SAML assertions for which the standard precisely defines the syntax and rules to be followed for requesting, creating and using these assertions.

SAML 2.0 was developed on the basis of already existing standards such as:

- **eXtensible Markup Language (XML)** [17] Defines the syntax for exchanging messages between entities
- **XML Signature** [18] Specifies the rules and syntax for calculation of signature. Which provides integrity, message authentication and signer authentication of the SAML messages.
- **XML Encryption** [19] Specifies the process for encryption and representation of the result in XML. The result of encryption data is an encryption element, which contains or references the cipher data.
- **Hypertext Transfer Protocol (HTTP)** [20] A protocol to transfer hypertext requests and information between server and browsers.
- **Simple Object Access Protocol (SOAP)** [21] A protocol used for the exchange of structured information in web services.

### 2.3.3 Drivers of SAML adoption

#### Single Sign-On (SSO))

SAML 2.0 is able to solve problem related to cross domain single sign-on by providing a protocol standard that allows to transfer information of a user from a web server to another without using the cookies. Previously most of the products claiming to provide web-based SSO use cookies to maintain user session and doesn't require to authenticate again to access the system. However, browser cookies are only accessible through the same DNS (Domain Name Service), therefore session's information is never available to other domain. Therefore they used proprietary mechanism for multi-domain SSO, which is specific to the single enterprise. This type of specific mechanism is viable in products from same enterprise but is not practical for products of business partners. Because of heterogeneous environments which make the use of proprietary protocol impossible. SAML solves this problem by providing a vendor-independent protocol for transferring session information between one web server to another independent of the DNS domains.

#### Federated identity

When different services creates a collaborative environment for mutual users, they don't only need to understand the syntax and protocol of the exchange information; they must have a common understanding of the user. Usually each provider maintains a local identifier for a user to interact with the service. In a federated identity a common shared identifier is used for identifying a user across organisational boundaries. This kind of common identity used between partners to refer a user is called federated identity. This will also decrease the management cost of individually maintaining user identity by each service. This kind of identity can reside with the user rather than on the service side.

#### Web services and other industry standards

SAML can be used to provide security assertion format to be used for non native SAML-based protocol. This allows it to be useful for other authorisation services (IETF, OASIS), identity frameworks, web services, etc. It provides a standard based approach for exchanging information, which is not easily conveyed using other WS-Security token formats. The OASIS WS-Security Technical Committee has defined a profile to provide SAML's rich assertion constructs within a WS-Security token for the transmission of SOAP messages.

### 2.3.4 SAML participants

In every SAML exchange there is a SAML asserting party and a relying party. SAML asserting party is the one which creates statement about a subject and issued in the form of statements, which are consumed by the relying party. When a asserting party or a relying party make a request to another entity, the one making the request is called requester and the one to whom the request is made is called the responder.

SAML entities can take variety of SAML roles. In the case of multi-domain Single Sign-On scenario, the entity providing the assertions is called the IdP and the one consuming the assertions is called SP. SP makes the decisions relating to the access control after authentication depending on the assertions.

At the heart of each SAML exchange there is a subject, which is trying to authenticate. A subject is a principal, which can be a human, computer or a company that needs to be authenticated to access service resources.

A typical assertion from a IdP provides the information about the subject. It can provide information about the identity of the subject and status of the authentication. SP can use the information and depending on the access policies can deny or allow access to the service resources.

### 2.3.5 Basic concepts

SAML is composed of smaller components which when combined together can be used for various use cases. These can be used for the transfer for identity, authorisation, authentication and attribute information between services from different organisation. Structure of both assertion and protocol messages for transfer of information are provided by SAML core.

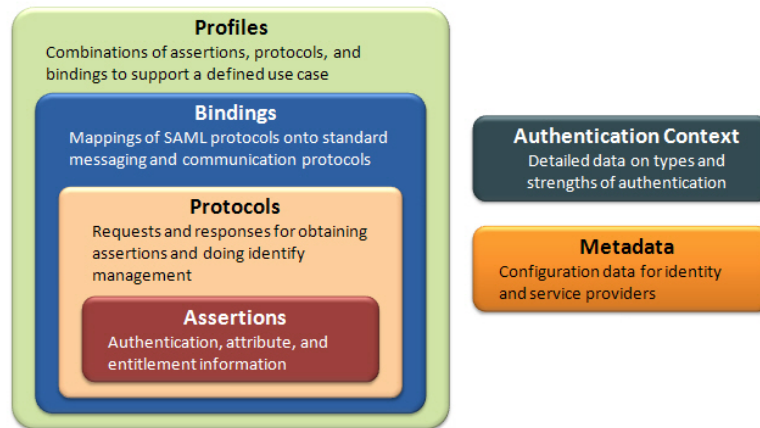


Figure 2.1: Relationship between SAML core components (source: [cse.wustl.edu](http://cse.wustl.edu)).

- **Assertions** SAML assertions contain statement about the subject, which are consumed by Service Provider to make decisions relating to access control after authentication. SAML assertions structure and contents are defined by SAML-defined assertion XML schema. These are created and digitally signed by the authentication authority (IdP), which also handles the authentication process.
- **Protocols** SAML protocols are used to for making SAML-defined request and return respective responses. SAML protocols structure and contents are defined by SAML-defined protocol XML schema.
- **Bindings** SAML bindings are the lower layer of the communication network with which SAML protocols are transferred between entities such as SOAP or HTTP.
- **Profiles** SAML profiles defines how SAML assertions, bindings and protocol must be combined for a specific use case such as multiple domain SSO. It defines constraints on the contents of these components to solve a use case in an appropriate way.
- **Metadata** Metadata is a XML document which is used to define and share configuration between SAML entities. It provides information about supported bindings, identity information, supporting identity attribute, cryptography algorithms and certificates. It is defined by it's own XML schema.
- **Authentication context** Authentication Context is used to provide details about the authentication at identity provider such as type and strength of authentication. An assertion authentication statement is used to carry this information. In some cases Service Provider can also ask Identity Provider to use specific type of authentication such as multi-factor authentication for resources that are critical. Authentication context is defined by XML schema and a set of SAML-defined Authentication Context Classes, which provide commonly used authentication methods.

### 2.3.6 SAML components

This will provide detail about each of the components that represent the assertion, protocol, binding, and profile concepts in a SAML environment.

## Assertions

SAML Assertions provide information about the principal in the form of statements. These are created and digitally signed by the authentication authority. For example when a user invokes the login operation on a Service Provider, that Service Provider redirect to the Identity Provider. Identity Provider validates the credentials and issues the SAML assertion with user information stating that the authentication was successful. Because it is digitally signed by the Identity Provider, the respective Service Provider on receiving the information can validate that it was created and issued by the user Identity Provider. SAML assertions has three kind of statements

- **Authentication statement** It provides information about the type of method used for authentication and time of the authentication for a successful operation.
- **Attribute statement** It provides user specific information of the authenticated user. Service Provider can use this information for allowing or denying access to the service resources.
- **Authorisation decision statement** It provide information about something that authenticated subject is allowed to do such as is entitled to access service resources as a user.

SAML assertion is consist of one or more statements and information relating to all common statement content. The first of these information is SAML:Assertion - that provide namespace of the assertion, version of SAML used and time of the creation of message. A SAML:Assertion tag has following syntax

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
Version="2.0" IssueInstant="2019-11-05T09:22:05Z">
```

SAML:Assertion tag contains the information of the issuer inside SAML:Issuer tag. A example of SAML:Issuer tag is as follow

```
<saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://wifi-auth-eid4u.polito.it/SP/metadata
</saml2:Issuer>
```

It also contains information about the subject inside SAML:Subject tag. It contains the name identity associated with it inside the SAML:NameID. The attribute Format is used to specify the format for the name identifier of the subject. It can specify different types of formats, such as email address, the value of the subject field of the X.509 certificate.

```
<saml:Subject>
    <saml:NameID
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
            3f7b3dcf-1674-4ecd-92c8-1544f346baf8
        </saml:NameID>
    </saml:Subject>
```

SAML:Assertion tag also contains SAML:Conditions tag which provide the conditions which are used to validate the assertions. For example in the example below it provides that the assertions are valid only if the time is in-between the *notBefore* and *NotOnOrAfter* time period.

```
<saml:Conditions
    NotBefore="20019-11-05T09:17:05Z"
    NotOnOrAfter="2019-11-05T09:27:05Z">
    <saml:AudienceRestriction>
```

It also provide SAML:AuthnStatement tag which contains information about when and with which mechanism subject is authenticated. An example is as following

```
<saml:AuthnStatement
  AuthnInstant="2019-11-05T09:22:00Z"
  SessionIndex="104e3435542aad34">
  <saml:AuthnContext>
    <saml:AuthnContextClassRef>
      urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
    </saml:AuthnContextClassRef>
  </saml:AuthnContext>
</saml:AuthnStatement>
```

The above example shows that the user is authenticated using password that was sent to the Identity Provider using secure communication channel. Other mechanism that can be used for authentication are

- **Password** In this authentication is done using password over an insecure communication channel HTTP.
- **Internet protocol** In this authentication is provided using IP address of the user.
- **Internet protocol password** In this authentication is provided using both IP address and password of the user.
- **Previous session** In this authentication is provided by a previous authenticated session of the user.

In addition assertions also provide additional information about the user form Identity Provider to Service Provide in SAML:AttributeStatement tag. This transfer of extra information is a powerful SAML feature that can be used for allowing access to the resources. These can be use to provide profile information of the user, which can be used to create a local account of the user or if required to provide consent of the usage of the information. For example to apply for driving license it requires the age of the user to verify that user is above 16 and is eligible for driving. These information can also be used to provide access to service resources depending on the attribute. The Service Provider and Identity Provider need to agree on the attribute name and values in the assertions.

```
<saml:AttributeStatement>
  <saml:Attribute Name="uid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
    format:basic">
    <saml:AttributeValue xsi:type="xs:string">test</saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute Name="mail" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
    -format:basic">
    <saml:AttributeValue xsi:type="xs:string">test@example.com</saml:
      AttributeValue>
  </saml:Attribute>
  <saml:Attribute Name="eduPersonAffiliation" NameFormat="urn:oasis:names:tc:
    SAML:2.0:attrname-format:basic">
    <saml:AttributeValue xsi:type="xs:string">users</saml:AttributeValue>
    <saml:AttributeValue xsi:type="xs:string">example1</saml:
      AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
```

## Protocols:

SAML Protocols defines request/response protocols for SAML elements including assertions. It also provides rules that SAML entities must follow in order to exchange messages. The following protocols are defined in SAML 2.0



- **Authentication request protocol** This protocol is used to support web Browser SSO Profile to establish a security context for the user at Service Provider. In this a (*AuthnRequest*) message is issued from Service Provider to Identity Provider to create a (*Response*) message containing one or more assertions pertaining to a principal.
- **Single logout protocol** This protocol is used to logout principal from all authenticated sessions. The logout can be initiated by the principal or because of a session time-out from Service Provider or Identity Provider.
- **Assertion query and request protocol** This provides a set of queries by which assertion may be obtained. The Request form of this protocol defines how a relying party (SP) can ask for an assertion to an asserting party (IdP) providing its assertion ID. The Query form of this request provides how a relying party can ask for assertion on the basis of a reference, subject or the statement type.
- **Artefact resolution protocol** This provides a mechanism to obtain a previously created assertion by providing a reference to relying party. The artefact is a small, fixed-length value used as a reference in SAML. The SAML protocol can refer to a artefact and Service Provider can obtain the assertion by using the artefact protocol. The artefact is typically passed using one SAML binding, such as HTTP-Redirect while resolution of the artefact take place over synchronous binding, such as SOAP.
- **Name identifier management protocol** This provides the mechanism to change the value or format of the the name identifier of the principal. It can be issued by a Service Provider or a Identity Provider. This protocol can also be used to terminate an already existing association of a name identifier between Service Provider and Identity Provider.
- **Name identifier mapping protocol** This provide the mechanism to map SAML name identifier from one Service Provider to another Service Provider. This can be used to provide interoperability between different Service Provider in an application integration environment.

## Bindings

SAML bindings provides in detail how the various SAML protocol messages can be transported in the the lower network communication layer. These are the bindings provided by SAML 2.0

- **HTTP redirect binding** This provides a mechanism to transport SAML protocol messages using HTTP redirect messages (302 status code response).
- **HTTP POST binding** This provides a mechanism to transport SAML protocol messages using base64-encoded content in HTML form control.
- **HTTP artefact binding** This provides a mechanism to transport artefact using HTTP protocol either using HTML form control or a query string in the URL from a sender to a receiver.
- **SAML SOAP binding** This provides a mechanism to transport SAML protocol messages using SOAP over HTTP.
- **Reverse SOAP (PAOS) binding** This provides a mechanism to exchange SOAP/HTTP messages in a multi-stage environment. This is used in enhanced Clients and Proxy profile to assist in IdP discovery.
- **SAML URI binding** This provides a mechanism to obtain a existing SAML assertion resolving a URI.

## Profiles

SAML Profiles define in details how the SAML assertion, protocol and bindings must be combined for a particular scenario. The primary use case of SAML Profile is Web Browser SSO. These are the Profiles provided in SAML 2.0

- **Web browser SSO profile** This Profile defines how Service Provider and Identity Provider uses SAML protocol for authentication request, SAML response messages and assertions to achieve single sign-on with standard web browser. It defines how these messages are used in combination HTTP POST, HTTP Redirect and HTTP Artefact bindings.
- **Enhanced Client and Proxy (ECP) profile** This profile defines a specialised single sign-on in which Reverse-SOAP (PAOS) and SOAP bindings are used by specialised clients or gateway proxies.
- **Identity provider discovery profile** This Profile defines a way in which the Service Provider can use the common domain cookies to check which of the Identity Providers are already visited by a user. When a request is received from a user, Service Provider requests the common domain cookie read service to check the Identity Providers previously visited.
- **Single logout profile** This Profile defines how Single Logout Protocol can be used using HTTP Redirect, HTTP Post, SOAP and HTTP Artefact bindings.
- **Assertion query/request profile** This Profile defines how to use SAML Query and Request Protocol for SAML assertions over a synchronous binding, such as SOAP.
- **Artefact resolution profile** This Profile defines how Artefact Resolution Protocol can be used to obtain a artefact over a synchronous binding, such as SOAP.
- **Name identifier management profile** This Profile defines how Name Identifier Mapping Protocol can be used with HTTP POST, HTTP Redirect, SOAP and HTTP Artefact bindings.
- **Name identifier mapping profile** This Profile defines how Name Identifier Mapping Protocol can be used over a synchronous binding, such as SOAP.

### 2.3.7 Privacy in SAML

In an information technology context, privacy generally refers to both a user's ability to control how their identity data is shared and used, and to mechanisms that inhibit their actions at multiple service providers from being inappropriately correlated.

SAML is often deployed in scenarios where such privacy requirements must be accounted for (as it is also often deployed in scenarios where such privacy need not be explicitly addressed, the assumption being that appropriate protections are enabled through other means and/or layers).

SAML has a number of mechanisms that support deployment in privacy .

- SAML supports the establishment of pseudonyms established between an identity provider and a service provider. Such pseudonyms do not themselves enable inappropriate correlation between service providers (as would be possible if the identity provider asserted the same identifier for a user to every service provider, a so-called global identifier).
- SAML supports one-time or transient identifiers - such identifiers ensure that every time a certain user accesses a given service provider through a single sign-on operation from an identity provider, that service provider will be unable to recognise them as the same individual as might have previously visited (based solely on the identifier, correlation may be possible through non-SAML handles).
- SAML Authentication Context mechanisms allow a user to be authenticated at a sufficient (but not more than necessary) assurance level, appropriate to the resource they may be attempting to access at some service provider.

- SAML allows the claimed fact of a user consenting to certain operations (e.g. the act of federation) to be expressed between providers. How, when or where such consent is obtained is out of scope for SAML.

### 2.3.8 Security in SAML

Just providing assertions from an asserting party to a relying party may not be adequate to ensure a secure system. How does the relying party trust what is being asserted to it? In addition, what prevents a “man-in-the-middle” attack that might grab assertions to be illicitly “replayed” at a later date? A brief description of these mechanisms are provided below.

SAML defines a number of security mechanisms to detect and protect against such attacks. The primary mechanism is for the relying party and asserting party to have a pre-existing trust relationship which typically relies on a Public Key Infrastructure (PKI). While use of a PKI is not mandated by SAML, it is recommended.

Use of particular security mechanisms are described for each SAML binding. A general overview of what is recommended is provided below:

- Where message integrity and message confidentiality are required, then HTTP over SSL 3.0 or TLS 1.0 is recommended.
- When a relying party requests an assertion from an asserting party, bi-lateral authentication is required and the use of SSL 3.0 or TLS 1.0 using mutual authentication or authentication via digital signatures is recommended.
- When a response message containing an assertion is delivered to a relying party via a user’s web browser (for example using the HTTP POST binding), then to ensure message integrity, it is mandated that the response message be digitally signed using XML Signature

## 2.4 Shibboleth

Shibboleth is an open source SAML implementation by the Shibboleth Consortium. It provides the development of a Shibboleth SP and IdP among others [23]. It provides authentication, authorisation and single sign-on services across a range of Identity Providers. It is composed of different components: Identity Provider, Service Provider and Discovery Services (DS). These can be deployed together or separately to provide specific functionality to an organisation. It supports most of the profile defined by SAML 1.1 and SAML 2.0.

## 2.5 eIDAS

Since the start of the XXI century European countries have developed digital identities for their nationals such as Italian SPID [24], Portugal-Chave Móvel Digital [25] and Spain-DNIe [26]. These lead to a very diverse landscape of electronic identity management. Therefore to provide cross border identification for European citizens, multiple initiatives were stated to provide federated identity management using existing electronic Identification (eID) systems.

STORK [27] and STORK 2.0 [28] projects showed that it is possible to create an electronic identity European infrastructure to provide electronic authentication and attribute management by providing interoperability between existing national eID. The feasibility of these projects became the base of the federated identity solution [29] in anticipation of the adoption of the Regulation (EU) 910/2014 [30], also called eIDAS Regulation. Which will be in force in all member states. This regulation provides a framework to provide secure and trusted electronic identification across European countries by providing interoperability between existing National eID. This creates a European level trust network for electronic services across borders. This Regulation also ensures that this electronic identification has the same legal status as traditional paper.

This federated identity solution consisted of member states, each member state is responsible to deploy a eIDAS Node to act as a Identity Provider for national eID scheme for any country. To maintain the security and trustfulness of the network, a governmental institution is responsible for the implementation and deployment of the eIDAS Node. All the Service Provider in the network are subscribed to the eIDAS Node of the country. Through this federated infrastructure every citizen of a member state is able to consume services in another member state.

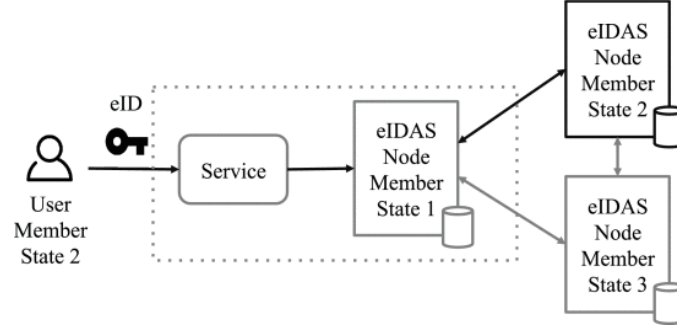


Figure 2.2: eIDAS basic architecture (source: [IEEE](#))

### 2.5.1 Main problems addressed by eIDAS

The key agenda of the eIDAS regulation was to provide a federated authentication system for cross border authentication. This will allow the citizens of each member state to authenticate themselves using their national eID across all member states. Which will allow public and private businesses to provide their services to all European citizens through a secure and trustworthy network.

This will also provide a framework for legal authentication across borders of member states, which has the same legal status as the traditional paper documentation. The eIDAS regulation provides these three key elements:

- “It upgrades the legal framework of electronic signatures replacing, the existing eSignature Directive. For instance, it allows you to “sig” with a mobile phone; it requires higher accountability for security; and it provides clear and stronger rules for the supervision of eSignature and related services.
- Through requiring mutual recognition between various national eID systems (different to harmonisation or centralisation), the Regulation extends the capabilities - the opportunities available with your existing eID - by making it functional across EU borders.
- Other trust services are included in the Regulation for the first time, meaning there will be a clear legal framework and more safeguards through strong supervision services of electronic seals, time stamping, electronic document acceptability, electronic delivery and website authentication.” [31]

### 2.5.2 eIDAS specification

The eIDAS specification defines that eIDAS Node can act in two federated modes: eIDAS-Connector and eIDAS-Service. All the Service Providers are subscribed to eIDAS-Connector Node in the same country and all the Identity Providers are subscribed to eIDAS-Service Node of that country. eIDAS-Connector Node will be responsible to request authentication across member states and eIDAS-Service Node will be responsible to provide authentication across member states. This interconnection of secure and trusted nodes across borders provides a trust network for cross border authentication, which is shown in Figure 2.3.

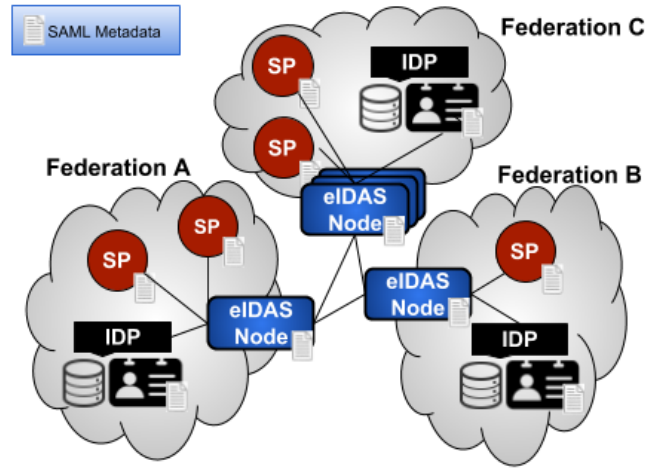


Figure 2.3: eIDAS architecture for cross border authentication (source: IEEE access [32])

The infrastructure is based on SAML 2.0 and defines a set of profiles/bindings that must be supported in order to implement network. Each Service Provider is subscribed to a Node-Connector in that member state, each Node-Connector is connected to Node-Services of all member state and each Node-Service is connected to Identity Providers in that member state. Identity Provider should support SAML profiles and bindings defined in the eIDAS specifications to be compatible with other nodes present in the federated network. It can implement its own mechanism for authentication and authorisation, but has to provide translation between SAML assertions.

To enable interoperability between eID schemes of member states, a process known as “notification” is defined for formalising the approved eID schemes. Once the Member State meet all the eIDAS security and quality requirement, it notifies its own eID scheme. Once the eID scheme is officially added, the member states have to recognise it “no later than 12 months after the publication to the Official Journal of the European Union” [33]. Figure 2.4 shows eID schemes status of Member States.

Each entity involved in the eIDAS network publish a standardised signed SAML metadata file to negotiate agreements between system entities about Identifiers, binding endpoints, certificates, cryptography algorithms, security and privacy policies [35]. SAML metadata is part of the SAML specification and it is closely related to eIDAS framework standard using SAML as format for message exchange. It is supported by SAML implementation and libraries.

### 2.5.3 eIDAS cryptography requirements for trust between eIDAS entities

The communication between eIDAS entities is performed via user’s browser. To secure the communication the SAML messages are protected with cryptography and transported using TLS [36] on the transport layer.

If supported by user’s browser, eIDAS-Nodes must only use TLS version 1.2. Otherwise it can use TLS version 1.1. It must use cipher suites that provide perfect forward secrecy. Recommending cipher suites are provided in the Figure 2.5. eIDAS-Nodes also should not use TLS compression, heartbeat extension [38], Session Renegotiation and truncated HMAC [39].

SAML is used for confidentiality, integrity of the information transferred and identification of the communication endpoints. It uses authentication based on X.509 certificates for XML Encryption and XML Signatures. In eIDAS, entities must sign the SAML request/response and encrypt SAML assertion within the SAML response. For XML Encryption/Signature, SHA-2 algorithm must be used with minimum length of 256.





























		1st Notification	High	Substantial	Low
Austria					
Belgium		27 Dec 2018	X		
Bulgaria					
Croatia		07 Nov 2018	X		
Cyprus					
Czechia		13 Sep 2019	X		
Denmark					
Estonia		07 Nov 2018	X		
Finland					
France					
Germany		26 Sep 2017	X		
Greece					
Hungary					
Ireland					
Italy		10 Sep 2018	X	X	X
Latvia					
Lithuania					
Luxembourg		07 Nov 2018	X		
Malta					
Netherlands		13 Sep 2019	X	X	
Poland					
Portugal		28 Feb 2018	X		
Romania					
Slovakia					
Slovenia					
Spain		07 Nov 2018	X		
Sweden					
United Kingdom		02 May 2019		X	X
Number of Countries			10	3	2

Figure 2.4: eID Schemes notified in Europe (source: EUROSMT [34])

	Key agreement and authentication mechanisms		Encryption	Mode of operation	Hash
TLS_	ECDHE_ECDSA_	WITH_	AES_128_	CBC_	SHA256
			AES_256_	GCM_	SHA384
	ECDHE_RSA_	WITH_	AES_128_	CBC_	SHA256
			AES_256_	GCM_	SHA384
	DHE_RSA_	WITH_	AES_128_	CBC_	SHA256
				GCM_	SHA384

Figure 2.5: TLS recommended cipher suites (source: eIDAS [37])

## XML Encryption with SAML

For XML Encryption a hybrid cryptography system is used. For each transmission a random symmetric key (session key) is generated to encrypt SAML assertion via symmetric algorithm and symmetric key is encrypted with the public key of the receiver entity. For Encryption of SAML assertion entities must use one of the algorithm provided below

- <http://www.w3.org/2009/xmlenc11#aes128-gcm>
- <http://www.w3.org/2009/xmlenc11#aes192-gcm>
- <http://www.w3.org/2009/xmlenc11#aes256-gcm>

For key Encryption entities must use either key transport or key agreement mechanism. In case of key agreement, the session key is encrypted using the public key of the receiver X.509 certificate. Otherwise a symmetric key pair is derived by means of a ECDH key agreement using an ephemeral key pair and the public key of the receiver X.509 certificate. Then the derived key is used to wrap the session key.

### XML signature for SAML and SAML metadata

SAML messages and metadata are signed for verification of the entities involved in the communication. For the generation/verification of the signatures, one of the following algorithms provided in the Table 2.1 must be used.

Algorithm	Minimal key length
RSASSA-PSS	3072
ECDSA	256

Table 2.1: Signature algorithms for SAML

### 2.5.4 eIDAS protocol

1. To access a web-based service using eIDAS framework, the SP asks user to select the home country in which he/she will authenticate. Upon country selection, the SP retrieves the respective eIDAS-Connector metadata and validates it using its metadata signing certificate stored locally in SP. Next it gets the endpoint from metadata file, generates a eIDAS authentication request (*AuthnRequest*), which is a special type of SAML2 *AuthnRequest*, next it digitally signed it with SP (asymmetric) “signing” private-key stored locally, and sent it to the eIDAS-Connector Node through the user’s browser. The eIDAS *AuthnRequest* contained a set of identification and authentication attributes requested by the SP to grant access to the service, e.g. **CurrentFamilyName**, **CurrentGivenName**, **DateOfBirth**, **PersonIdentifier**. It also send the *RequestedAuthnContext* level i.e. the minimum authentication level required to authenticate the user. The citizen country was sent to the eIDAS-Connector as well, e.g. as a parameter in the HTTP POST request: country: IT.
2. Next, the eIDAS-Connector Node validates the SP by retrieving and validating metadata with SP metadata signing certificate stored locally. Then it validates the *AuthnRequest* received from the SP by using its “signing” public-key, which is provided by the metadata file of the SP. After validation of the eIDAS *AuthnRequest*, eIDAS-Connector Node creates a new eIDAS *AuthnRequest* depending on the attribute received from SP, which was digitally signed with the eIDAS-Connector (asymmetric) “signing” private-key and subsequently it was sent to the eIDAS-Service Node depending on the home country of the user by using the SAML2 POST Binding.
3. The eIDAS-Service Node validates the eIDAS-Connector by retrieving and validating metadata with eIDAS-Connector metadata signing certificate stored locally. Then it validates the eIDAS *AuthnRequest* received from eIDAS-Connector Node by using its (asymmetric) “signing” public-key, which is provided by the metadata file. After that it constructs a new eIDAS *AuthnRequest*, which was digitally signed with the eIDAS-Service (asymmetric) “signing” private-key. It can ask the user to select the IdP to which it wants to authenticate and sent the eIDAS *AuthnRequest* to the respective IdP using the SAML2 POST Binding. For example, in Italy, the support for multiple IdP is provided and user can select from a list of IdPs at Italian eIDAS-Service node.
4. The IdP validates the eIDAS-Service by retrieving and validating metadata using eIDAS-Service’s metadata signing certificate stored locally. Then it validates the eIDAS *AuthnRequest* received from eIDAS-Service Node by using its (asymmetric) “signing” public-key, which is provided by the metadata file. Next it authenticates the citizen by using

national authentication credential(s) and an authentication process/protocol specific to each country.

5. After successful authentication the IdP constructs a eIDAS *Response*, which was also a special type of SAML2 *Response* element. The authentication response included the requested identification and authentication information in a SAML2 *Assertion* element. Every identification and authentication attribute was packaged in a specific SAML2 *Attribute* element, which has been defined in the SAML2 eIDAS profile. Then the eIDAS attributes element is encrypted by a session key generated randomly and session is encrypted using (asymmetric) “encryption” public key of the eIDAS-Service Node, which is provided by the metadata file. Next the authentication response was digitally signed by the IdP with its private-key stored locally and was sent to the eIDAS-Service Node by using the SAML2 POST binding provided by the metadata file of the eIDAS-Service Node.
6. The eIDAS-Service Node validates the IdP by retrieving and validating metadata with IdP metadata signing certificate stored locally. Next it validates the signature on the SAML2 *Response* using IdP public-key, which is provided in the metadata file. Then eIDAS-Service Node decipher the SAML2 *Response* session key using its (asymmetric) “encryption” private-key and use the session key to decipher encrypted attributes element to get the identification/attributes information. All the attributes were extracted from the SAML2 *Assertion* element, then they were *filtered*, meaning that attributes that have not been requested were discarded, while the ones that have been requested were checked if they have been valued. In case mandatory attributes have not been returned, an error was generated and the transaction stopped. In addition, the eIDAS-Service requires the user’s consent to forward his attributes to the eIDAS-Connector Node. If the consent was given, the eIDAS-Service encrypts SAML attributes using a session key generated randomly and session is encrypted using (asymmetric) “encryption” public key of the eIDAS-Connector Node and creates a newly signed SAML2 *Response* using its (asymmetric) “signing” private-key. Next, sent it to the eIDAS-Connector through the user’s browser with the HTTP POST binding.
7. The eIDAS-Connector Node validates the eIDAS-Service by retrieving and validating metadata with eIDAS-Service metadata signing certificate stored locally. Next it validates the signature on the SAML2 *Response* by using the public key of eIDAS-Service Node provided in the metadata file available online, the eIDAS-Connector Node decipher the SAML2 *Response* session key using its (asymmetric) “encryption” private-key and use the session key to decipher encrypted attributes element to get the identification/attributes information. Next, it extracts all the attributes from the SAML2 *Assertion* element and mapped them (if necessary) to a format recognised by the SP. Since eIDAS protocol was used in the communication between SP and eIDAS-Connector, no mapping/filtering of attributes had to be further done on eIDAS-Connector Node. The eIDAS-Connector Node creates a newly signed and encrypted SAML2 *Response* containing the valued attributes and sent it to the SP, where the certified attributes were extracted and verified to grant access to the user.

### 2.5.5 Attributes

eIDAS extended the SAML 2.0 format to hold the most common personal user attributes, e.g. `PersonIdentifier`, `CurrentFamilyName`, `CurrentGivenName`, `DateOfBirth`, or even the `marital status`, `TaxReference` and `Nationality`. Moreover, it also defined the authentication **Level of Assurance** (LoA) level indicating the quality of authentication being requested. To establish trust, the authentication methods used by the national eID systems were classified into three LoA assurance classes (low, substantial, and high). The details of the LoA levels are provided in Figure 2.6 [41]. To provide access to a service, an SP can request a certain LoA level in the authentication request, which was sent to the IdP in charge of the actual authentication process of the citizen. If the IdP authenticated the citizen with a method whose assurance level met the requested LoA (or above), then the authentication response and the attributes were sent to the SP, otherwise an error was generated and the transaction was stopped.

Note that most of these personal attributes had a simple structure, typically each attribute had one value. A selection of some important attributes for natural persons used for cross-border



Level of Assurance	Identity assurance (identity proofing at registration)	Authentication assurance
Low	Present ID from authoritative source (remote or in-person)	Single factor (e.g., password or PIN)
Substantial	<ul style="list-style-type: none"> <li>· Present ID (remote or in-person)</li> <li>· ID verification performed by registration authority</li> </ul>	Multi-factor (e.g., mobile phone + PIN)
High	<ul style="list-style-type: none"> <li>· In-person ID proofing at registration authority</li> <li>· ID verification using official government sources and documents</li> </ul>	<ul style="list-style-type: none"> <li>· Multi-factor</li> <li>· Must access private data/keys stored on tamper-resistant hardware token</li> <li>· Cryptographic protection of personally identifying information (PII)</li> </ul>

Figure 2.6: eIDAS authentication Level of Assurance (source: [SCRIVE](#))

identification and authentication is presented in Table 2.2, while the complete set of attributes defined in eIDAS is found at [40].

SAML Attribute	Description
PersonIdentifier	Unique identifier of a natural person
CurrentGivenName	Given name of a natural person
CurrentFamilyName	Family name of a natural person
DateOfBirth	Date of birth of a natural person

Table 2.2: A selection of natural person attributes in eIDAS infrastructures

The eIDAS infrastructure supports a number of person attributes to be exchanged through the eIDAS nodes. The so-called eIDAS minimum data set for natural persons contains the following mandatory attributes: unique identifier (**PersonIdentifier**), current family name(s) (**CurrentFamilyName**), current first name (s) (**CurrentGivenName**), date of birth (**DateOfBirth**). Additionally, the following attributes are also defined for persons, but they are considered optional: first name(s) and family name(s) at birth, place of birth, current address and gender. eIDAS defines also attributes for legal persons, e.g. the legal name, legal address, VAT registration number, the tax registration number or the legal entity identifier. The eIDAS Regulation establishes three levels of assurance (LoA) for identification schemes that are directly proportional to their legal value: low, substantial and high.

## 2.6 Related work

In the last couple of decades due to the rapid increase in the number of wireless devices, which lead to many initiatives to provide federated solution for wireless roaming. We will provide an overview of the technologies that are related to the work done in this thesis and are already available.

### 2.6.1 eduroam

eduroam (education roaming) is a solution proposed by the TERENA Task Force on Mobility [43] to provide a international roaming network. It would provide Internet access securely at academic campuses across Europe to the users of National Research and Educational Networks (NRENs). eduroam works as the authentication of a user is carried out at the home institution of the user using home institution credentials remotely and authorisation to allow access to the resources is managed by the visited institution by trusting the infrastructure.

## Introduction

eduroam is a international roaming network to allow secure Internet access to the user in research and higher education across academic campuses. To develop eduroam, TERENA has developed requirement specifications for the infrastructure with the following characteristics [44]

- Overhead of managing each user
- Ease of usability
- Security requirements for all entities
- Ease of scaling of network

The architecture that would be able to abide by these characteristics is based on a number of technologies and agreements, which would provide the eduroam experience “open your laptop and be online” [45]. These are defined by a set of technical and organisational requirements, which should be agreed upon by each member of the eduroam Confederation (by signing and following the European eduroam Confederation policy declaration) [46]. The crucial mechanism underpinning the authentication and authorisation of eduroam involves [47]

- Authentication of a user provided by an Identity Provider using its own authentication mechanism.
- After successful authentication, proper authorisation of the user at Service Provider to allow access to the network resources.

TERENA Task Force Mobility finalise 802.1X [48] authentication with a RADIUS (Remote Authentication Dial-In User Service) [49] hierarchy based back-end for the development of eduroam because of the fact that WPA and 802.11i security standards are all build on 802.1X. Even though not all institutions supports it because of the their legacy equipment.

## eduroam specification

The European eduroam service provides this facility as a confederated service, built hierarchically. At the top level sits the confederation level service, which primarily provides the confederation infrastructure required to grant network access to all participating members of the eduroam service at any time. This confederation service is built upon the national roaming services, operated by the national roaming operators (NROs) (in most cases, NRENs). National roaming services make use of other entities, for example, campuses and regional facilities.

A hierarchy consists of RADIUS servers at the participating institutions, national RADIUS servers and regional top-level RADIUS are used to transport authentication request from the visited institution to the home institution of the user. Each institution included in the network has a Radius server with the local database of the users of that institution. It is responsible to provide the identification of its users.

Because the user’s credentials travel via a number of intermediate servers, not under the control of the home institution of the user, it is important that the credentials are protected. This requirement limits the types of authentication methods that can be used. Basically there are two categories of useful authentication methods: those that use credentials in the form of some public-key mechanism with certificates and those that use so-called tunnelled authentication. Most institutions use a tunnelled authentication method that only requires server certificates. These server certificates are used to set up a secure tunnel between the mobile device and the authentication server, through which the user credentials are securely transported.

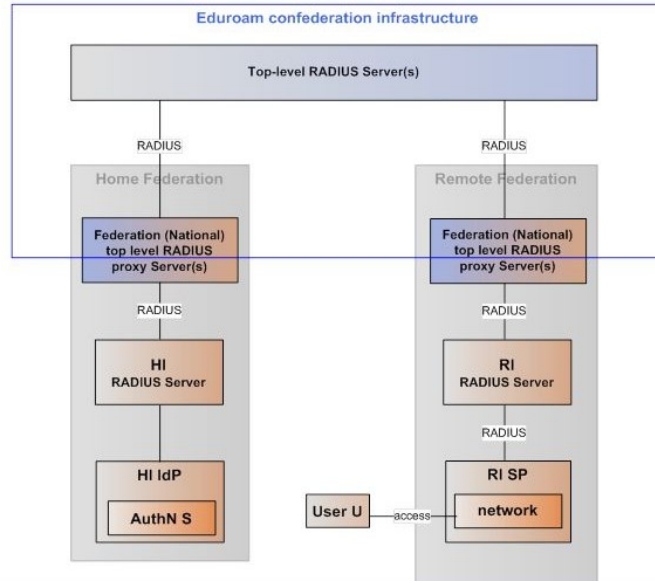


Figure 2.7: eduroam confederation structure (source: [GEANT](#)).

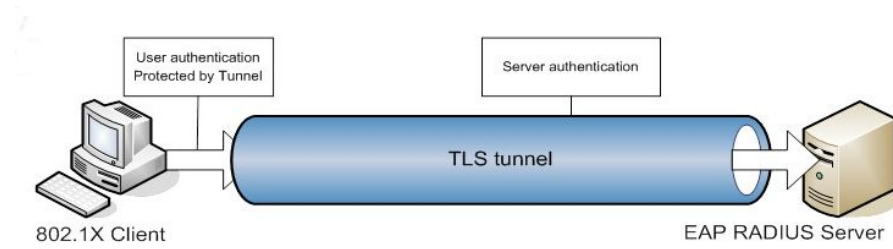


Figure 2.8: Tunnelled authentication (source: [Alfa&Ariss](#)).

### eduroam protocol

1. The authentication starts with a user with username “toto@institution\_b.be” from institution B trying to authenticate at institution A. From realm **institution\_b.be** (realm is the home institution’s DNS domain name often of the form institution.tld (tld=top-level domain; both country-code TLDs and generic TLDs are supported)) of the user, institution b gets the institution of the user. The user is from another institution so it proxies to the national RADIUS server.
2. The national Radius server checks that institution B is in another country, it will proxies it to regional top-level Radius server.
3. The regional top-level Radius server will proxies it to the national Radius server of the country in which institution B is located.
4. The national server has the list of the all the institutions participating in the eduroam network in that country and forwards the credentials to the home institution B.
5. The Radius server of the institution B verifies the credentials and “acknowledge” of the authentication travels back over the proxy-hierarchy to the visited institution A and the user is granted access.

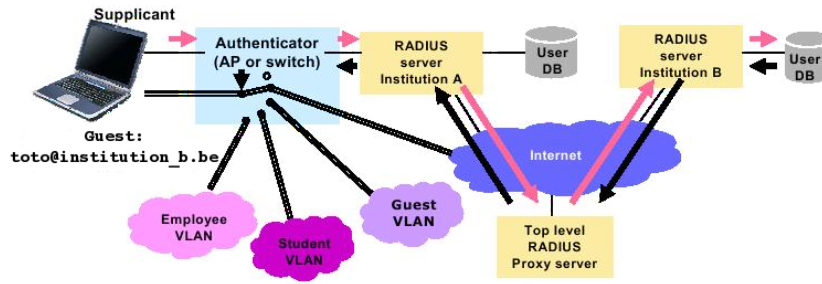


Figure 2.9: eduroam basic authentication flow (source: [Belnet](#)).

### Elements of the eduroam infrastructure

- European Top-level RADIUS Servers (ETLRS)** Currently, the European Top-level RADIUS Servers (ETLRS) for the European Confederation are located in the Netherlands and Denmark. Each server has a list of connected, federation top-level domains (.nl, .dk, .hr, .de, .be etc.) serving the appropriate NRENs. The servers also maintain exception rules for domains whose federation membership is not immediately identifiable in the realm (typically gTLD realms such as ‘.edu’, ‘.eu’, ‘.net’, etc.). The servers accept requests for the federation domains they are responsible for, and subsequently forward them to the associated RADIUS server for that federation, and transport the response (i.e. result of the authentication request) back. Requests for the federation domains that the servers are not responsible for are forwarded to the proper federation TLRS. As well as European NRENs, there are eduroam participants in other parts of the world (.au, .jp, .cn etc). These realms are also handled by the TLRS in Europe (ETLRS), although these NRENs are not members of the European confederation.
- Federation-level RADIUS Server (FLRS)** A federation RADIUS server has a list of connected eduroam IdP servers and their associated realms, as well as the connected eduroam Service Providers within a federation. It is connected to the ETLRS. The purpose of the FLRS is to receive requests from the ETLRS and eduroam SPs, and forward these requests to the responsible eduroam Identity Provider (either using static routing, or by performing DNS lookups for dynamic request routing).
- eduroam Identity Providers (IdPs)** An eduroam IdP’s RADIUS server is responsible for authenticating its own users (at home or remotely when visiting another institution) by checking the credentials against a local Identity Management System. The Identity Management System contains information on end users (for example, usernames and passwords). They must be kept up-to-date by the eduroam Identity Provider. Note that the eduroam Identity Provider’s RADIUS server has the most complex task of all. Whereas the other RADIUS servers merely proxy requests, the Identity Provider’s server also needs to actually authenticate users, and therefore, needs to be able to terminate EAP requests and perform identity management system lookups.
- eduroam Service Providers (SPs)** An eduroam Service Provider’s (SP) RADIUS server is responsible for forwarding requests from users visiting this SP to the responsible eduroam IdP, either by forwarding the request along the hierarchy, or by discovering the responsible server with DNS. Upon proper authentication of a user, the eduroam SP’s RADIUS server may assign a VLAN to the user.

Small SP that do not require VLAN assignment do not necessarily need their own RADIUS server, and can instead connect their network access elements to the respective FLRS.

In most cases, an educational institution participating in eduroam acts as an IdP and SP at the same time.

## 2.6.2 govroam

govroam (government roaming) [50] is another solution by TERENA to provide roaming for government agencies. It is developed on the same architecture as eduroam. It started from Netherlands and is adopted by many countries. The operations of govroam are still managed by TERENA worldwide. govroam is based on policies and principles provided by “NL Service Policy” document [51], which has to be agreed and followed by all the organisations participating in the roaming network.

govroam supports the trend toward multi-disciplinary activities across organisations, such as the convergence of health and social care. It make possible connectivity to the Internet and availability of resources present at home organisation by connecting from the network of a visiting organisation. The staff can access the govroam network provided by any participating organisation, using a single, securely-authenticated sign-on managed by their organisation.

### govroam specification

govroam is developed by creating a trust relationship between the organisation present in the network. NL policy document provides responsibilities of the entities involved in the federated network, so all entities have a clear understanding of their responsibilities to provide there services in a trusted manner. These responsibilities are also enforced by sanctions against organisations that do not comply with these policies and results in suspension or ejection from the govroam community.

govroam is a loose federation of related organisations. In order to work successfully, it depends on an implicit tripartite trust relationship between an IdP, a SP and the RO (Roaming Operator). The IdP advertises the govroam service to its users, and trusts that the SP will provide the service in a manner consistent with expectations, recognising that its users will sometimes rely on govroam services to the exclusion of making other arrangements. IdPs further trust that SPs will secure their users’ credentials and respect the confidentiality of their users’ communications.

SPs trust that the user identities asserted by an IdP are verified members of their organisation in good standing, and that an IdP has a contractual hold over those users in the form of an acceptable use policy or equivalent. SPs trust that IdPs will take action in terms of their organisational policies should abuse be reported. Some SPs have legal or governance obligations to retain information about the people they provide their service to, and trust that IdPs will do so on their behalf in exchange for reducing the complexity of gaining access.

Both IdPs and SPs trust RO to both provide the necessary infrastructure and oversight, and to respect the privacy of their respective users and their communications.

### Elements of govroam infrastructure

1. **govroam Identity Provider (IdP)** It is responsible for authenticating its own users by checking the credentials against a local identity management system. IdPs assert the identity of their users to govroam Service Providers when required. As they hold information about the organisation a user is affiliated with, IdPs are often referred to as a user’s Home Organisation or Home Institution and the terms are sometimes used interchangeably.
2. **govroam Service Provider (SP)** It maintains a network and provides Internet access, usually wirelessly, to govroam visitors from other organisations once they are successfully authenticated. For this reason, Service Providers are often and interchangeably referred to as a user’s Visited Organisation or Visited Institution.
3. **Roaming Operator (RO)** It performs a coordinating role - it provides RADIUS proxy servers to ensure that authentication requests from the SP reach the right IdP, which in the future may involve govroam them to other Roaming Operators in other countries. The Roaming Operator also maintains governance and oversight of govroam within the country in which they operate. In The Netherlands the Roaming Operator is the Foundation

Government Roaming Nederland (the Foundation). In the course of their on-going collaboration, the Foundation has assigned the provision of some aspects of the govroam services to SURFnet, who also performs operational tasks for the Roaming Operator of govroam.

Because other countries are also starting govroam services it is foreseen that, likewise the eduroam structure, a future Roaming Confederation (RC) may bring together a number of ROs serving a geographical region is. This is yet to be decided.

## 2.7 Review of possible networking tools for implementing Captive Portal with SAML

### 2.7.1 PacketFence

PacketFence [52] is an open source, trusted, free and full support tool for network access control (NAC) solution. It provides the capability to create Captive Portal for registration and remediation. It also features a unified wired and wireless management, 802.1X support, layer-2 isolation of problematic devices. It can be integrated with other network security applications [54] such as Snort IDS and the Nessus vulnerability scanner to provide an effective secure network for small to very large organisation network [53].

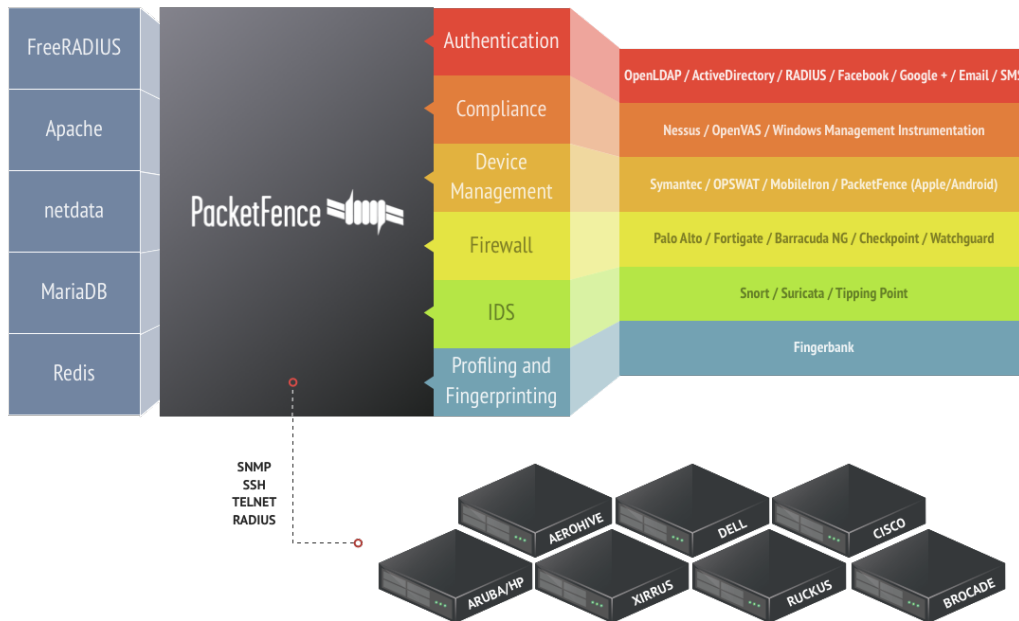
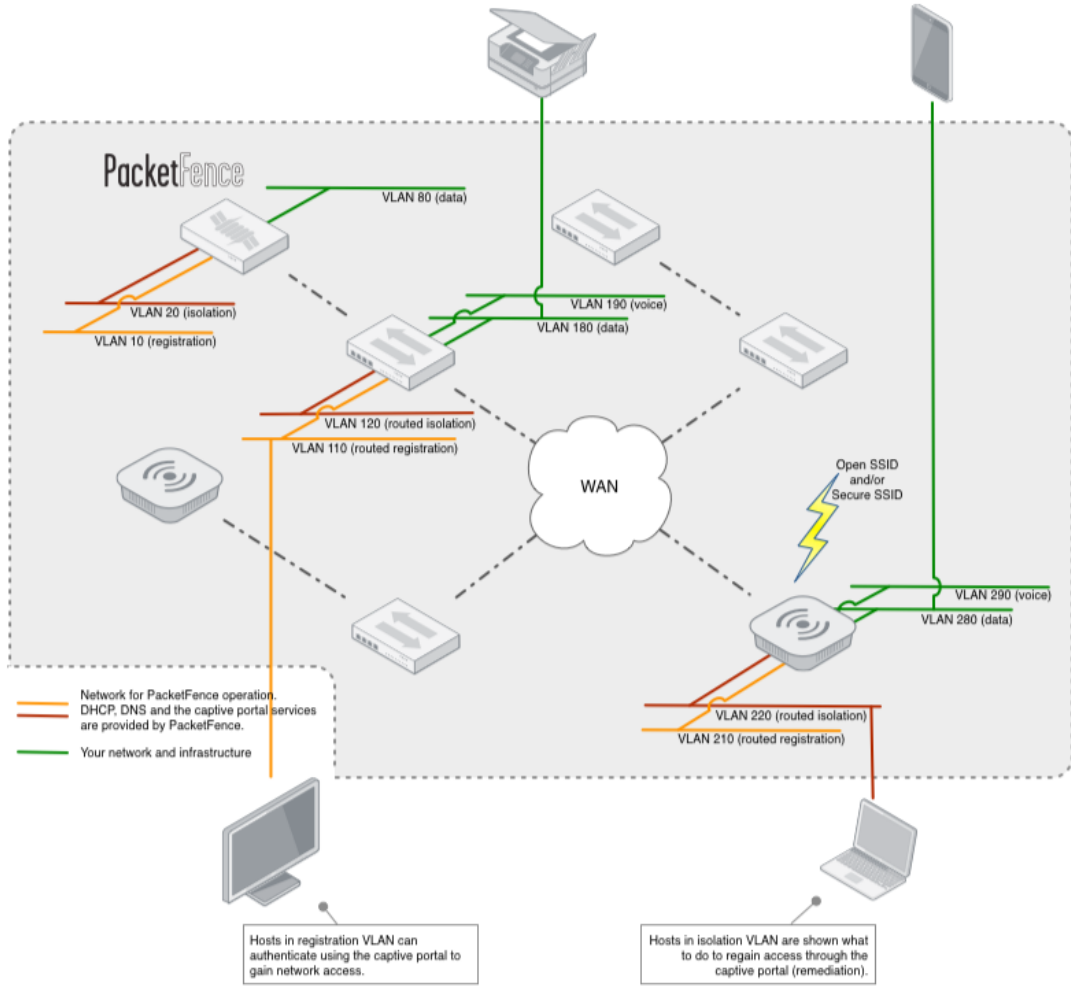


Figure 2.10: PacketFence components (source: [PacketFence](#)).

### Enforcement mode

Packet fence provide both out-of-band and inline mode of deployment. Although out-of-line is preferred because it allows to scale the solution geographically and is more resilient to failures. When used with proper technology out-of-line mode can secure hundreds of switches and many thousands nodes connected to them. On the other hand inline mode is the only solution for unmanageable wired and wireless network. These two modes can also coexist very well together in a network.

Figure 2.11: PacketFence architecture (source: [PacketFence](#)).

## Authentication and registration

PacketFence provide authentication for Captive Portal with a number of mechanism which includes Microsoft Active Directory (AD) [55], OAuth2 Authentication, eduroam, SAML Authentication and External API Authentication. The SAML based authentication can be extended to provide eIDAS authentication.

### 2.7.2 NoDogSplash

NoDogSplash (NDS) [56] is a high performance with small footprint Captive Portal. By default it provides a simple HTML splash page with restricted connectivity to Internet, along with that it incorporates an API which can be integrated with sophisticated authentication applications. NoDogSplash can control data rate on the basis of IP connection by integrating with Smart Queue Management (SQM) configured separately, with NDS being fully compatible.

NDS is composed of two main functions, which allows to integrate different authentication mechanism

- **Capturing** Functionality for capturing clients
- **Authentication** Functionality for authenticating clients.



## Authentication and registration

NDS provide simple web-page to authenticate user using local credentials (email, password). Along with that it also provides the tool to create customise sophisticated authentication mechanism for Captive Portal.

- **Forward Authentication Service (FAS)** FAS provides user validation by forwarding the authenticating process to another application, which can be present on local area network or on the the Internet .
- **PreAuth** It is a special case of FAS where authentication application is served by the NDS. It doesn't requires the full implementation of the FAS and is good for NDS with limited memory resources.
- **BinAuth** It provides authentication using a POST authentication script or by running an extension.

### 2.7.3 Zeroshell

Zeroshell is a Linux based distribution for servers and embedded devices to provide network services [57]. The main features of Zeroshell are load balancing and failover of multiple Internet connections, UMTS/HSDPA connections by using 3G modems, RADIUS server for providing secure authentication and automatic management of encryption keys to wireless networks, Captive Portal to support web login [58]. Description of some of the important features are given below

- Balancing and Failover of multiple Internet connections;
- RADIUS server to provide authentication and automatic management of encryption keys to Wireless networks.
- Captive Portal for web login support on wireless and wired networks. Zeroshell provide native implementation of Captive Portal without using any other application. It allows to authenticate using various mechanism, that includes credential based and Shibboleth based.
- QoS management (Quality of Service) and traffic shaping for traffic control on congested networks.
- Support for Wireless Access Point functionality with Multi SSID.
- Firewall Packet Filter and Stateful Packet Inspection (SPI) with filters applicable both in routing and in bridging on all types of network interfaces.



## Chapter 3

# Design and implementation of WiFi access with eIDAS through Zeroshell

### 3.1 Introduction

Zeroshell is a pre-build Linux based distribution that aims to provide network services like firewall, Dynamic DNS, VPN, RADIUS server, Captive Portal e.t.c. We selected Zeroshell [64] because of it's ability to create a captive portal to authenticate users against an Identity Provider (IdP) using Shibboleth SAML 2.0, a DHCP server for dynamic allocation of IP address and easy management of white-listing for eIDAS-nodes/IdP. Zeroshell provide two interfaces for configuration, a CLI (command line interface) and a web interface. As the name says, all of it's configuration can be done using web interface.

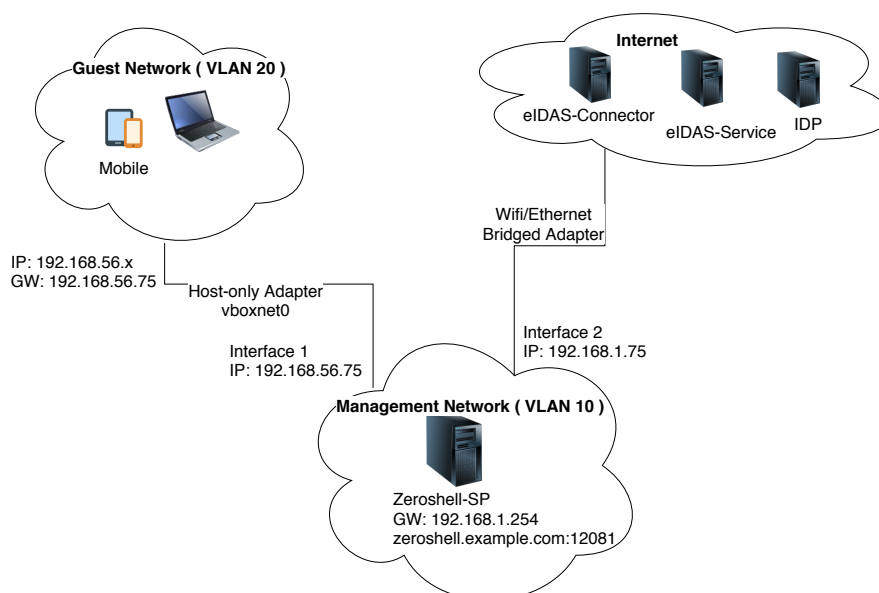


Figure 3.1: Zeroshell Captive Portal infrastructure in test environment

We set up a dedicated machine to act as eIDAS-enabled Service provider. We installed Zeroshell on it using virtual box and modified the configuration of the Shibboleth SP (in Zeroshell) to generate an eIDAS authentication request (instead of a plain SAML request) which is sent to the eIDAS-Connector, and then subsequently through the eIDAS infrastructure until the user reaches

his IdP. User will authenticate itself on IdP and its identity will be sent to Zeroshell Shibboleth SP through the eIDAS infrastructure. Shibboleth-SP will then allow access to Internet.

## 3.2 Authentication flow

The user will start the authentication by connecting to the internal network of the Zeroshell. Zeroshell will assign IP address to the user dynamically using DHCP. The user will try to access http URL, in our case we entered [www.abc.com](http://www.abc.com). Zeroshell will detect that user is not authenticated and redirects to login page of Captive Portal at URL <https://192.168.56.75:12081/cgi-bin/zscp>. It will pass following parameters in the URL.

```
Section: CPAuth
Action: Show
ZSCPRedirect: abc.com::http://abc.com/?
```

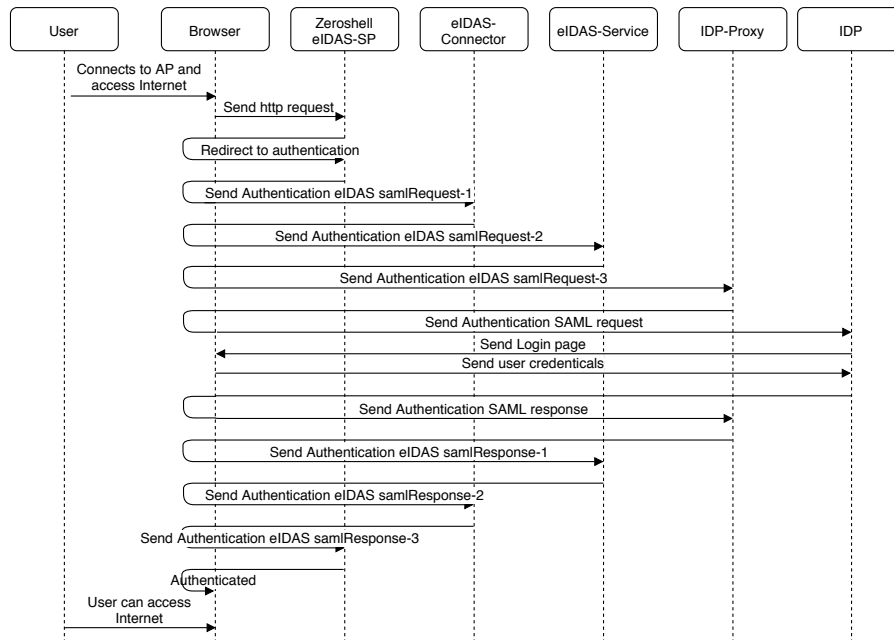


Figure 3.2: Zeroshell authentication flow

It will then show Captive Portal login page, we are using eIDAS framework for authentication for which we clicked AAI (Authentication Authorisation Infrastructure) button. That will create eIDAS (*AuthnRequest*) for eIDAS-Connector and send it at URL <https://connector-test-eid4u.polito.it/EidasNode/ServiceProvider> using user's browser. It will pass following parameters in the request.

```
country: IT
sendmethods: POST
postLocationUrl: https://connector-test-eid4u.polito.it/EidasNode/
ServiceProvider
redirectLocationUrl: https://connector-test-eid4u.polito.it/EidasNode/
ServiceProvider
RelayState: ss:mem:17419df78809c9d7b59631cf6edab9ad
SAMLRequest: <SAMLRequest>
```

eIDAS-connector receives eIDAS (*AuthnRequest*) and verifies with the metadata and certificate present locally in eIDAS-Connector for Zeroshell eIDAS-SP. After successfully verification of the request, it will create a eIDAS (*AuthnRequest*) for eIDAS-Service depending on the country parameter. In this case it will create a eIDAS (*AuthnRequest*) for Italian eIDAS-Service

at URL <https://service-test-eid4u.polito.it/EidasNode/ColleagueRequest> and send it using user's browser. It will pass following parameters in the request.

```
RelayState: ss:mem:a1bf8bbf23f2a688b8e57e6372bbdc3f
token: 2sjh1vPMkUkTZCM-mEGUIUA15k$
SAMLRequest: <SAMLRequest>
```

eIDAS-Service receives eIDAS (*AuthnRequest*) and verifies with the metadata and certificates of eIDAS-Connector. After successfully verification of the request, it will create a eIDAS (*AuthnRequest*) for IdP-Proxy because IdP-proxy is used in Italian hierarchy to create a bridge between eIDAS-Service and IdP. It exchanges messages with eIDAS-Service in eIDAS message format and with IdP in SPID message format. eIDAS-Service will send eIDAS (*AuthnRequest*) to IdP-Proxy at URL <https://idp-proxy-test-eid4u.polito.it/idpproxy/idpeurequest> using user's browser. It will pass following parameters in the request.

```
messageFormat: eidas
SAMLRequest: <SAMLRequest>
```

IdP-Proxy receives eIDAS (*AuthnRequest*) and verifies with the metadata and certificates of the eIDAS-Service. After successful verification of the request, it ask for the selection of IdP provider. We are using *InfoCert* IdP for the test and clicked on *InfoCert* button. IdP-Proxy will create a SPID (*AuthnRequest*) for *InfoCert* IdP and send it at URL <https://identitycl.infocert.it/spid/samlssso> using user's browser. It will pass following parameters in the request.

```
RelayState: SPID_REQUEST_RELAYSTATE
SAMLRequest: <SAMLRequest>
```

*InfoCert* IdP receives SPID (*AuthnRequest*) and verifies with the metadata and certificates of the IdP-Proxy. After successful verification of the request, it will ask for user credentials using HTML page at URL <https://identitycl.infocert.it/spid/basicauth.page>. We provided test credentials and clicked on “Entra con SPID” button. After verifying user's credentials, it will create a SPID (*Response*) for IdP-Proxy at URL <https://idp-proxy-test-eid4u.polito.it/idpproxy/spidresponse> and send it using user's browser. It will pass following parameters in the response.

```
RelayState: SPID_REQUEST_RELAYSTATE
SAMLResponse: <SAMLResponse>
```

IdP-Proxy receives SPID (*Response*) and verifies it with IdP metadata. Next it gets attributes element from the (*Response*). It will then create an eIDAS (*Response*) for eIDAS-Service and send it at URL <https://service-test-eid4u.polito.it/EidasNode/IdpResponse> using user's browser. It will pass following parameters in the response.

```
username: username
SAMLResponse: <SAMLResponse>
```

eIDAS-Service receives eIDAS (*Response*) and verifies it with IdP-Proxy metadata. Next it decipher the session key using its “encryption” private-key and decipher attributes element using the session key. It will then create an eIDAS (*Response*) for eIDAS-Connector at URL <https://connector-test-eid4u.polito.it/EidasNode/ColleagueResponse> and send it using user's browser. It will pass following parameters in the response.

```
RelayState: MyRelayState
SAMLResponse: <SAMLResponse>
```

eIDAS-Connector receives eIDAS (*Response*) and verifies it with eIDAS-Service metadata. Next it decipher the session key using its “encryption” private-key and decipher attributes element using the session key. It will then create an eIDAS (*Response*) for Zeroshell eIDAS-SP at URL <https://zeroshell.example.com:12081/Shibboleth.sso/SAML2/POST> and send it using user's browser. It will pass following parameters in the response.

```
RelayState: ss:mem:99e78aaa4f6d4e7ebc79bd47072c5d39
SAMLResponse: <SAMLResponse>
```

Zeroshell eIDAS-SP receives eIDAS (*Response*) and verifies it with eIDAS-Connector metadata. Next it deciphers the session key using its “encryption” private-key and deciphers attributes element using the session key. It will then authenticate the user and redirect to URL <http://www.abc.com>. It will also create another window to manage the session of the user at <https://zeroshell.example.com:12081/cgi-bin/zscp>.

### 3.3 Zeroshell setup

First we installed and configured Zeroshell on virtual box. Then we used CLI for network configuration of Zeroshell web interface. Once we are able to access web interface, we created a profile, which will enable us to save out work. Then we created two network interfaces, one for internal network of Captive Portal and other one to connect to the Internet. After that we enabled captive portal and changed Shibboleth SP to provide authentication using eIDAS framework. Finally we will discuss the problems encountered while connecting Zeroshell with eIDAS infrastructure.

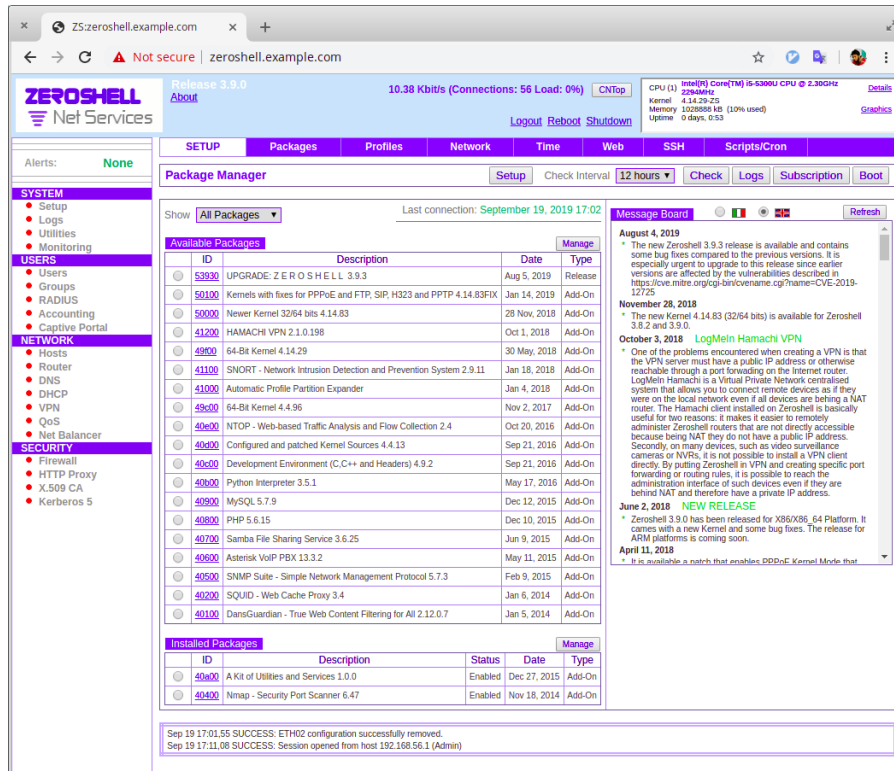


Figure 3.3: Zeroshell web interface

#### 3.3.1 Virtual Box initialisation

For this set-up we used a Ubuntu machine with Zeroshell installed on a Virtual Box. The version we use for Zeroshell is 3.9.0. First step was to download and install the virtual box. Then we configure virtual box and installed Zeroshell. We created a new Virtual Machine for Zeroshell. We provided 2048 MB of RAM and created a new hard drive of type VDI. We choose dynamic allocation and provided 10 GB of space. We provided the Zeroshell ISO image downloaded from the official Zeroshell website at URL <https://zeroshell.org/download/>. Then we created two interfaces one Host-only Adapter for guest clients to connect to internal network and the other

one Bridged Adapter to connect to the Internet. We created the Host-only Adapter “vboxnet0” with network 192.168.56.0/24. After that we started the VM and it takes to the CLI. The CLI interface is shown in Figure 3.5.

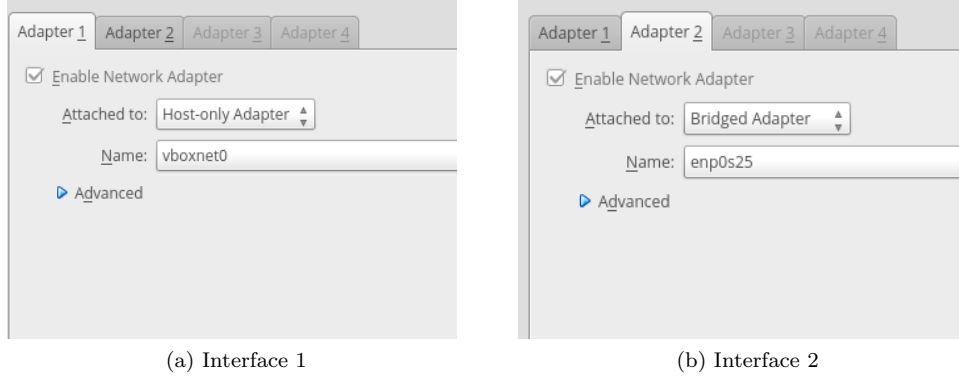


Figure 3.4: Network interfaces of Zeroshell

### 3.3.2 Accessing web interface

Command line interface is required to configure IP for the web interface and some other basic parameters. The Zeroshell interface is set by default to 192.168.0.0/24 network, more precisely with IP address 192.168.0.75. To access the network we either set a static IP address within the 192.168.0.0/24 network or we can change the IP of Zeroshell using the CLI. As we have two interfaces on Zeroshell, we decided to use the Host-only Adapter “vboxnet0” network to access the web interface. We change the IP of the eth0 of Zeroshell from the CLI to 192.168.56.75. After that we are able to access the web interface from browser at <https://192.168.56.75/>. We provided the Zeroshell credentials to access the configuration which is by default as username: admin and password: zeroshell.

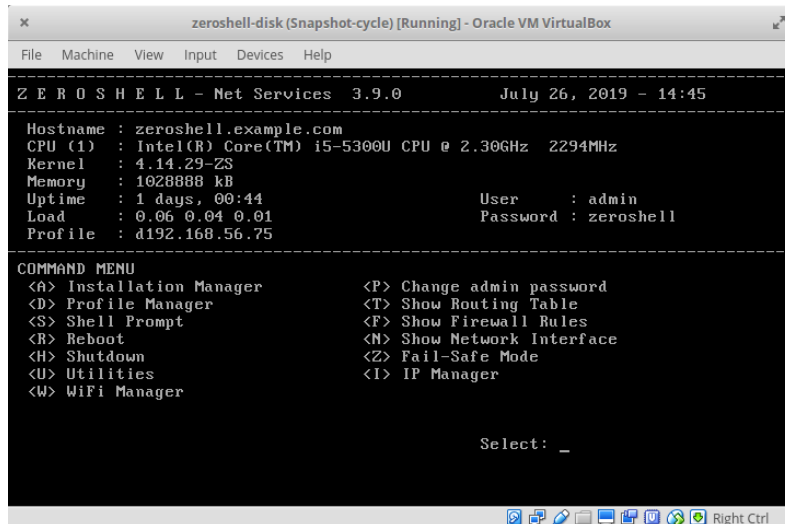


Figure 3.5: Zeroshell command line interface

### 3.3.3 Profile creation

Now we can create profile and configure captive portal using web interface. Profile is created using Profiles tab, which is accessible from the web interface at

## Setups &gt; Profiles

By selecting the hard disk you want to save the profile, it will show the profile creation button. It is shown in Figure 3.6, after clicking the button Zeroshell opens a new window for configuration of the profile Figure 3.7. We used default parameters for most of the configuration. We select our internal network interface in the Ethernet Interface and given IP address *192.168.56.75*. After that we have to configure network configuration in the profile. After creating the profile, select

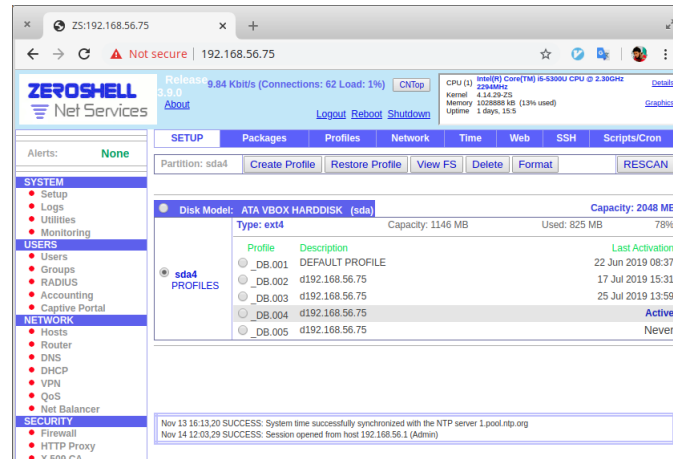


Figure 3.6: Zeroshell profile creation

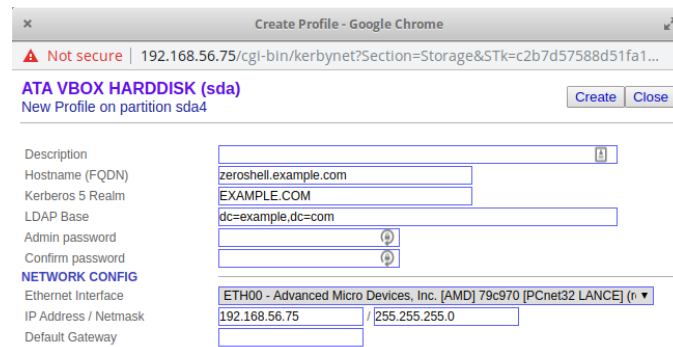


Figure 3.7: Zeroshell profile creation parameters

the profile and click Activate button to activate the profile. This will reboot the Zeroshell and requires to login again using the credentials.

### 3.3.4 Network configuration

Network is configured from the web interface and is accessible at

## Setups &gt; Network

We have two interfaces, one for the internal network and another for the Internet. On ETH00 we have our ip 192.168.56.75 on vboxnet0 and on ETH01 we add a IP on the Ethernet or WiFi network. In our case we are using 192.168.1.75 to get to the Internet. It is shown in Figure 3.8. Then we set the gateway to router IP which in our case is 192.168.1.1. We also enabled DHCP on ETH00 interface to provide dynamic IP address to the guest users.

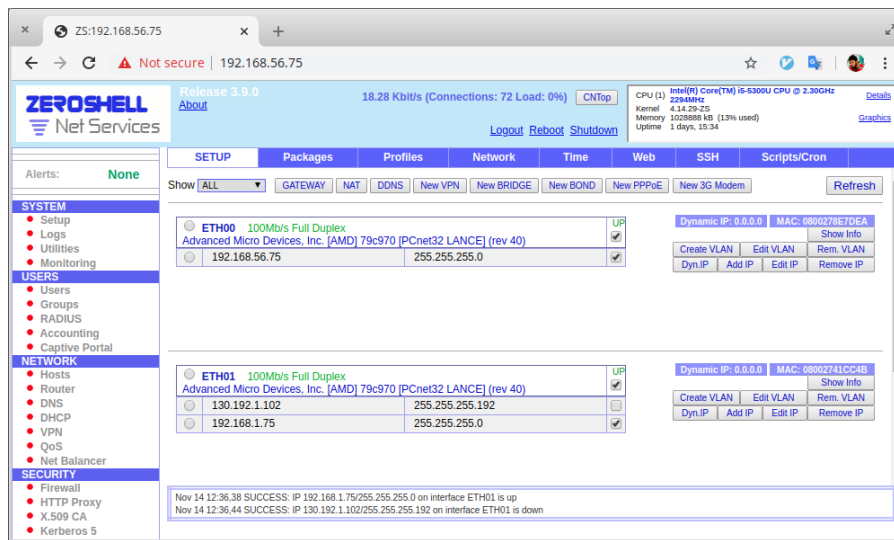


Figure 3.8: Zeroshell network configuration

### 3.3.5 Captive Portal

Zeroshell allows to create a Captive Portal through the use of default gateway that captures request from guest clients. We have interface (ETH00) as internal network to which we want to enable Captive Portal. So from **Captive Portal** tab we selected the interface (ETH00) from the drop down and then checked the “GW” check-box to enable Captive Portal. Now At this stage Captive Portal is working on ETH00 which allows to authenticate using username/password.

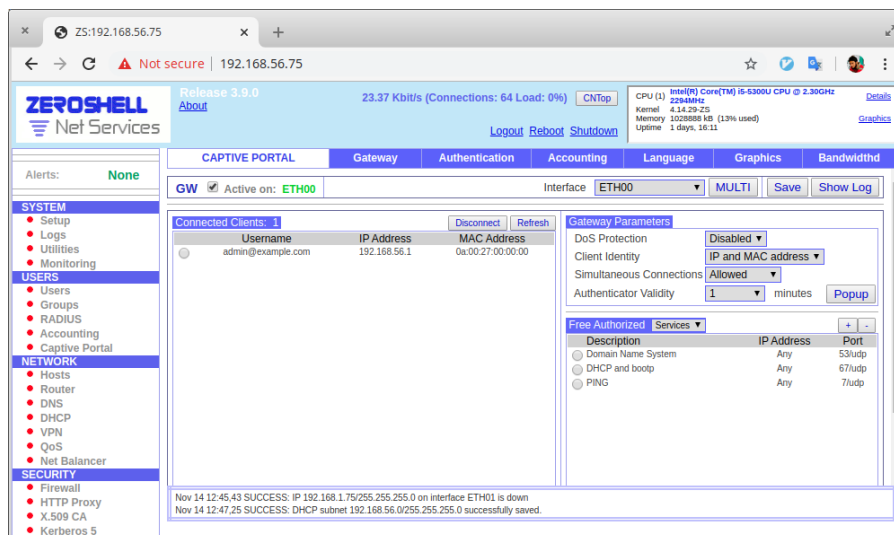


Figure 3.9: Enabling Captive Portal on Zeroshell

### 3.3.6 Shibboleth authentication

Zeroshell out of the box provide the ability to use shibboleth authentication for captive Portal. It is enabled by going to Authentication and change the Shibboleth Authentication drop-down to Enabled as shown in Figure 3.10. Now it will allow to authenticate using both username/password and Shibboleth SAML 2.0. It is accessible at

Captive Portal > Authentication > Shibboleth Authentication

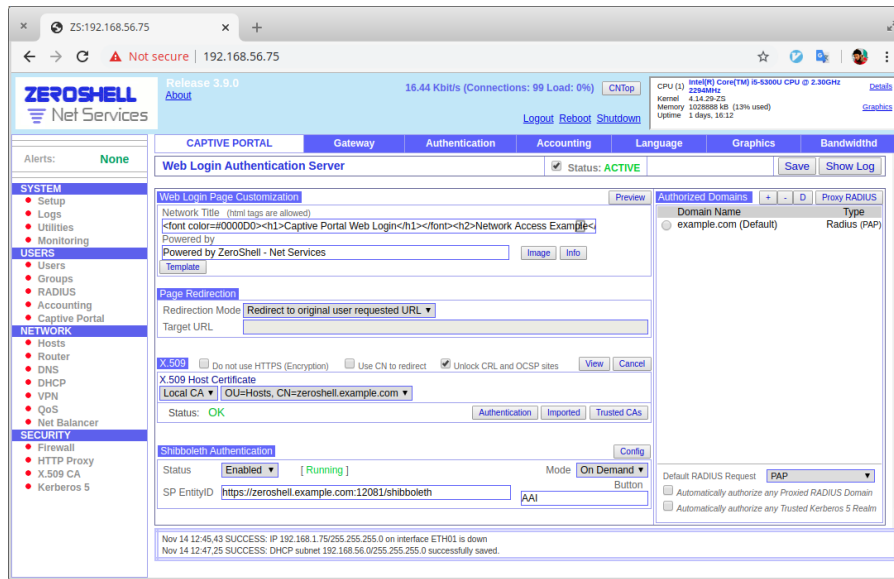


Figure 3.10: Enabling Captive Portal on Zeroshell

### 3.3.7 Shibboleth configuration files

Zeroshell provides a basic “Web File Editor” for Shibboleth configuration. Which is accessible at

Captive Portal > Authentication > Shibboleth Authentication > Config

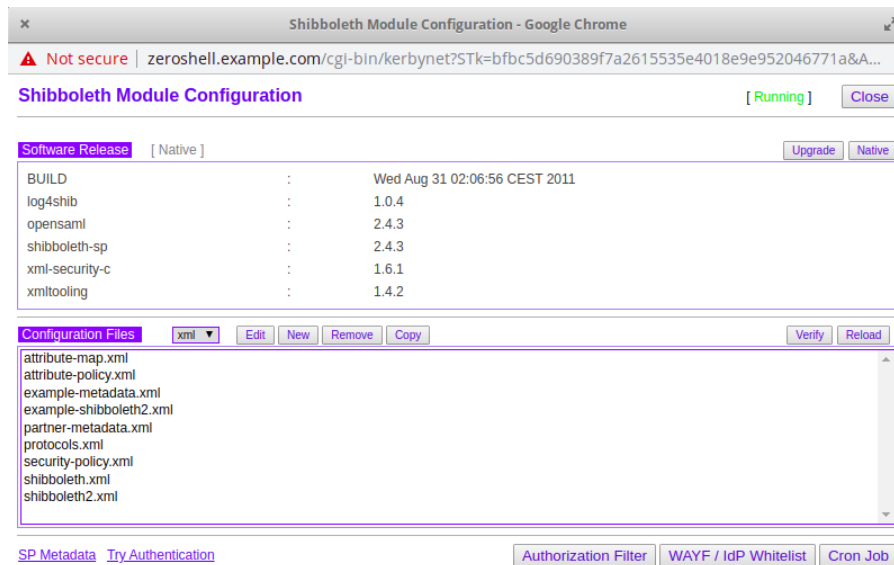


Figure 3.11: Configuration of Shibboleth files

## 3.4 Configuring Shibboleth authentication with eIDAS

To enable Captive Portal to authenticate with eIDAS framework, it should send an eIDAS *AuthnRequest* to eIDAS-Connector. Changes to send an eIDAS *AuthnRequest* are discussed in this section.



### 3.4.1 EntityID of the SP

We need to specify the entityID in the Host element to zeroshell.example.com for metadata.

```
<Host name="zeroshell.example.com" port="12081" scheme="https">
  <Path name="secure" authType="shibboleth" requireSession="true"/>
</Host>
```

### 3.4.2 ApplicationDefaults element

In the ApplicationDefaults, entityID is set to “https://zeroshell.example.com:12081/shibboleth”. Remote\_User parameter is used to identify the user after authentication. In our case we are using “FirstName PersonIdentifier DateOfBirth” as Remote\_User. We also enabled signing and encryption for XML messages.

```
<ApplicationDefaults entityID="https://zeroshell.example.com:12081/shibboleth"
  REMOTE_USER="FirstName PersonIdentifier DateOfBirth"
  signing="true" encryption="true">
```

### 3.4.3 MetadataProvider

eIDAS SAML communication requires agreement between system entities regarding identifiers, binding support and endpoints, certificates and keys, and so forth. A metadata specification is useful for describing this information in a standardised way. To communicate with eIDAS-Connector, Zeroshell needs to get its metadata/certificate and vice-versa. We provided eIDAS-Connector’s metadata and certificate locally in MetadataProvider.

```
<MetadataProvider type="XML" file="partner-metadata.xml">
  <MetadataFilter type="Signature" certificate="fedsigner.pem" />
</MetadataProvider>
```

### 3.4.4 Cryptography certificates

Cryptography certificates are used for confidentiality/integrity of the messages and verification of the endpoints. Three certificates are used

- **SAML Metadata signature** It is used for verifying the Metadata.
- **SAML signature** It is used for integrity/authenticity of the SAML message.
- **XML encryption** It is used for encrypting the attributes information in the eIDAS response.

Two certificates are used for signing and one for encryption. These configuration are provided in openssl.cnf configuration file. The content of the file is as following

```
[ eidas_sign ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature,nonRepudiation

[ eidas_enc ]
basicConstraints=critical,CA:FALSE
keyUsage=keyEncipherment
```

These certificates are generated using a shell script `create_cert.sh`. The contents of the file is provided below.

```

DIR="$(pwd)"
service="zeroshell"

SSLCONF="${DIR}/openssl.cnf"
_KEYLEN=${KEY_LEN:-4096}
_KEYTYPE=${KEY_TYPE:-"rsa"}
_DIGEST=${DIGEST:-"sha512"}
_BASE_NAME=${BASE_NAME:-"zeroshell"}
_SIGN="-saml-signature"
_ENC="-saml-encryption"
_METASIGN="-metadata-signature"

_C=${C:-"IT"}
_O=${O:-"None"}
_OU=${OU:-"None"}
_CN_PREFIX=${CN_PREFIX:-""}
#Sign
NAME=${_BASE_NAME}${_SIGN}
EXT="eidas_sign"
openssl req -new -nodes -config ${SSLCONF} \
    -newkey ${_KEYTYPE}:${_KEYLEN} -passout file:${NAME}.key.pass \
    -keyout ${NAME}.key -x509 -${_DIGEST} -days 3650 -out ${NAME}.pem \
    -subj "/C=${_C}/O=${_O}/OU=${_OU}/CN=${_CN_PREFIX}SAML Signature" \
    -extensions ${EXT}
#Metadata
NAME=${_BASE_NAME}${_METASIGN}
EXT="eidas_sign"
openssl req -new -nodes -config ${SSLCONF} \
    -newkey ${_KEYTYPE}:${_KEYLEN} -passout file:${NAME}.key.pass \
    -keyout ${NAME}.key -x509 -${_DIGEST} -days 3650 -out ${NAME}.pem \
    -subj "/C=${_C}/O=${_O}/OU=${_OU}/CN=${_CN_PREFIX}Metadata Signature" \
    -extensions ${EXT}
# ENC
NAME=${_BASE_NAME}${_ENC}
EXT="eidas_enc"
openssl req -new -nodes -config ${SSLCONF} \
    -newkey ${_KEYTYPE}:${_KEYLEN} -passout file:${NAME}.key.pass \
    -keyout ${NAME}.key -x509 -${_DIGEST} -days 3650 -out ${NAME}.pem \
    -subj "/C=${_C}/O=${_O}/OU=${_OU}/CN=${_CN_PREFIX}SAML Encryption" \
    -extensions ${EXT}

```

Cryptography certificates for signature and encryption of the eIDAS messages are added in *CredentialResolver* element in *shibboleth2.xml*. We have two *CredentialResolver* elements, one for encrypting the request and second for signing the request.

```

<CredentialResolver type="File" key="zeroshell-saml-signature.key"
certificate="zeroshell-saml-signature.pem" use="signing"/>
<CredentialResolver type="File" key="zeroshell-saml-encryption.key"
certificate="zeroshell-saml-encryption.pem" use="encryption"/>

```

### 3.4.5 Node-Country selection

The Node-Country provides the user's country of origin. The eIDAS-Connector is responsible for connecting to the respective eIDAS-Service depending on this. The Node country of the user is send as parameter to the post request as in the case of Italian scenario it is send as country: IT.

### 3.4.6 SessionInitiator AuthnRequest element

We are using AuthnRequest element, this is used as a template for the request issued. It is useful for supplying advanced request content that cannot be configured in a simpler way.

```
<saml2p:AuthnRequest
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:oidas="http://oidas.europa.eu/saml-extensions"
  xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" ForceAuthn="true"
  IsPassive="false" Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
  Destination="https://connector-test-oid4u.polito.it/EidasNode/ServiceProvider"
  ProviderName="zeroshell-SP"
  ID="foo" Version="2.0" IssueInstant="2012-01-01T00:00:00Z">
  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://zeroshell.example.com:12081/shibboleth</saml2:Issuer>
  <saml2p:Extensions>
    <oidas:SPTYPE>public</oidas:SPTYPE>
    <oidas:RequestedAttributes>
      <oidas:RequestedAttribute FriendlyName="FamilyName"
        Name="http://oidas.europa.eu/attributes/naturalperson/CurrentFamilyName"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
      <oidas:RequestedAttribute FriendlyName="FirstName"
        Name="http://oidas.europa.eu/attributes/naturalperson/CurrentGivenName"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
      <oidas:RequestedAttribute FriendlyName="DateOfBirth"
        Name="http://oidas.europa.eu/attributes/naturalperson/DateOfBirth"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
      <oidas:RequestedAttribute FriendlyName="PersonIdentifier"
        Name="http://oidas.europa.eu/attributes/naturalperson/PersonIdentifier"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
    </oidas:RequestedAttributes>
  </saml2p:Extensions>
  <saml2p:NameIDPolicy AllowCreate="true"
    Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"/>
  <saml2p:RequestedAuthnContext Comparison="minimum">
    <saml2:AuthnContextClassRef>http://oidas.europa.eu/LoA/low
  </saml2:AuthnContextClassRef>
  </saml2p:RequestedAuthnContext>
</saml2p:AuthnRequest>
```

### 3.4.7 Attributes configuration

Configuration of attributes requested after authentication are provided in attribute-map.xml file. We are using “eIDAS minimum dataset”, that includes FamilyName, FirstName, DateOfBirth and PersonIdentifier. The configuration for attribute-map file are as following

```
<Attributes xmlns="urn:mace:shibboleth:2.0:attribute-map"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Attribute name="http://oidas.europa.eu/attributes/naturalperson/CurrentFamilyName"
    id="FamilyName" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" />
  <Attribute name="http://oidas.europa.eu/attributes/naturalperson/CurrentGivenName"
    id="FirstName" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" />
  <Attribute name="http://oidas.europa.eu/attributes/naturalperson/DateOfBirth"
    id="DateOfBirth" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" />
  <Attribute name="http://oidas.europa.eu/attributes/naturalperson/PersonIdentifier"
    id="PersonIdentifier" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" />
</Attributes>
```

### 3.4.8 White-listing

In general, since the user has not been authenticated yet, the captive portal deny all access to Internet, But we need to configure the captive to allow access to eIDAS-nodes and IdP in

Internet, while all the other Internet traffic would not be allowed (until the user is successfully authenticated). This problem is identified previously in the paper [65], to solve it we add them manually in the “whitelist”, which is shown in Figure 3.12. **Whitelist** is accessible at

Captive Portal > Authentication > Shibboleth Authentication > Config > WAYF/IDP Whitelist



Figure 3.12: Captive Portal White-listing for eIDAS-nodes/IDP

### 3.4.9 Problems encountered

There were several problems faced while developing Shibboleth SP in Zeroshell.

1. Incompatibilities between the cryptography algorithms supported by the Zeroshell (AES256-CBC) and the security requirements of the eIDAS node (requires AES 128-GCM, AES192-GCM, AES256-GCM) as described in the session 2.5.3.
2. White-listing (auto-discovery). There is no automatic discovery of all the actors (IdPs, eIDAS Node’s components, other element such as the IdP Proxy) that need to be reached by the user’s browser and thus they have to be white-listed on the Zeroshell SP. It is required to manually add the location (names) of these elements in the Zeroshell SP.
3. Citizen country selection. By default, Zeroshell does not allow to select a citizen country where the user will be authenticated. When enabled with eIDAS, it is required to make modification on Zeroshell SP side to allow to select the country in which the citizen will be authenticated.
4. Rendering public the Zeroshell SP’s metadata. In the default installation, Zeroshell SP’s metadata is statically configured on the IdP. When enabled with eIDAS, the eIDAS Connector requires a public URL from where the Zeroshell SP’s metadata would be downloaded. However, since Zeroshell does not expose as a public URL its metadata this would again require a modification in Zeroshell’ source code.

## 3.5 Authentication cycle

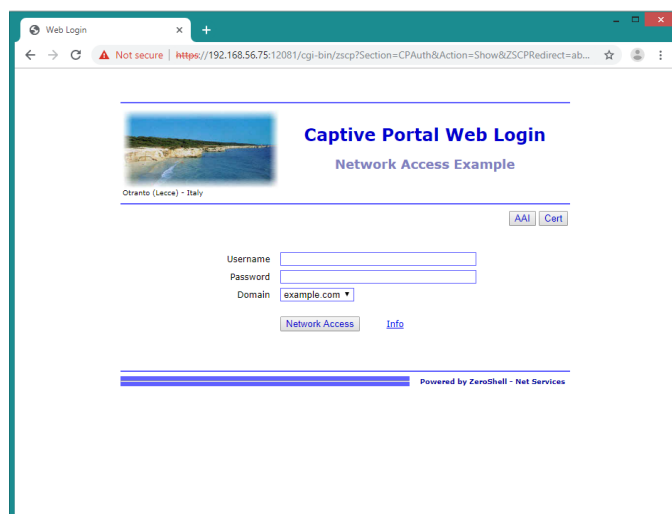


Figure 3.13: Zeroshell Captive Portal authentication Step-1

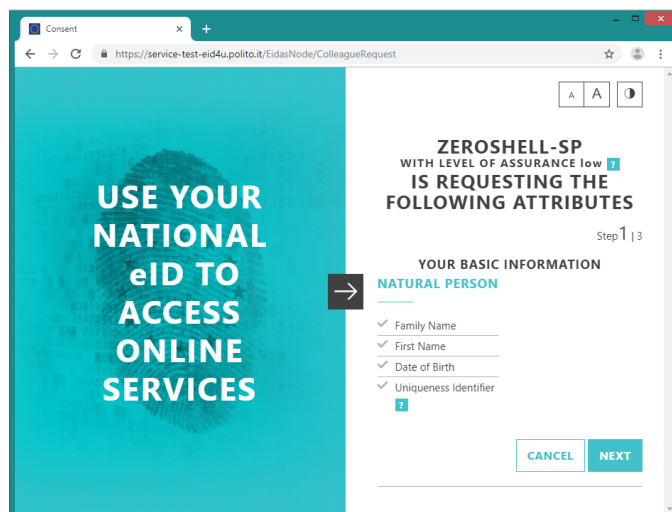


Figure 3.14: Zeroshell Captive Portal authentication Step-2

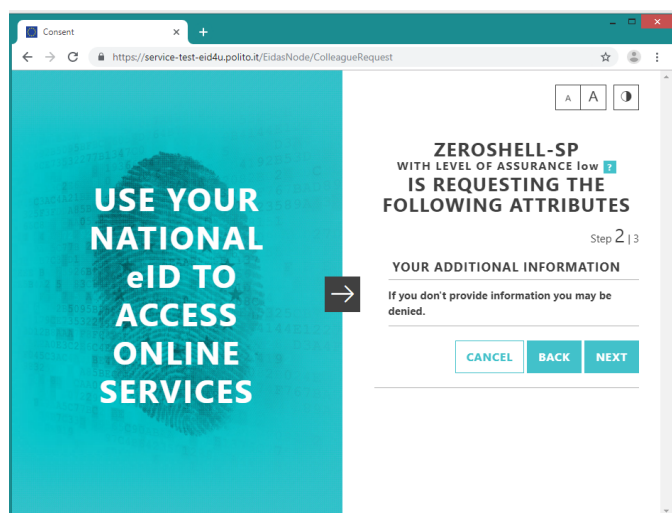


Figure 3.15: Zeroshell Captive Portal authentication Step-3

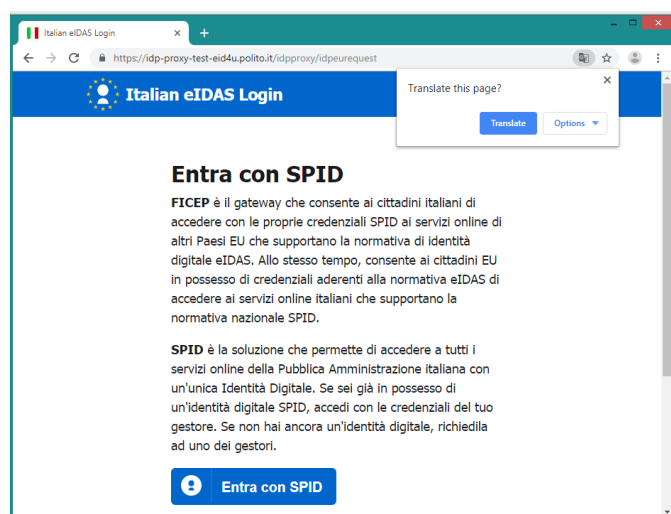


Figure 3.16: Zeroshell Captive Portal authentication Step-4

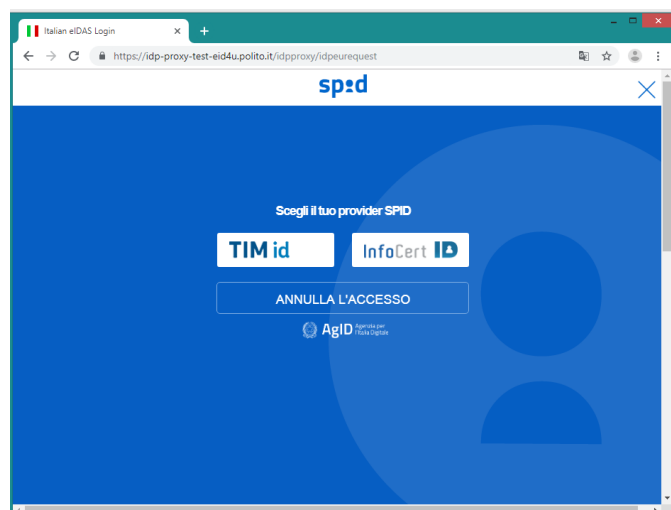


Figure 3.17: Zeroshell Captive Portal authentication Step-5

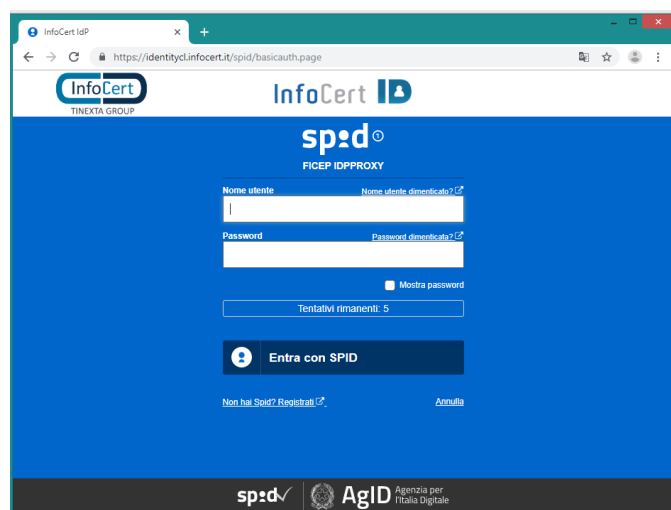


Figure 3.18: Zeroshell Captive Portal authentication Step-6

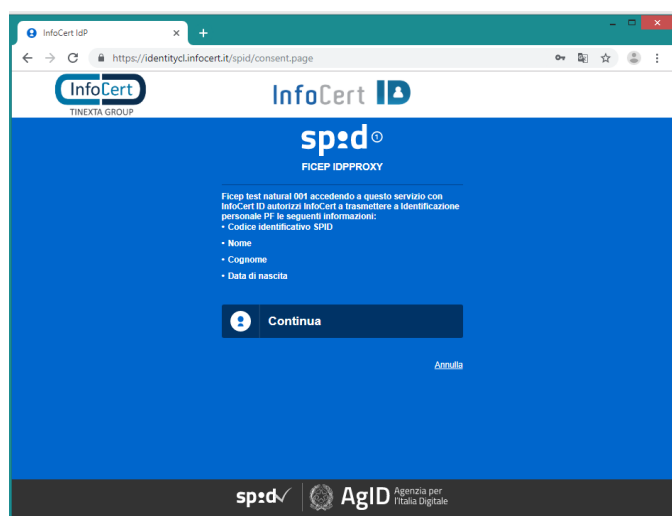


Figure 3.19: Zeroshell Captive Portal authentication Step-7

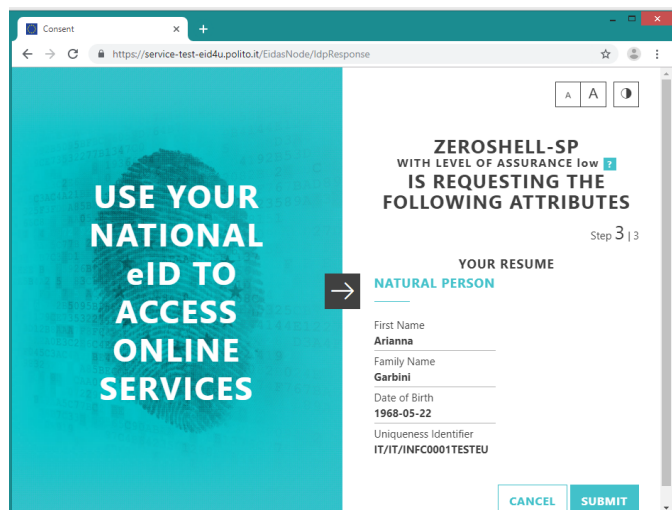


Figure 3.20: Zeroshell Captive Portal authentication Step-8

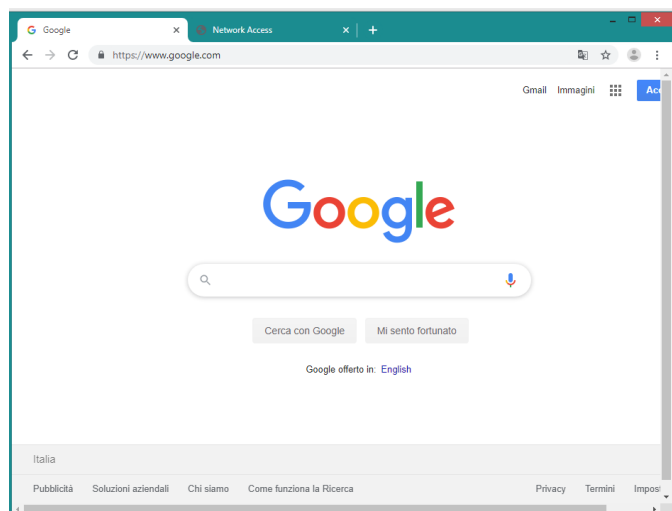


Figure 3.21: Zeroshell Captive Portal authentication Step-9

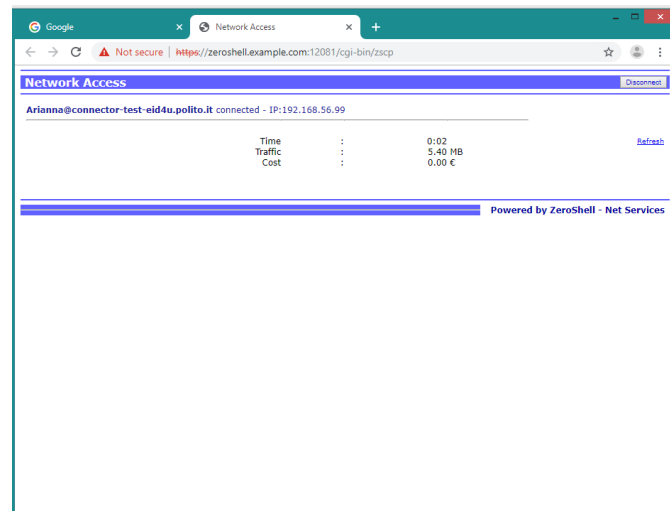


Figure 3.22: Zeroshell Captive Portal authentication Step-10



## Chapter 4

# Design of WiFi access with eIDAS through WiFi-Auth eIDAS-SP and Polito wireless infrastructure

### 4.1 Introduction

This chapter focuses on the hardware based solution for the development of a captive portal to authenticate guest users. This solution exploits the WiFi infrastructure of Politecnico di Torino and extends it using eIDAS code for federated authentication. For that we developed an ad hoc application Wifi-Auth eIDAS-SP to authenticate using eIDAS framework. Next we configured hardware infrastructure consists of Cisco-WLC, Cisco AP 3700, HP Switch and Fortigate-60D to provide Captive Portal functionality and use Wifi-Auth for authentication as shown in Figure 4.1. This is the same infrastructure deployed in Politecnico di Torino, which makes it extremely easy to deploy in production.

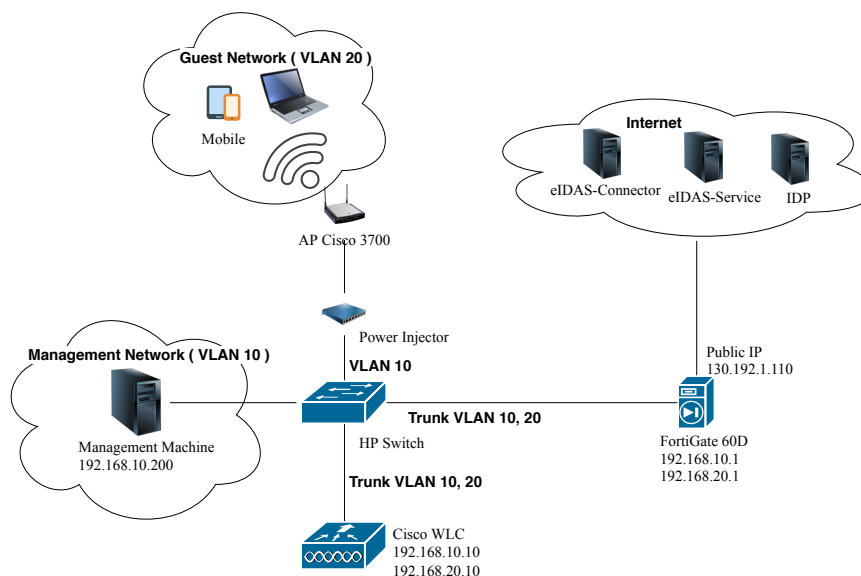


Figure 4.1: WiFi access test-bed setup with Polito wireless infrastructure

## 4.2 Authentication process

In a typical scenario a user from a device will connect to the AP using WiFi. In the modern browser it will automatically detect Captive Portal is present on the network and makes a http request to get to the authentication page. The user can also manually make a http request and it will be redirected by the WLC to the authentication page. The WLC gets the request and checks users authentication. If user is not authenticated, it redirects to Wifi-Auth eIDAS-SP for authentication. Wifi-Auth eIDAS-SP used eIDAS framework for authentication using national credentials. In case of Italian citizen authentication will be done using SPID credentials. A generic authentication flow is shown in Figure 4.3. After authentication is completed, Wifi-Auth will create a guest user on WLC. The guest credentials are send to the browser and submitted automatically on WLC using user's browser. WLC verifies the credentials and authenticate the user to access the internet.

### 4.2.1 Wifi-Auth eIDAS-SP set-up

Our set-up includes a Wifi-Auth eIDAS-SP and Polito wireless infrastructure. Polito wireless infrastructure consists of Cisco-WLC, Cisco-AP 3500 and Fortigate-60D. Wifi-Auth eIDAS-SP provides authentication through eIDAS infrastructure. Cisco-WLC is used for managing authenticated users, Captive Portal and separating network traffic. Fortigate-60D is used for firewall policies and protecting the internal network. As shown in the Figure 4.2 we dedicated a separate machine for the deployment of Wifi-Auth eIDAS-SP in the management network and use another machine in the management network for configuring eIDAS-SP.

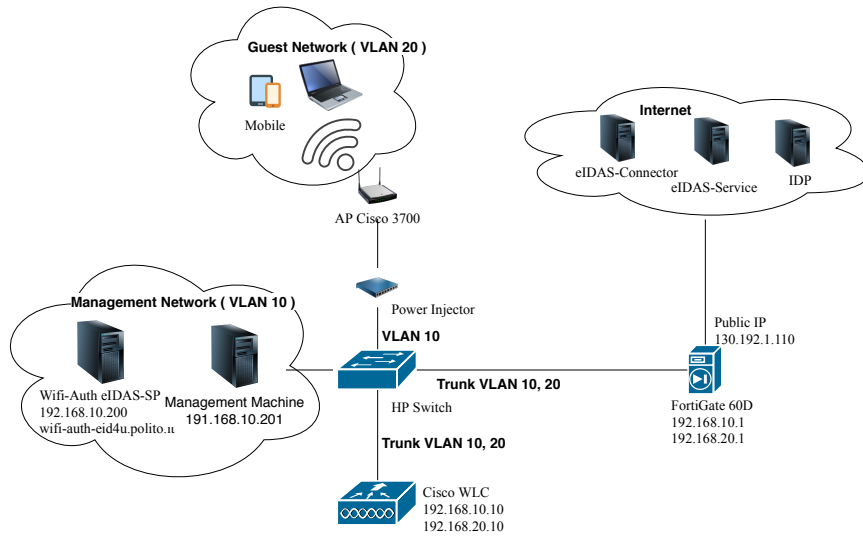


Figure 4.2: WiFi access test-bed setup with Wifi-Auth eIDAS-SP and Polito wireless infrastructure

## 4.3 Authentication flow

This section describes an overview of the authentication process. It is shown in Figure 4.3. The process is composed of a request part and a response part. In a generic scenario the work-flow will be like this.

The request part starts with the guest user connecting to the public “eIDAS” AP using Wi-Fi. Next the user tries to access the internet. WLC detects that user is not authenticated and redirects to Wifi-Auth eIDAS-SP for authentication. eIDAS-SP asks for user's country and creates a eIDAS (*AuthnRequest*) to get minimum data set. It then sends the request to the eIDAS-Connector

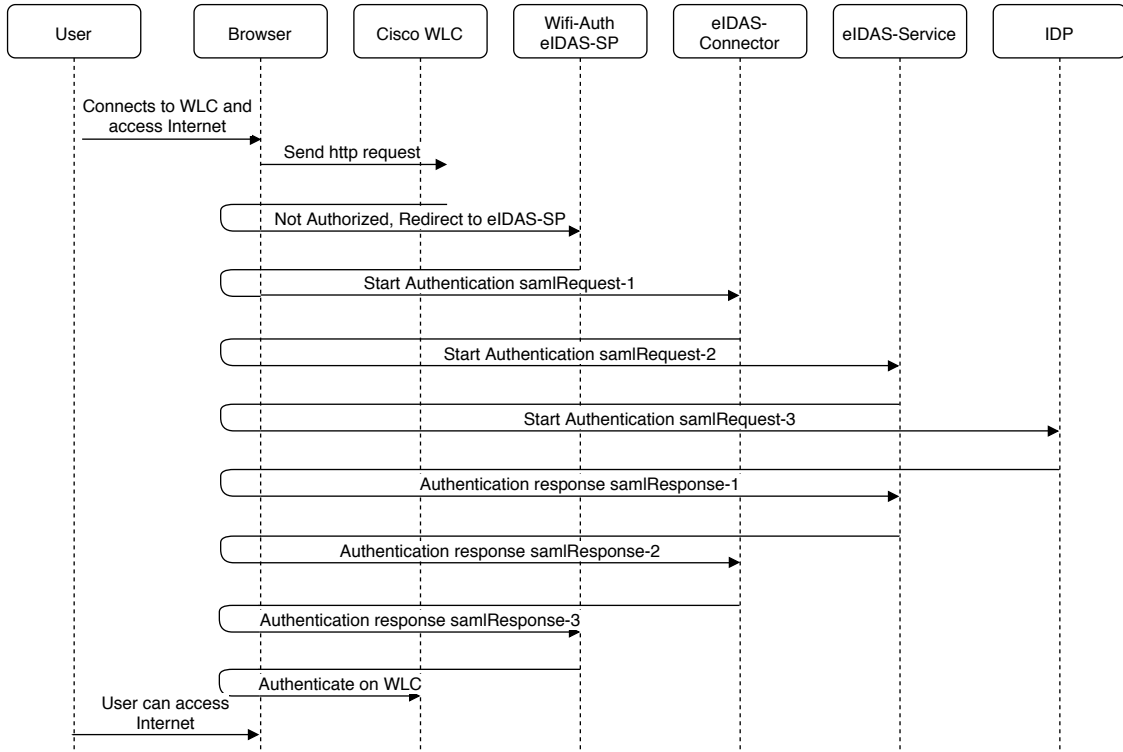


Figure 4.3: Overview of authentication flow for generic scenario

using browser. eIDAS-Connector validates eIDAS-SP, verifies eIDAS (*AuthnRequest*), get user's country and other parameters from eIDAS (*AuthnRequest*). It then creates an eIDAS (*AuthnRequest*) for respective eIDAS-Service and sends using user's browser. eIDAS-Service validates eIDAS-Connector, verifies eIDAS (*AuthnRequest*), get parameters from eIDAS (*AuthnRequest*) and creates an eIDAS (*AuthnRequest*) for IdP and sends using user's browser. IdP validates eIDAS-Service, verifies eIDAS (*AuthnRequest*), get parameters from eIDAS (*AuthnRequest*). It then asks for user credentials and verifies user.

After successful verification IdP sends an eIDAS (*Response*) to eIDAS-Service with user's minimum data set. eIDAS-Service validates IdP, verifies eIDAS (*Response*), decipher eIDAS attributes and creates an eIDAS (*Response*) and sends to the eIDAS-Connector using browser. eIDAS-Connector validates eIDAS-Service, verifies eIDAS (*Response*), decipher eIDAS attributes and creates an eIDAS (*Response*) and sends to the Wifi-Auth eIDAS-SP using browser. eIDAS-SP validates eIDAS-Connector, verifies eIDAS (*Response*), decipher eIDAS attributes and gets user data. It then connects to WLC using SSH and creates a guest user with user's person identifier. It then sends guest user credentials to WLC using a form from user's browser. WLC checks the request and authenticate the user. After authentication user can successfully use the internet.

#### 4.3.1 Authentication flow Italian scenario

In Italian scenario we have a specific case where we have another node IdP-Proxy between eIDAS-Service and IdP. Instead of eIDAS-Service talking directly to IdP, it exchanges messages to IdP-Proxy using eIDAS protocol and IdP-proxy exchange messages to IdP using SPID protocol. Request part of authentication flow for Italian scenario looks like this

1. Guest user starts the authentication process by connecting to the public eIDAS AP using WiFi. Then the user tries to access the internet.
2. WLC detects that user is not authenticated and redirects to Wifi-Auth eIDAS-SP for authentication.

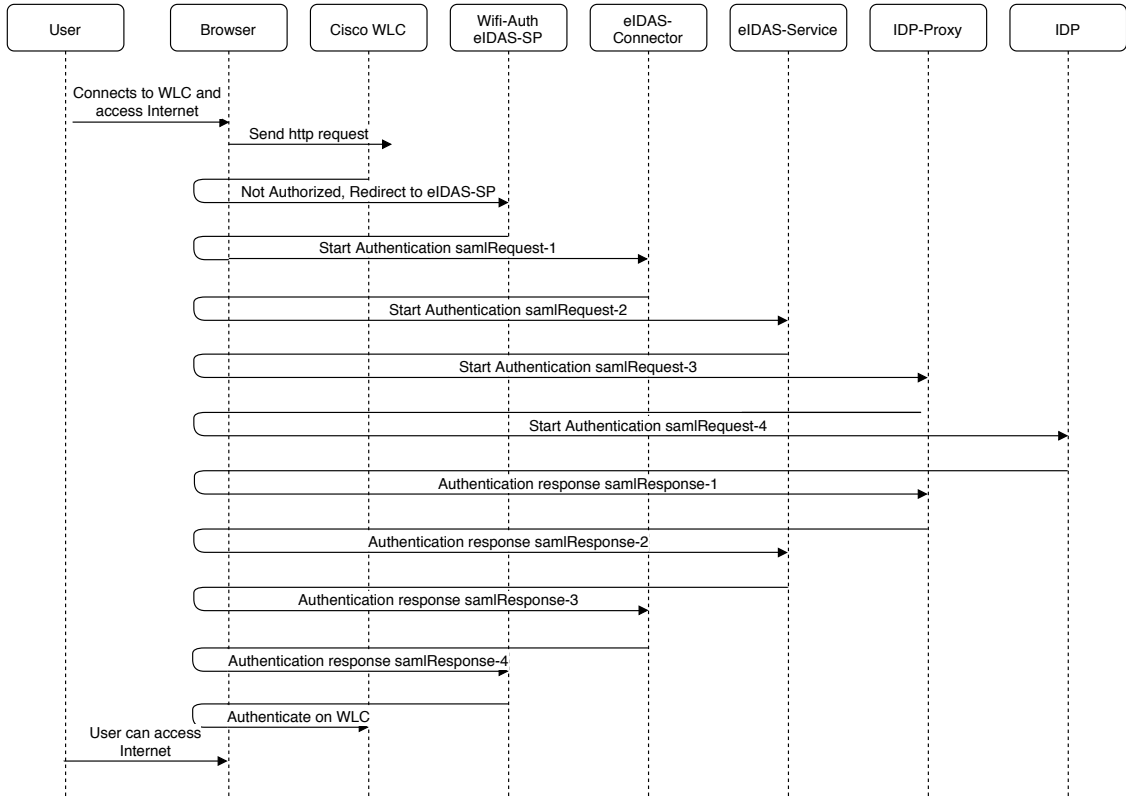


Figure 4.4: Overview of authentication flow for Italian scenario

3. eIDAS-SP asks for user's country and creates a eIDAS (*AuthnRequest*) to get minimum data set after authentication. It then sends request to the eIDAS-Connector using user's browser.
4. eIDAS-Connector Node retrieves and validates eIDAS-SP **metadata** using its metadata signing certificate stored locally. Next it validates eIDAS (*AuthnRequest*) using eIDAS-SP "signing" public-key, which is provided by the metadata file. After validation, it gets user's country and other parameters provided by the eIDAS (*AuthnRequest*). It then creates an eIDAS (*AuthnRequest*) for respective eIDAS-Service and sends it using user's browser.
5. eIDAS-Service Node retrieves and validates eIDAS-Connector **metadata** using its metadata signing certificate stored locally. Next it validates eIDAS (*AuthnRequest*) using eIDAS-Connector "signing" public-key, which is provided by the metadata file. After validation, it gets parameters provided by the eIDAS (*AuthnRequest*). It then creates eIDAS (*AuthnRequest*) for IdP-Proxy and sends it using user's browser.
6. IdP-proxy retrieves and validates eIDAS-Service **metadata** using eIDAS-Service's metadata signing certificate stored locally. Next it validates eIDAS (*AuthnRequest*) using eIDAS-Connector "signing" public-key, which is provided by the metadata file. After validation, it gets parameters provided by the eIDAS (*AuthnRequest*). It then creates SPID (*AuthnRequest*) for IdP and sends it using user's browser.
7. IdP retrieves and validates IdP-Proxy **metadata** using IdP-Proxy's metadata signing certificate stored locally. Next it validates SPID (*AuthnRequest*) using eIDAS-Connector "signing" public-key, which is provided by the metadata file. After validation, it gets parameters provided by the SPID (*AuthnRequest*). It then asks for user credentials and authenticate user.

Response part for authentication flow for Italian scenario follows these steps

8. After successful authentication IdP creates an SPID (*Response*) for IdP-Proxy with user's minimum data set. Next IdP digitally signs the response with its private-key stored locally and sends it to IdP-Proxy using user's browser.

9. IdP-Proxy validates the SPID (*Response*) using IdP “signing” public-key, which is provided by the metadata file. It then creates an eIDAS (*Response*) with user’s minimum data set. Attribute information was packaged in a specific SAML2 *Attribute* element, which has been defined in the SAML2 eIDAS profile. Then the eIDAS attributes element is encrypted by a session key generated randomly and session key is encrypted by (asymmetric) “encryption” public key of the eIDAS-Service Node, which is provided by the metadata file of eIDAS-Service. Next the authentication response was digitally signed by the IdP-Proxy with its private-key stored locally and sends to eIDAS-Service Node using user’s browser.
10. eIDAS-Service validates the eIDAS (*Response*) using IdP “signing” public-key, which is provided by the metadata file. Next it decipher session key using its (asymmetric) “encryption” private-key and use session key to decipher the SAML2 (*Response*) encrypted element to get the identification/attributes information. All the attributes were extracted from the SAML2 Assertion element, eIDAS-Service requests the user’s consent to forward his attributes for authentication. If the consent was given, the eIDAS-Service creates a SAML2 (*Response*) by encrypting SAML attributes using session key generated randomly and session key is encrypted by (asymmetric) “encryption” public key of the eIDAS-Connector Node. Next it is digitally signed using its (asymmetric) “signing” private-key and sends to eIDAS-Connector Node using user’s browser.
11. eIDAS-Connector validates the eIDAS (*Response*) using eIDAS-Service’s “signing” public-key, which is provided by the metadata file. Next it decipher session key using its (asymmetric) “encryption” private-key and use session key to decipher the SAML2 (*Response*) encrypted element to get the identification/attributes information. eIDAS-Connector encrypts SAML attributes using session key generated randomly and session key is encrypted by (asymmetric) “encryption” public key of the eIDAS-SP and creates a newly signed SAML2 (*Response*) using its (asymmetric) “signing” private-key and sends to eIDAS-SP using user’s browser.
12. eIDAS-SP validates the eIDAS (*Response*) using eIDAS-Connector’s “signing” public-key, which is provided by the metadata file. Next it decipher session key using its (asymmetric) “encryption” private-key and use session key to decipher the SAML2 (*Response*) encrypted element using its (asymmetric) “encryption” private-key to get the identification/attributes information. It then connects to WLC using SSH and creates a guest user with user’s **PersonIdentifier**. It then sends guest user credentials to WLC using a auto-submit form using user’s browser.
13. WLC validates the request, authenticate the user and allow access to the internet.

### 4.3.2 Authentication flow detail: request part

1. As a Guest user, I starts the authentication process by connecting to the public eIDAS AP using WiFi. Then I writes a http URL <http://www.giornalone.it> in the browser and press enter. The browser tries to access the URL and sends a request on the internet.
2. The request gets from AP to WLC. A Captive Portal is implemented on WLC, it checks if user is authenticated to access the internet. This is a new connection so user is not authenticated and there is no record available in the WLC. WLC creates a http redirect response and redirects to <https://wifi-auth-eid4u.polito.it/SP/populateIndexPage> on Wifi-Auth eIDAS-SP for authentication.

```
[4]
switch_url: http://1.1.1.1/login.html
ap_mac: f4:cf:e2:4e:b7:80
client_mac: 5c:e0:c5:26:c7:e4
wlan: eIDAS
redirect: abc.co
```

3. The browser receives the request and sends a get request to [SP/populateIndexPage](#) On Wifi-Auth eIDAS-SP. Wifi-Auth replies with a HTML page asking user to select eIDAS-Connector (Test, Pre-Production, Production) and its country. For this test I am using the Italian test credentials for completing the cycle. So I selected Test Connector and IT, then I clicked “Login with eIDAS”. Which sends a POST request at [SP/IndexPage.action](#) with all attributes.

```
[7]
nodeMetadataUrl:
  https://connector-test-eid4u.polito.it/EidasNode/ConnectorResponderMetadata
eidasNameIdentifier: urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
eidasloa: http://eidas.europa.eu/LoA/low
eidasloaCompareType: minimum
eidasSPType: public
spType: public
switch_url: http://1.1.1.1/login.html
citizenEidas: IT
http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier: http://eidas.
  europa.eu/attributes/naturalperson/PersonIdentifier
http://eidas.europa.eu/attributes/naturalperson/PersonIdentifierType: true
http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName: http://eidas.
  europa.eu/attributes/naturalperson/CurrentFamilyName
http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyNameType: true
http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName: http://eidas.
  europa.eu/attributes/naturalperson/CurrentGivenName
http://eidas.europa.eu/attributes/naturalperson/CurrentGivenNameType: true
http://eidas.europa.eu/attributes/naturalperson/DateOfBirth: http://eidas.europa.
  eu/attributes/naturalperson/DateOfBirth
http://eidas.europa.eu/attributes/naturalperson/DateOfBirthType: true
```

After getting the attributes Wifi-Auth creates a digitally signed eIDAS (*AuthnRequest*) for the selected eIDAS-Connector with its (asymmetric) “signing” private-key, which is stored locally. It then send the (*AuthnRequest*) to eIDAS-Connector using user’s browser at <https://connector-test-eid4u.polito.it/EidasNode/ServiceProvider>.

```
[9]
postLocationUrl: https://connector-test-eid4u.polito.it/EidasNode/
  ServiceProvider
redirectLocationUrl: https://connector-test-eid4u.polito.it/EidasNode/
  ServiceProvider
country: IT
RelayState: MyRelayState
SAMLRequest:
```

4. eIDAS-Connector receives the eIDAS (*AuthnRequest*) at [EidasNode/ServiceProvider](#). It checks and verifies the authenticity of the request and SP using metadata file as described in subsection 2.5.4. Then it gets the attributes from the request and creates a eIDAS (*AuthnRequest*) for the respective eIDAS-Service depending on selected country. In this case it creates a digitally signed eIDAS (*AuthnRequest*) for Italian eIDAS-Service Node with its (asymmetric) “signing” private-key, which is stored locally and sends it to [https://service-test-eid4u.polito.it/EidasNode/ColleagueRequest](#).

```
[11]
SAMLRequest:
RelayState: MyRelayState
token: -CyVdin41Ps34qss77hotqLYlpI$
```

5. eIDAS-Service receives the POST eIDAS (*AuthnRequest*) at [EidasNode/ColleagueRequest](#). It checks and verifies the authenticity of the request and eIDAS-Connector using metadata file. It then replies with a HTML page asking for consent. It shows the attributes requested by the SP. First it shows the “Basic Information”, I clicked on the “Next” button. After

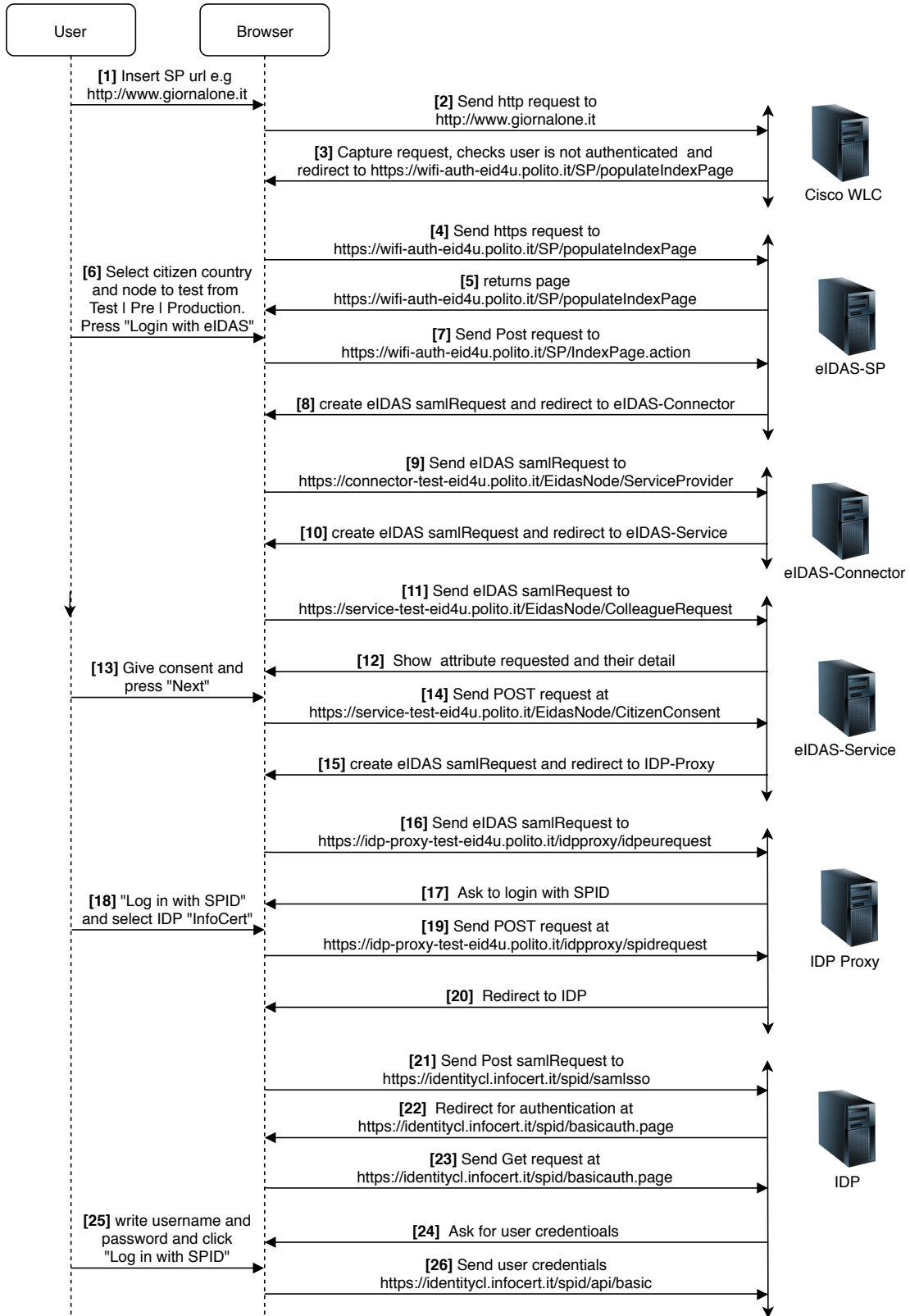


Figure 4.5: Detailed authentication flow request for Italian scenario

that it shows “Additional Information” requested if any, I clicked on the “Next” button. Which sends a POST request with the attributes at [EidasNode/CitizenConsent](https://service-test-eid4u.polito.it/EidasNode/CitizenConsent).

[14]

```

token: mduXkEmKDs__duKCFuQzdWfa90o$
requestId: _T406JgPLEUzsQKksRgNBIQJMWw0oZ-
aPipax93gy68_ZQ_5CSYMHyyaDpHxScuP
http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName: http
://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName
http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName: http://
eidas.europa.eu/attributes/naturalperson/CurrentGivenName
http://eidas.europa.eu/attributes/naturalperson/DateOfBirth: http://eidas
.europa.eu/attributes/naturalperson/DateOfBirth
http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier: http://
eidas.europa.eu/attributes/naturalperson/PersonIdentifier

```

Which creates a digitally signed eIDAS (*AuthnRequest*) for IdP-Proxy with its (asymmetric) “signing” private-key, which is stored locally and sends it using user’s browser at <https://idp-proxy-test-eid4u.polito.it/idpproxy/idpeurequest>.

```

[16]
SAMLRequest:
messageFormat: eidas

```

6. IdP-proxy receives the POST eIDAS (*AuthnRequest*) at [idpproxy/idpeurequest](https://idpproxy/idpeurequest). It checks and verifies the authenticity of the request and eIDAS-Service using metadata file. Then it replies with a HTML page asking for the type of identification. I am using SPID for the authentication, so I clicked on “Login with SPID”. Which then shows all the available IdP’s for the verification of SPID. I am using test credentials for InfoCert IdP, so I clicked on “INFOCERT ID”. It sends a POST request at [idpproxy/spidrequest](https://idpproxy/spidrequest).

```

[19]
idpEntityId: https://identitycl.infocert.it
selectedNode: https://identitycl.infocert.it/metadata/metadata.xml
EidasSAMLID:
_m5IJ5xXKgLnDs50_VA0nmpyQt8vATktTfRgL78L5XMKjZrd2cat0Ql1Aafy3N3

```

Which creates a digitally signed SPID (*AuthnRequest*) for IdP with its (asymmetric) “signing” private-key, which is stored locally and sends it using user’s browser at <https://identitycl.infocert.it/spid/samlssso>.

```

[21]
RelayState: SPID_REQUEST_RELAYSTATE
SAMLRequest:

```

7. IdP receive the POST (*AuthnRequest*) and replies with HTML page at [spid/basicauth.page](https://spid/basicauth.page), which ask for user’s credential. I entered username, password and clicked “Log in with SPID”. It sends a POST request with user’s credential at [spid/api/basic](https://spid/api/basic).

```

[26]
consenso: true
hsf18jt38e21boaugela8n15ah: 84d1eb5f-f1a6-4fe7-a185-226b009dd920
password: <password>
username: <username>

```

Which redirect to [spid/consent.page](https://spid/consent.page) and ask for the consent of the attributes requested. I clicked on “Go on”, which sends a POST request at [spid/api/consent](https://spid/api/consent). Which sends a POST request at [spid/samllead](https://spid/samllead), then a GET request at [spid/samlout](https://spid/samlout), finally creates a digitally signed SPID (*Response*) for IdP-Proxy.

### 4.3.3 Authentication flow detail: response part

8. After successful authentication IdP creates a digitally signed SPID (*Response*) for IdP-proxy with its (asymmetric) “signing” private-key, which is stored locally and sends using user’s browser at <https://idp-proxy-test-eid4u.polito.it/idpproxy/spidresponse>.



[9]

RelayState: SPID\_REQUEST\_RELAYSTATE  
SAMLResponse:

9. IdP-proxy receives (*Response*) at [idpproxy/spidresponse](#). It checks and verifies the authenticity of the response and IdP using metadata file. It get the attributes from the SPID (*Response*) and creates an encrypted and digitally signed eIDAS (*Response*) for eIDAS-Connector as described in subsection 2.5.4. It then sends the eIDAS (*Response*) using user's browser at <https://service-test-eid4u.polito.it/EidasNode/IdpResponse>.

[11]

SAMLResponse:  
username: username

10. eIDAS-Service receives eIDAS (*Response*) at [EidasNode/IdpResponse](#). It checks and verifies the authenticity of the response and IdP-Proxy using metadata file. It decipher the attribute and replies with a HTML page showing the attribute values received to get user's consent. I clicked on "Submit" button, which sends a POST request at [EidasNode/LogSaml](#).

[14]

consentOk:  
logSamlToken: NTM5M2EwNGMtM2JkZi00NDM2LTlhMjEtNTU0NDQyNzgZDgy  
token: mVux00zp3MzuM8OrdZF\_Y\_zB0bY\$

Which then creates an encrypted and digitally signed eIDAS (*Response*) for eIDAS-Connector and sends it using user's browser at <https://connector-test-eid4u.polito.it/EidasNode/ColleagueResponse>.

[16]

SAMLResponse:  
RelayState: MyRelayState

11. eIDAS-Connector receives eIDAS (*Response*) at [EidasNode/ColleagueResponse](#). It checks and verifies the authenticity of the response and eIDAS-Service using metadata file. It gets the attributes by deciphering the attributes element and creates an encrypted and digitally signed eIDAS (*Response*) with the attributes for eIDAS-SP. It then sends the eIDAS (*Response*) using browser at <https://wifi-auth-eid4u.polito.it/SP/ReturnPage>.

[18]

SAMLResponse:  
RelayState: MyRelayState

12. Wifi-Auth eIDAS-SP receives eIDAS (*Response*) at [SP/ReturnPage](#). It checks and verifies the authenticity of the response and eIDAS-Connector using metadata file. It then decipher the eIDAS (*Response*) and sends the attributes at [SP/populateReturnPage](#). Where it connects to the WLC using SSH at the IP "192.168.10.10" and port "20". It then creates a guest user using Person Identifier attribute on WLC and replies with a HTML page containing a auto-submit form with user credentials at <http://1.1.1.1/login.html> (IP address of the WLC).

[22]

username: <username>  
password: <password>  
buttonClicked: 4  
redirect\_url: https://www.google.com/

13. WLC receives the POST request at [urllogin.html](#). It validate credentials and authenticate the user. It then redirects the user to <http://www.giornalone.it> because it is provided in the "redirect\_url" parameter.

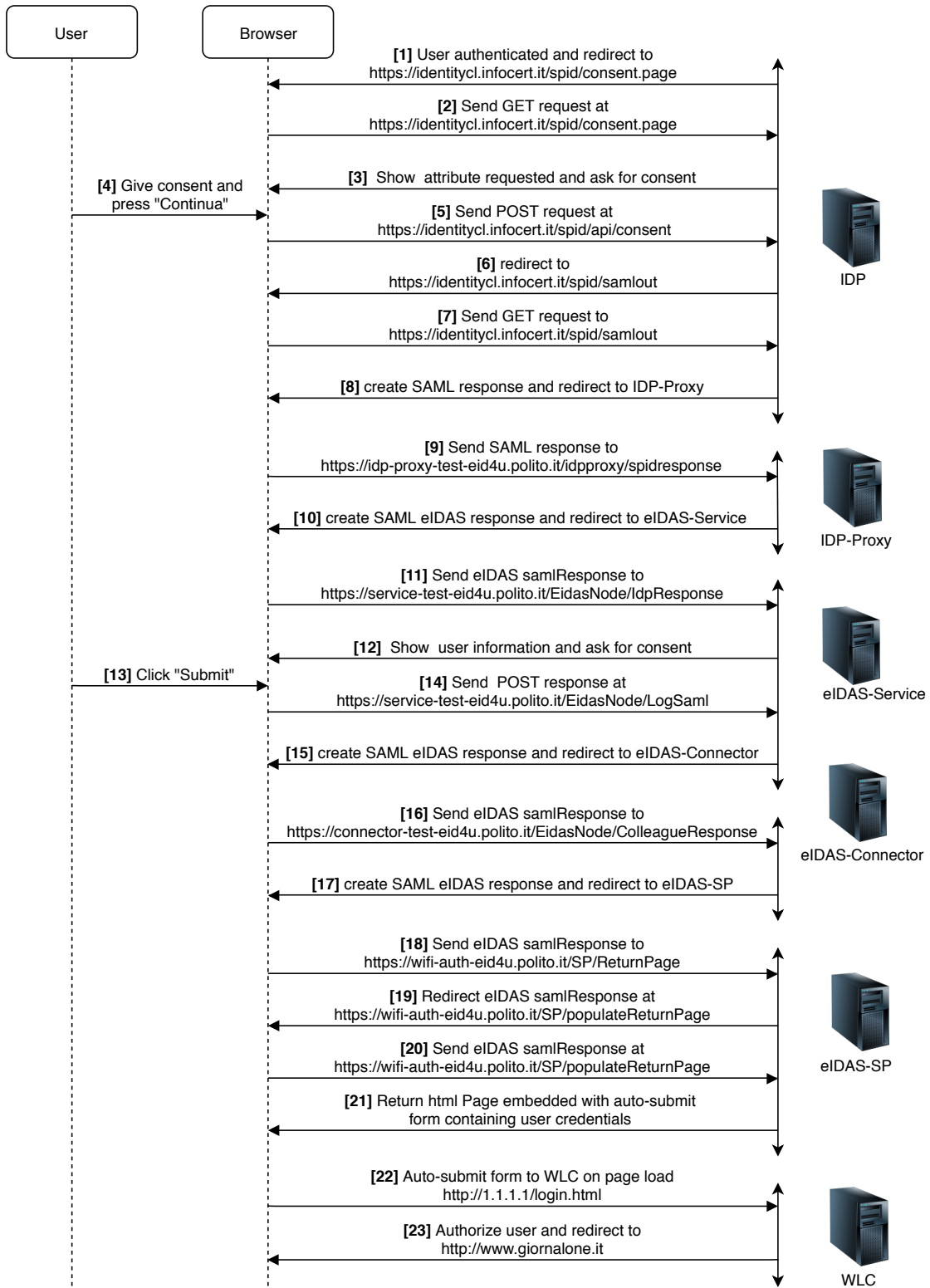


Figure 4.6: Detailed authentication flow response for Italian scenario

## 4.4 Wifi-Auth eIDAS-SP implementation

### 4.4.1 SSH library

After authenticating using eIDAS framework Wifi-Auth connects to WLC using SSH connection and creates a guest user. This is implemented using `jsch` library. It also uses `slf4j` library for log purposes. Both of these libraries are added in `pom` file, which is present in the project directory at location:

```
eidass > build > EIDAS-WIFI-AUTH > pom.xml
```

`slf4j` library is already included in the code, lines of code for adding `jsch` library is provide in Figure 4.7.

---

```
<dependency>
  <groupId>com.jcraft</groupId>
  <artifactId>jsch</artifactId>
  <version>0.1.55</version>
</dependency>
```

---

Figure 4.7: `jsch` library for user creation.

### 4.4.2 Guest user properties

All of the properties for creating user is present in a separate file to make the changes easier. These are present in `user.properties` file. Which includes both the properties for creating the guest user on WLAN and also for connecting to the WLC using SSH. Properties for creating user are as following:

- **u\_wlan:** The SSID of the WLAN.
- **u\_type:** The type of the user created, in our case it is guest.
- **u\_lifetime:** This is the time in seconds for how long user will last. After that it will be deleted automatically.
- **u\_description:** This fields gives a description of the user.
- **tokenString:** This string is compose of characters which are used for creating a random token. We are using eIDAS `PersonIdentifier` attribute and combining it with a random string of 10 characters to create username. The username looks like `PersonIdentifier_xxxxxxxxxx`.
- **passwordString:** We have a separate string for creating password because it also includes special characters. These are not included in *tokenString because special characters are not allowed in username field*.

Properties for SSH connection are as following

- **sshIP:** This field includes the IP of the WLC.
- **sshPort:** The port to connect. In our case it is 22.
- **sshUsername:** Username of the user for authenticating SSH.
- **sshPassword:** Password of the user for authenticating SSH.

---

```
u_wlan=2
u_type=guest
u_lifetime=600
u_description=guest
tokenString=0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
passwordString=0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz!@#%$%^&*~_+

sshIP=192.168.10.10
sshPort=22
sshUsername=admin.user
sshPassword=abcdef12345
```

---

Figure 4.8: Attributes configuration file.

### 4.4.3 Function createUser

Guest user is created after the authentication is completed in `emReturnAction.java` file. The function is provided in Figure 4.9. It takes person identifier return from the authentication as parameter for creating the user. First it create the command to create user using `emcreateCMD` function. After that it creates a session using `jsch` library and set the configurations. After connecting in SSH with WLC, it requires the admin to login first. So three commands are send for creating the guest user: username as first, password as second and create user command as third.

### 4.4.4 Function createCMD

This function takes as parameter person identifier and use it to create username. It creates random string of 10 characters using `randomString()` function and password of 15 characters using `randomPassword()` function. It then creates username as a combination of random string and person identifier (`PersonIdentifier|xxxxxxxx`). Finally it creates the command for creating guest user using the parameters and return it as a string to send to WLC. It is shown in Figure 4.10.

### 4.4.5 Functions randomString and randomPassword

These functions implement the functionality to generate random String and password using characters provided by `tokenString` and `passwordString` respectively. These are shown in Figure 4.11

### 4.4.6 Login form

After guest user is created its username and password is provided as a form to the `returnPage.jsp`. Which is automatically submitted on load to WLC to authenticate automatically.

```
POST: http://1.1.1.1/login.html
buttonClicked: 4
password: FXotwYSW^qbEZIL
redirect_url: https://www.google.com/
username: IT/IT/INFC0001TESTEU_nSHrZAM61w
```

```
public void createUser(String PID) throws IOException, JSchException,
InterruptedException {
    logger.error("create user");
    sshCMD = createCMD(PID);
    logger.error(sshCMD);
    java.util.Properties configuration = new java.util.Properties();

    configuration.put("kex",
        "diffie-hellman-group1-sha1,diffie-hellman-group-exchange-sha1,
        diffie-hellman-group14-sha1,diffie-hellman-group-exchange-sha256");
    configuration.put("StrictHostKeyChecking", "no");
    try {
        final JSch jsch = new JSch();
        final Session session = jsch.getSession(sshUsername, sshIP,22);

        session.setPassword(sshPassword);
        session.setConfig(configuration);
        session.connect();
        if (!session.isConnected()) {
            throw new RuntimeException("cannot connect session");
        }
        List<String> commands = new ArrayList<String>();
        commands.add(sshUsername);
        commands.add(sshPassword);
        commands.add(sshCMD);

        Channel channel=session.openChannel("shell");
        channel.setOutputStream(System.out,true);
        PrintStream shellStream = new PrintStream(channel.getOutputStream());
        channel.connect();
        for(String command: commands) {
            shellStream.println(command);
            shellStream.flush();
        }
        TimeUnit.MILLISECONDS.sleep(1000);

        channel.disconnect();
        session.disconnect();
    } catch (Exception e) {
        logger.error("ERROR");
        System.err.println("ERROR: Connecting via shell to "+sshIP);
        e.printStackTrace();
    }
}
```

---

Figure 4.9: Function for creating user on WLC.

#### 4.4.7 Cryptography certificates

Cryptography certificates for integrity, authenticity of the messages and identification of the entities involved are created by docker container: `letsencrypt` and `certs`. `letsencrypt` is a docker container, which uses `certbot` [62] to provide TLS/SSL certificates from Let's Encrypt [63]. We use `certbot` to generate a certificate and use it in `certs` container to generate SAML/SAML Metadata Signing and XML Encryption certificates.

First we generate a certificate using `letsencrypt` by running the `Dockerfile` present at

---

```
public String createCMD(String pid) {
    String cmd;
    userUsername = pid + "_";
    String usertoken = randomString(10);
    userUsername += usertoken;
    userPassword = randomPassword(15);
    cmd = "config netuser add " + userUsername + " " + userPassword + "
        wlan " + u_wlan +
            " userType "+ u_type + " lifetime " + u_lifetime + "
            description " + u_description;
    return cmd;
}
```

---

Figure 4.10: Function for user creation command for WLC.

---

```
static SecureRandom rnd = new SecureRandom();

String randomString( int len ){
    StringBuilder sb = new StringBuilder( len );
    for( int i = 0; i < len; i++ )
        sb.append( tokenString.charAt( rnd.nextInt(tokenString.length()) ) );
    return sb.toString();
}

String randomPassword( int len ){
    StringBuilder sb = new StringBuilder( len );
    for( int i = 0; i < len; i++ )
        sb.append( passwordString.charAt( rnd.nextInt(passwordString.length()) ) );
    return sb.toString();
}
```

---

Figure 4.11: Function for generating string for username and password.

---

location

```
letsencrypt > test > docker-compose.yml
```

It will generate two files `fullchain.pem` and `privkey.pem`. We need to copy these files to “certs/test/rev-proxy” with name `rev-proxt-tls.pem` and `rev-proxt-tls.key` respectively. Next we need to delete certificates from “certs/test/wifi-auth” and restart the Wifi-Auth docker. On-start it will check if certificates are present, if not it will generate all three certificates using the certificate present in `rev-proxy` file.

## 4.5 Configuration of network elements

### 4.5.1 Fortigate-60D introduction

Fortigate-60D provides all in one security solution for organisations. It allows to create a secure network using firewall policies and interfaces.

## Firewall interfaces

Interface either physical or virtual allow the flow of traffic between internal networks and internet. For our specific set-up we are using two physical interfaces internal7 as internal network and wan1 as external network (Internet). In internal7 we created two VLAN (Virtual LAN), one for management network and other one for WiFi clients. Virtual LAN makes the flow of data isolated at data link layer. This is to allow only users from management network to access configuration interface.

Physical (6)					
	dmz		10.10.10.1 255.255.255.0		Physical Interface
	internal7		0.0.0.0 0.0.0		Physical Interface
	management		192.168.10.1 255.255.255.0		VLAN
	wifi_clients		192.168.20.1 255.255.255.0		VLAN
	wan1		130.192.1.110 255.255.255.192		Physical Interface
	wan2		0.0.0.0 0.0.0		Physical Interface

Figure 4.12: Fortigate firewall interfaces

## Firewall policy

After creating the interfaces we need to specify firewall policies for communicating between interfaces. As shown in the Figure 4.13 we have created 5 policies.

ID	Name	Source	Destination	Schedule	Service	Action	NAT
management →  wan1 1							
1	management2internet	all	all	always	ALL	ACCEPT	Enabled
management →  wifi_clients 1							
2	management2wifi_clients	all	all	always	ALL	ACCEPT	Enabled
wan1 →  management 1							
5	internet2authserver	external-eidas-auths	wifi-auth-server	always	HTTP HTTPS	ACCEPT	Disabled
wifi_clients →  wan1 1							
3	wifi_clients2internet	all	all	always	ALL	ACCEPT	Enabled
Implicit 1							
0	Implicit Deny	all	all	always	ALL	DENY	

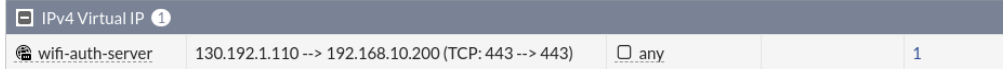
Figure 4.13: Fortigate firewall policies

- management2internet: This policy allow all devices in management network to access internet.
- management2wifi\_clients: This policy allows all devices in management network to access all devices in wifi\_clients network.
- internet2authserver: This policy allows only authorised IP (eIDAS-Connector, IdP) from internet to have access to Wifi-Auth server in internal management network.
- wifi\_clients2internet: This policy allows all devices in wifi\_clients network to access internet.
- Implicit Deny: This policy deny all communication which is not allowed by any other policy.

## Virtual IP

Fortigate is used as NAT for the communication between external and internal network. NAT is the mapping of one IP to another IP address. When NAT is not used Fortigate call these Virtual

IP addresses IP. [59]. This allows the mapping of a public IP to an internal network IP. Wifi-Auth eIDAS SP is present in the internal network, So it is not possible to access it from outside network. To make it accessible from outside a virtual IP is created as shown in Figure 4.14. Which will redirect connections from outside on port 443 ( HTTPS ) to Wifi-Auth server.



IPv4 Virtual IP 1			
wifi-auth-server	130.192.1.110 --> 192.168.10.200 (TCP: 443 --> 443)	any	1

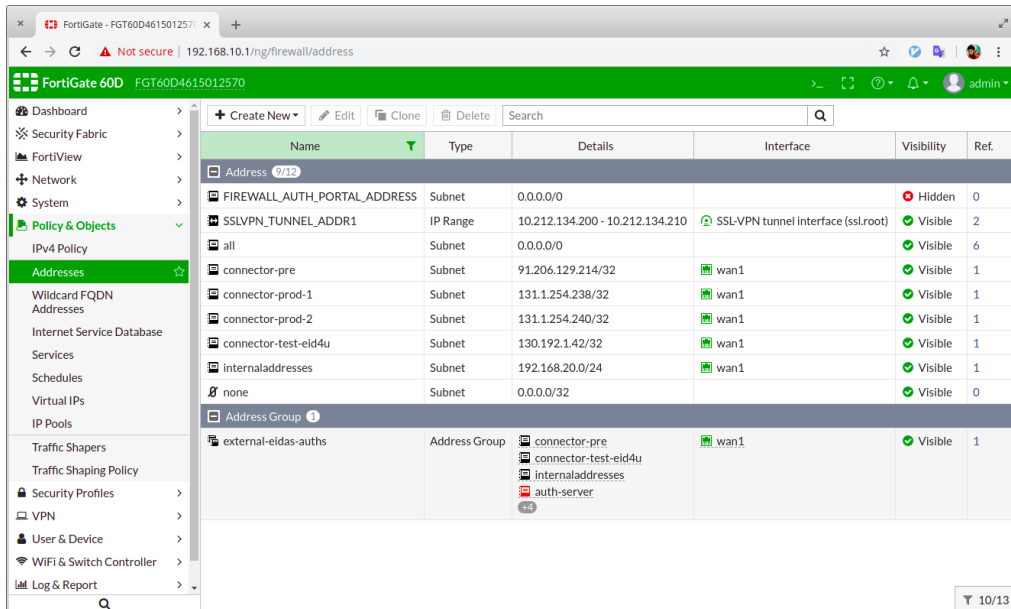
Figure 4.14: Virtual IP for Wifi-Auth SP

## Address and Address Group

Address or address ranges are created to apply a specific security policy. We have created following addresses as shown in Figure 4.13.

- connector-test-eid4u: Address for testing connector machine.
- connector-pre: Address for pre production connector machine.
- connector-pro-1: Address 1 for production connector machine.
- connector-pro-2: Address 2 for production connector machine.
- internaladdresses: Address range for management network for testing.

Address groups can also be created in Fortigate to make it easy to apply same policies on a group of addresses or address ranges. In our case we have created a group of address to allow access to Wifi-Auth eIDAS SP. All eIDAS eIDAS-Connector are added in this group to have access to Wifi-Auth eIDAS SP metadata.



Name	Type	Details	Interface	Visibility	Ref.
<b>Address 9/12</b>					
FIREWALL_AUTH_PORTAL_ADDRESS	Subnet	0.0.0.0/0		Hidden	0
SSLVPN_TUNNEL_ADDR1	IP Range	10.212.134.200 - 10.212.134.210	SSL-VPN tunnel interface (ssl.root)	Visible	2
all	Subnet	0.0.0.0/0		Visible	6
connector-pre	Subnet	91.206.129.214/32	wan1	Visible	1
connector-prod-1	Subnet	131.1.254.238/32	wan1	Visible	1
connector-prod-2	Subnet	131.1.254.240/32	wan1	Visible	1
connector-test-eid4u	Subnet	130.192.1.42/32	wan1	Visible	1
internaladdresses	Subnet	192.168.20.0/24	wan1	Visible	1
none	Subnet	0.0.0.0/32		Visible	0
<b>Address Group 1</b>					
external-eidas-auths	Address Group	connector-pre connector-test-eid4u internaladdresses auth-server	wan1	Visible	1

Figure 4.15: Addresses and Address Group for policies

## 4.5.2 Cisco WLC 2504

Cisco WLC provides a single solution for configuring, managing and supporting corporate wireless network. It provides a centralised solution for providing network access in a large network with



multiple Access Point. It provides both CLI and web GUI for configuration. The CLI interface is required to assign a IP to the WLC and some other important parameters. After that all configuration can be done using web GUI. Web GUI is shown in the Figure 5.5. Web GUI provides two interfaces Simple and Advanced. For our configuration we will be using Advanced interface as it provides more control. For our set-up we created a WLAN ( Wireless LAN ) for guest user to connect. To provide authentication to guest users we are using Captive Portal with eIDAS SP authentication. Once user is authenticated it allows them to access internet.

### Access Control List (ACL)

ACL works as white-listing to filter traffic to enter or leave WLAN. For our set-up, before authentication we need to only allow traffic to pass through to the entities involved in eIDAS network e.g Wifi-Auth eIDAS SP, eIDAS Connector, IdP etc.

### WebAuth SecureWeb for HTTPS

Once eIDAS authentication cycle is completed, it requires to submit a form on WLC with the respective credentials to allow access to the internet. To make this request on HTTPS it requires a valid certificate on WLC. If valid certificate is not present, it will show warning on submitting the form. Once a valid certificate is downloaded on the WLC using Web GUI, the warning will not be received. In our set-up we don't have a valid certificate in this phase of project. So we have disabled WebAuth SecureWeb to make this request on http for the testing phase. For production environment, a valid SSL certificate is required and local DNS server needs to redirect to the WLC. Which will allow to send authentication request to WLC over HTTPS.

## 4.6 Script for creating ACL rules

The script is created in python to connect to the WLC using SSH and create ACL rules on WLC for domain names. A list of domain names are provided to the script. It loops through the domain names and find all the IP's associated with that domain name. It is using socket library function `gethostbyname_ex(dns_name)` to get IP's registered to a domain name. Then it adds two ACL rule for each IP, one for inbound and other for outbound. The important parts of the script is provided below

```
ips = ["192.168.10.10"] # WLC
urls = ["posteid.poste.it", "idp-proxy.pre.eid.gov.it",
        "identity.infocert.it", "connector.eid.gov.it",
        "wifi-auth-eid4u.polito.it", "connector-test-eid4u.polito.it", "identitycl
        .infocert.it", "connector.pre.eid.gov.it",
        "sipeps-test.gov.si", "sicas.setcce.si", "vidp.gv.at",
        "test1.a-trust.at", "identity.sieltecloud.it"]
for ip in ips:
    cisco_wlc = {
        'device_type': 'cisco_wlc',
        'ip': ip,
        'username': un,
        'password': pw}
    devices.append(cisco_wlc)
for device in devices:
    logger.info("Connecting to %s", device['ip'])
    # connect to the device w/ netmiko
    try:
        net_connect = netmiko.ConnectHandler(**device)
    except:
        logger.error("Failed to connect to %s", device['ip'])
```

```
        continue

    commands = [] # commands to run
    # create ACL
    commands.append("config acl create acl-guest ")
    i = 1
    for url in urls:
        # getting all ips from domain name
        url_ips = socket.gethostbyname_ex(url)[2]
        for url_ip in url_ips:
            # outbound
            commands.append("config acl rule add acl-guest " + str(i))
            commands.append("config acl rule action acl-guest " + str(i) + "
                permit")
            commands.append("config acl rule source address acl-guest "+ str(i)
                +" "+ url_ip +" 255.255.255.255")
            i+=1
            # inbound
            commands.append("config acl rule add acl-guest " + str(i))
            commands.append("config acl rule action acl-guest " + str(i) + "
                permit")
            commands.append("config acl rule destination address acl-guest "+
                str(i) +" "+ url_ip +" 255.255.255.255")
            i+=1
    for cmd in commands:
        logger.info("Sending cmd: %s", cmd)
        this_cmd = net_connect.send_command(cmd)
        config_filename_f = open(config_filename, 'a')
        config_filename_f.write(this_cmd)
        config_filename_f.write('\n')
        config_filename_f.close()
```

## 4.7 Testing authentication cycle using TestCafe

For rigorously testing the Wifi-Auth eIDAS-SP with the eIDAS network. We create a automated test for authentication of a test user using Node.js [60] tool TestCafe [61]. TestCafe allows to create tests using JavaScript and run them on different browsers. The code for testing the service is provided below

```
import { Selector } from 'testcafe';
import { ClientFunction } from 'testcafe';

const getLocation = ClientFunction(() => document.location.href);

fixture 'Getting Started'
    .page 'https://wifi-auth-eid4u.polito.it/SP/populateIndexPage';

for(let c = 0; c < 1; c++)
    test('eIDAS Authentication test'+c, async t => {
        await t
            .click('#citizeneidas[value=IT]')
            .click('.button-spide-fix')
            .click('#buttonNextSlide1')
            .click('#buttonNextSlide2')
            .click('.showBTN2')
```

```

.click(Selector("a").withAttribute('title', 'Inforcert S.P.A.'))
.typeText('#username', 'TESTE01')
.typeText('#password', 'abcdef1234')
.click('.button-spid')
.click('.button-spid') // keep it going
.click('#buttonNext') //service

.expect(getLocation()).contains('https://1.1.1.1/login.html');
});

```

## 4.8 Authentication cycle

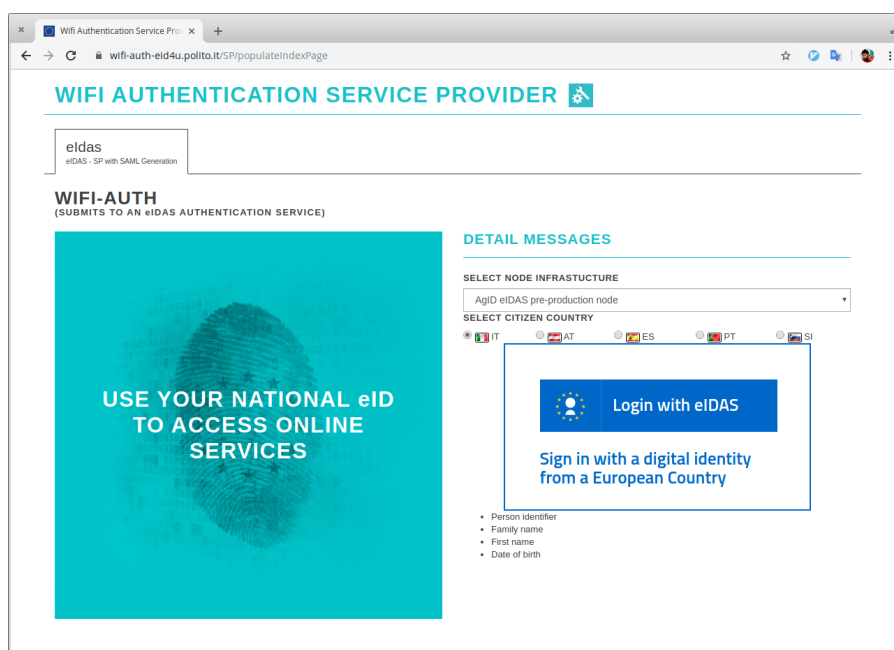


Figure 4.16: Wifi-Auth eIDAS-SP Captive Portal authentication Step-1

USE YOUR NATIONAL eID TO ACCESS ONLINE SERVICES

WIFI-AUTH  
WITH LEVEL OF ASSURANCE low  
IS REQUESTING THE FOLLOWING ATTRIBUTES

Step 1 | 3

YOUR BASIC INFORMATION  
NATURAL PERSON

- ✓ Family Name
- ✓ First Name
- ✓ Date of Birth
- ✓ Uniqueness Identifier

CANCEL NEXT

Figure 4.17: Wifi-Auth eIDAS-SP Captive Portal authentication Step-2

USE YOUR NATIONAL eID TO ACCESS ONLINE SERVICES

WIFI-AUTH  
WITH LEVEL OF ASSURANCE low  
IS REQUESTING THE FOLLOWING ATTRIBUTES

Step 2 | 3

YOUR ADDITIONAL INFORMATION

If you don't provide information you may be denied.

CANCEL BACK NEXT

Figure 4.18: Wifi-Auth eIDAS-SP Captive Portal authentication Step-3

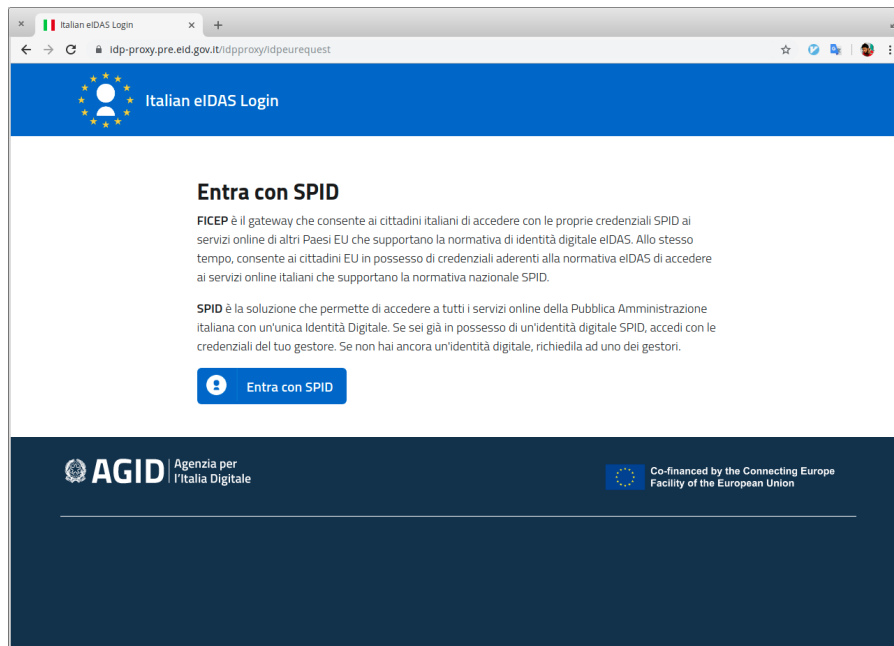


Figure 4.19: Wifi-Auth eIDAS-SP Captive Portal authentication Step-4

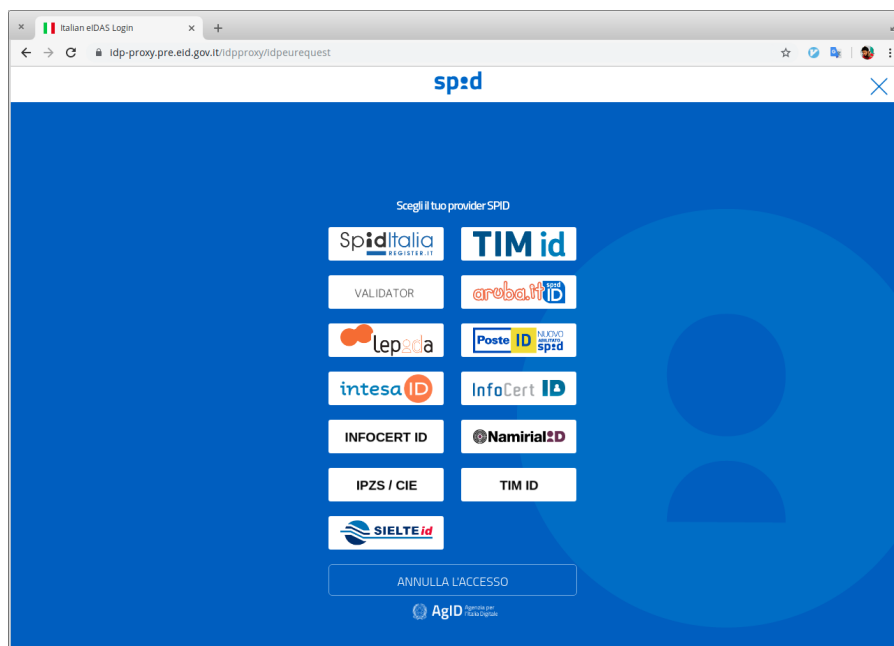


Figure 4.20: Wifi-Auth eIDAS-SP Captive Portal authentication Step-5

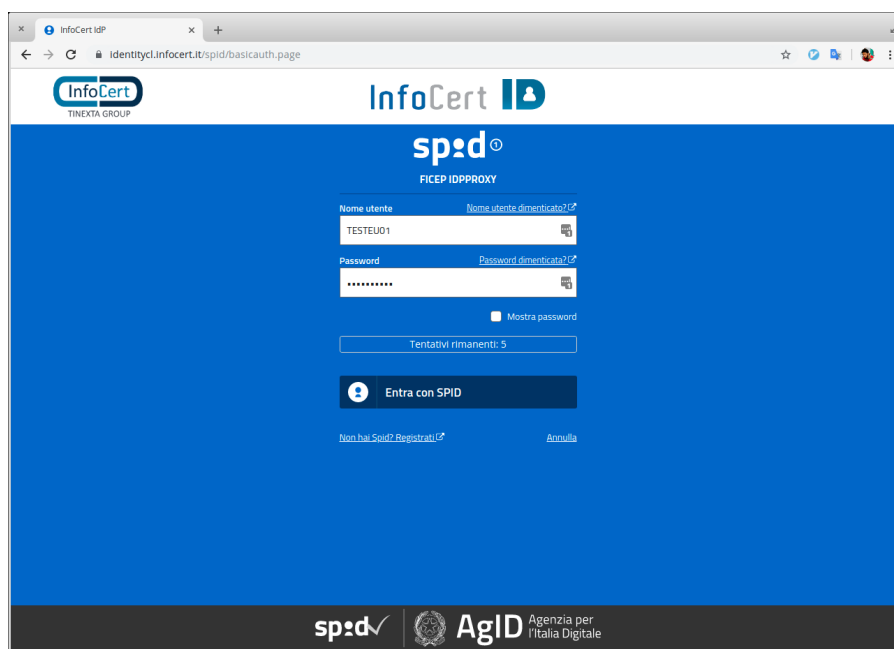


Figure 4.21: Wifi-Auth eIDAS-SP Captive Portal authentication Step-6

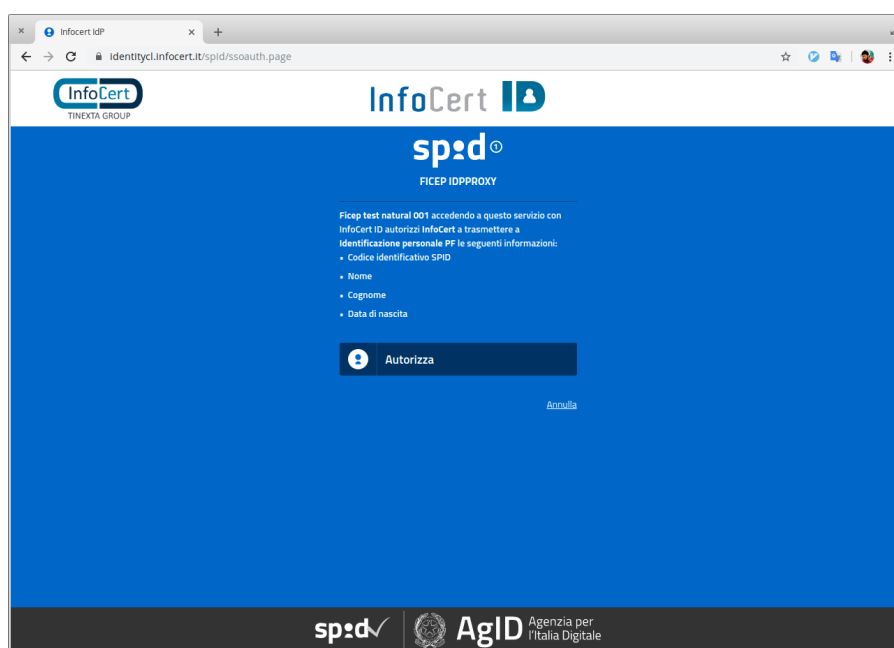


Figure 4.22: Wifi-Auth eIDAS-SP Captive Portal authentication Step-7

Consent

service.pre.eid.gov.it/EidasNode/IdpResponse

USE YOUR NATIONAL eID TO ACCESS ONLINE SERVICES

WIFI-AUTH  
WITH LEVEL OF ASSURANCE low 7  
IS REQUESTING THE FOLLOWING ATTRIBUTES

Step 3 | 3

YOUR RESUME

NATURAL PERSON

First Name  
Arianna

Family Name  
Garbini

Date of Birth  
1968-05-22

Uniqueness Identifier  
IT/IT/INFC0001TESTEU

CANCEL SUBMIT

Figure 4.23: Wifi-Auth eIDAS-SP Captive Portal authentication Step-8

Wifi Authentication Service Provider

wifi-auth-eid4u.polito.it/SP/ReturnPage

WIFI AUTHENTICATION SERVICE PROVIDER

WIFI-AUTH  
SAMLRESPONSE RECEIVED BY THE SP

REDIRECTION IN PROGRESS, PLEASE WAIT

USE YOUR NATIONAL eID TO ACCESS ONLINE SERVICES

Waiting for wifi-auth-eid4u.polito.it...

Figure 4.24: Wifi-Auth eIDAS-SP Captive Portal authentication Step-9

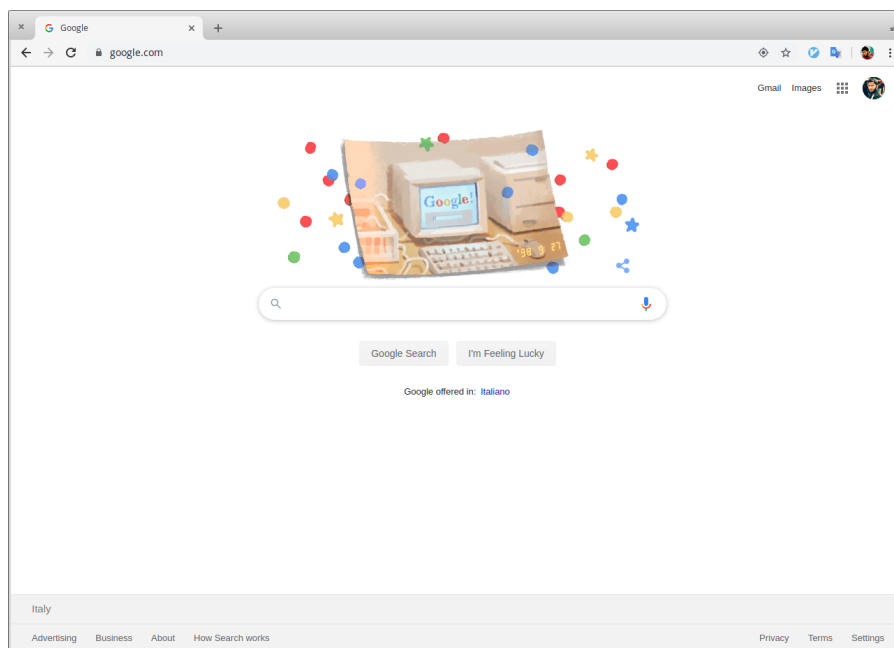


Figure 4.25: Wifi-Auth eIDAS-SP Captive Portal authentication Step-10



## Chapter 5

# Installation and configuration of WiFi access with eIDAS-SP and Polito wireless infrastructure

### 5.1 Installation of Wifi-Auth eIDAS-SP application

We developed Wifi-Auth eIDAS SP compatible with eIDAS version 1.4.4. We added a separate machine on the management network to work as eIDAS SP as shown in Figure 4.2. We selected Ubuntu server 18.04 as platform for SP, because it is latest LTS (Long Term Support) version available and will be supported for next five years till 2023. After successful installation, we were able to connect to the SP machine using SSH from management machine. We are using docker and docker-compose for easier management and re-usability of code. Also we are using git for version control of the code.

#### 5.1.1 Architecture

The code is available on our git repository <https://git-sec.polito.it/electronic-identity/wifi-auth-eid4u.git>. Which allows us to clone the repository on new machine. We are using docker and docker-compose for the easier management and re-usability. The architecture of our code is shown below.

```
/
├── 1: certs
│   ├── 1.1: test
│   │   └── 1.1.1: Dockerfile
│   └── 1.2: Dockerfile
├── 2: eidas
│   ├── 2.1: build
│   │   ├── 2.1.1: EIDAS-WIFI-AUTH
│   │   └── 2.1.2: Dockerfile
│   ├── 2.2: config
│   ├── 2.3: wifi-auth
│   │   └── 2.3.1: test
│   │       └── 2.3.1.1: Dockerfile
├── 3: haproxy
│   ├── 3.1: test
│   │   └── 3.1.1: Dockerfile
└── 4: letsencrypt
    ├── 4.1: test
    │   └── 4.1.1: docker-compose.yml
```

```
└─ 5: maven
   └─ 5.1: Dockerfile
└─ 6: tomcat
   └─ 6.1: Dockerfile
└─ 7: docker-compose.yml
```

### 5.1.2 Docker and Docker-compose installation

- We installed Docker and it's pre-requisites following installations instruction provided in docker documentation [66]. The steps we followed are given below

1. Installing packages to allow apt to use a repository over HTTPS:

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent
software-properties-common
```

2. Adding Docker's official GPG (GNU Privacy Guard) [67] key

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
```

3. Setting up the stable repository

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/
linux/ubuntu \
    $(lsb_release -cs) stable"
```

4. Updating the apt package index and installing latest docker engine

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

- Then we installed docker compose on the machine following from tutorial provided at <https://docs.docker.com/compose/install/>. Docker-Compose is a tool for defining and running multi-container Docker applications. It use a YAML file to configure application's services.

1. Downloading the current stable release of Docker Compose

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/
docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2. Applying executable permissions to the binary

```
sudo chmod +x /usr/local/bin/docker-compose
```

### 5.1.3 Source code

To access the code from repository we added SSH key to our GitLab. We created SSH key, which we will use to authenticate and clone the repository.

1. Creating SSH key

```
ssh-keygen -t rsa -b 4096 -C "muhammadali.anjum@studenti.polito.it"
```

2. Cloning the repository form git

```
git config user.email "muhammadali.anjum@studenti.polito.it"
git clone git@git-sec.polito.it:electronic-identity/wifi-auth-eid4u.git
```

3. Finally running Wifi-Auth using docker-compose

```
sudo docker-compose -f docker-compose.test.yml up --build
```

## 5.2 Fortigate-60D

Fortigate-60D is acting as a firewall and NAT. Only authorised devices can access Wifi-Auth eIDAS-SP from outside the network. It provides a web interface for configuration. To configure network interface In the Figure 4.12 it shows the interfaces and their configuration.

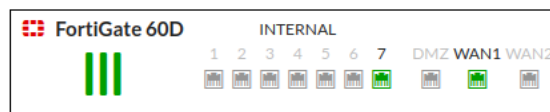


Figure 5.1: Fortigate-60D

### 5.2.1 Firewall interfaces

In this section it will describe how to create and configure interfaces in Fortigate-60D using web GUI. Configuration of interfaces can be find at:

Network > Interfaces

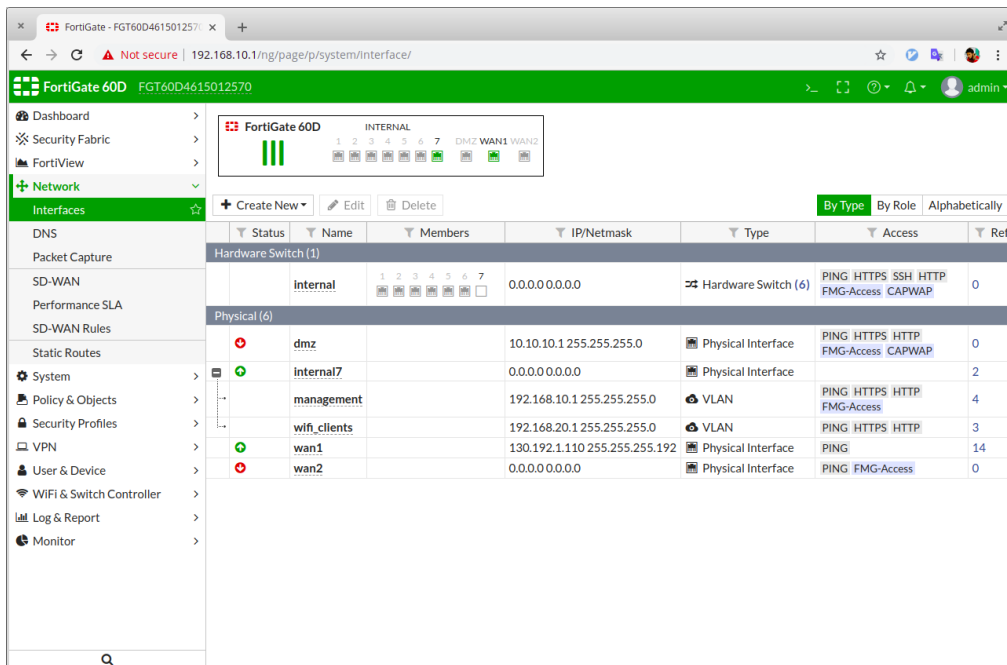


Figure 5.2: Fortigate-60D GUI interface configuration

### 5.2.2 Firewall policy

Configuration of policies can be find at:

## Policy &amp; Objects &gt; IPv4 Policy

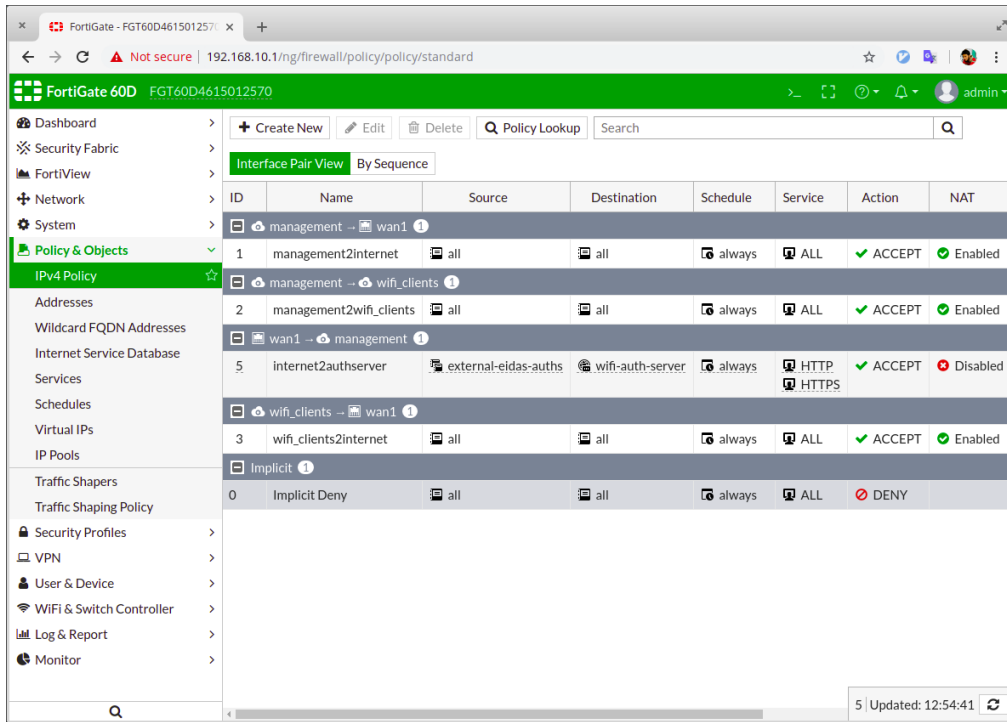


Figure 5.3: Fortigate-60D GUI policy configuration

### 5.2.3 Virtual IP

Virtual IP can be configured from GUI at:

## Policy &amp; Objects &gt; Virtual IPs

Creating a Virtual IP

1. Go to Policy & Objects > Virtual IPs
2. Click Create New > Virtual IP
3. Write a unique name in the **Name** Field
4. Write the public IP in the **External IP Address/Range** field
5. Write the internal IP in the **Mapped IP Address/Range** field. In over case we added the IP of the Wifi-Auth eIDAS SP.
6. Enable **Port Forwarding** and select TCP port 443 in both **External Service Port** and **Map to Port**
7. press **OK** to complete creation of Virtual IP.

## 5.3 Cisco WLC 2504

Cisco WLC provides a web interface as well as command line interface for configuration. Basic web interface provides a user friendly dashboard to display network summary and to configure some basic functionality. For more complicated configurations it provides a **Advanced** web interface, which is shown in Figure 5.5

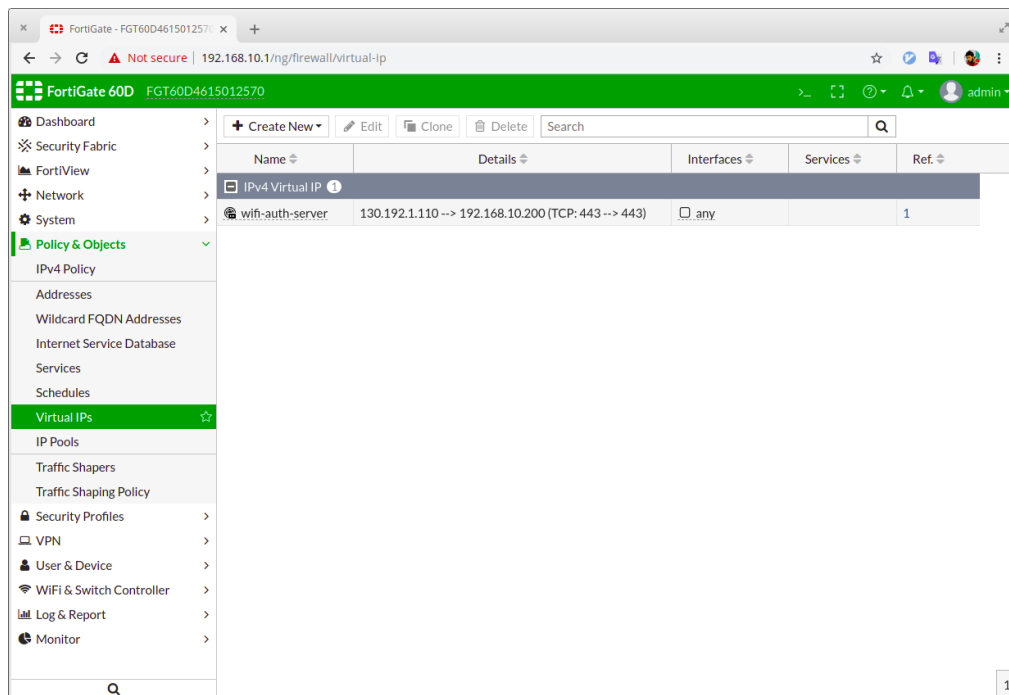


Figure 5.4: Virtual IP configuration GUI

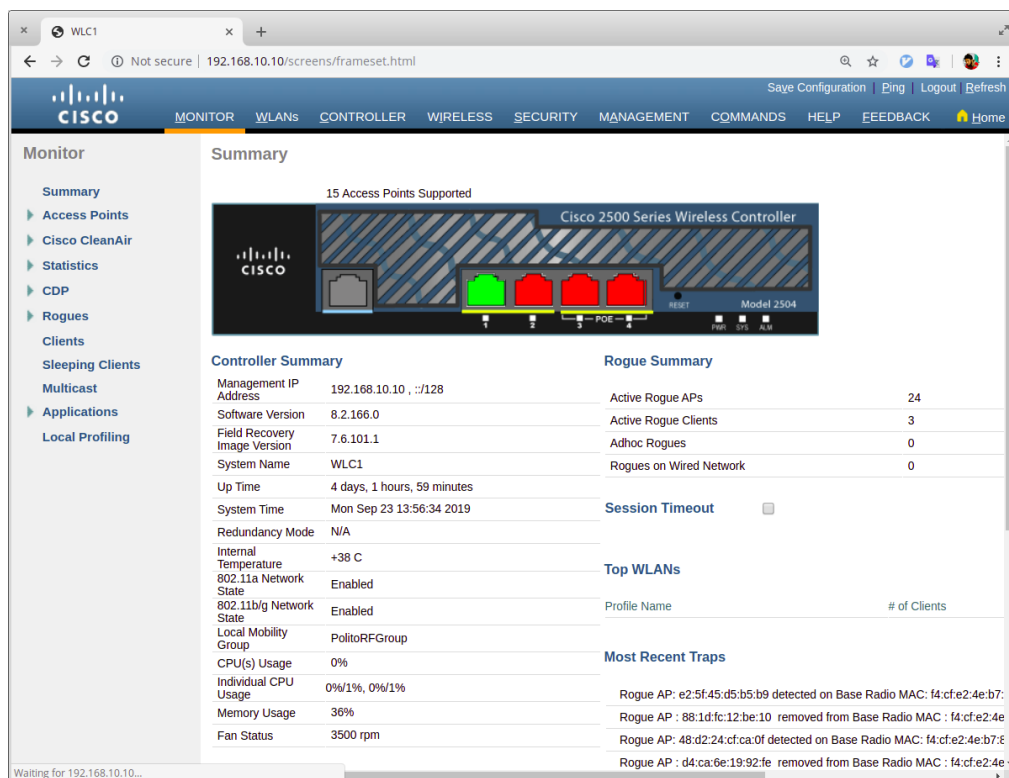


Figure 5.5: Cisco WLC web interface

### 5.3.1 WLAN

WLAN is a local area network of two or more devices using wireless communication. The configuration of WLAN is accessible from WLAN tab from advanced web interface.

### 5.3.2 Creating WLAN

The steps needed to create a WLAN is as following:

1. Go to WLANs
2. Select **Create New** and press **Go**
3. Write unique profile name up to 32 characters in **Profile Name** field.
4. Write SSID up to 32 characters in **SSID**.
5. Press **Apply** to commit.
6. Enable **Status** to make WLAN enable.
7. Select wifl\_clients from **Interface/Interface Group(G)**
8. Go to Security >Layer 3
9. Select **Web Policy** from drop down.
10. Select **Authentication** radio button
11. Select specific ACL ( Access Control List ) from drop down in **Preauthentication ACL**. Creating and editing ACL is explained later.
12. In **Web Auth type** select **External(Re-direct to external server)** from drop down.
13. Write URL of the Wifi-Auth eIDAS SP in the **URL** field.
14. Press **Apply** to commit
15. Press **Save Configuration** to make changes permanent.

### 5.3.3 ACL (Access Control List)

ACL is created to allow guest user to access eIDAS nodes without authentication. We have to add ACL for domain names for eIDAS SP, eIDAS Connector and eIDAS IdP. In our configuration ACL we have allowed access to following domain names:

---

```
["wifi-auth-eid4u.polito.it", "connector-test-eid4u.polito.it", "
  identitycl.infocert.it", "connector.pre.eid.gov.it",
"idp-proxy.pre.eid.gov.it", "identity.infocert.it", "connector.eid.gov.it
  ", "posteid.poste.it",
"sipeps-test.gov.si", "sicas.setcce.si", "vidp.gv.at", "test1.a-trust.at",
  "identity.sieltecloud.it"]
```

---

Figure 5.6: Attributes configuration file.

We have to add inbound and outbound rule for each domain name and our ACL looks like this

Creating ACL

1. Go to Security >Access Control List
2. Press **New**
3. Write name of ACL up to 32 characters in **Access Control List Name**

Seq	Action	Source IP/Mask	Destination IP/Mask	Protocol	Source Port	Dest Port	DSCP	Direction
<a href="#">1</a>	Permit	62.241.13.85 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">2</a>	Permit	0.0.0.0 0.0.0.0	/ 62.241.13.85 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">3</a>	Permit	62.241.12.85 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">4</a>	Permit	0.0.0.0 0.0.0.0	/ 62.241.12.85 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">5</a>	Permit	130.192.1.110 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">6</a>	Permit	0.0.0.0 0.0.0.0	/ 130.192.1.110 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">7</a>	Permit	130.192.1.42 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">8</a>	Permit	0.0.0.0 0.0.0.0	/ 130.192.1.42 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">9</a>	Permit	185.102.42.76 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">10</a>	Permit	0.0.0.0 0.0.0.0	/ 185.102.42.76 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">11</a>	Permit	91.206.129.214 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">12</a>	Permit	0.0.0.0 0.0.0.0	/ 91.206.129.214 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">13</a>	Permit	131.1.252.51 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">14</a>	Permit	0.0.0.0 0.0.0.0	/ 131.1.252.51 255.255.255.255	/ Any	Any	Any	Any	Any

Figure 5.7: ACL part 1

<a href="#">15</a>	Permit	185.102.40.105 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">16</a>	Permit	0.0.0.0 0.0.0.0	/ 185.102.40.105 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">17</a>	Permit	131.1.254.240 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">18</a>	Permit	0.0.0.0 0.0.0.0	/ 131.1.254.240 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">19</a>	Permit	88.200.65.135 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">20</a>	Permit	0.0.0.0 0.0.0.0	/ 88.200.65.135 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">21</a>	Permit	88.200.65.141 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">22</a>	Permit	0.0.0.0 0.0.0.0	/ 88.200.65.141 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">23</a>	Permit	93.189.28.37 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">24</a>	Permit	0.0.0.0 0.0.0.0	/ 93.189.28.37 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">25</a>	Permit	213.164.5.202 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">26</a>	Permit	0.0.0.0 0.0.0.0	/ 213.164.5.202 255.255.255.255	/ Any	Any	Any	Any	Any
<a href="#">27</a>	Permit	185.107.185.52 255.255.255.255	/ 0.0.0.0 0.0.0.0	/ Any	Any	Any	Any	Any
<a href="#">28</a>	Permit	0.0.0.0 0.0.0.0	/ 185.107.185.52 255.255.255.255	/ Any	Any	Any	Any	Any

Figure 5.8: ACL part 2

4. Press **Apply** to commit
5. Press **Save Configuration** to make changes permanent.

To allow traffic to flow between and IP, It is required to create two ACL rule per IP. One for outbound traffic and other one for inbound traffic. Creating a rule to allow outbound traffic from IP in ACL

1. Press **Add New Rule**
2. Select **IP Address** from drop down in Source
3. Write IP in **IP Address** and 255.255.255.255 in **Netmask**

4. Select **Outbound** from drop down in Direction
5. Select **Permit** from drop down in Action
6. Press Apply to commit.

Creating a rule to allow inbound traffic from IP in ACL

1. Press **Add New Rule**
2. Select **IP Address** from drop down in Destination
3. Write IP in **IP Address** and 255.255.255.255 in **Netmask**
4. Select **Inbound** from drop down in Direction
5. Select **Permit** from drop down in Action
6. Press Apply to commit.
7. Press **Save Configuration** to make changes permanent.

#### 5.3.4 WebAuth SecureWeb

Disabling HTTPS for WebAuth SecureWeb

1. Go to Management >HTTP-HTTPS
2. Select Disabled in drop down for **WebAuth SecureWeb**
3. Press **Apply** to commit changes
4. Press **Save and Reboot** from COMMANDS >Reboot to reboot WLC.



## Chapter 6

# Results

The requirement for using the deployed service is to have a valid digital national eID from the home country of the user. Depending on the specific country and IdP the authentication schema can be a credential based (username, password), a mobile-based or a digital certificate based authentication. The Wifi-Auth eIDAS-SP allows to authenticate using national eID of all the countries part of eIDAS framework. Table 6.1 shows the list countries and their electronic identity that has been tested successfully for our service.

Country	eID
Italy	SPID
Austria	Buergerkarte
Spain	DNIe
Portugal	Chave Mòvel Digital
Slovenia	National eID card

Table 6.1: eID of countries used for testing the Wifi-Auth eIDAS-SP

We also validated the deployed service and done a survey using the users (students, researchers, entrepreneur) from Turin. We collected data about the IdP, browser and operating system used by the user to access the service.

Identity Provider	Number of users
Aruba	2
InfoCert	1
Poste	14
Sielte	3

Table 6.2: IdPs used by number of users for authentication

Table 6.2 shows the number of users who used specific IdP for authentication. During the test we have to add the IdPs used by users in the ACL to allow access for authentication. In most cases we add IP based ACL rule and in some cases we added URL based ACL. As in the case of authentication using Poste IdP, it provides the ability to authenticate using credential-based schema and mobile-based schema. In case of mobile based schema it uses an another service separate from the Poste IdP, which provide authentication by scanning bar-code from Poste mobile application. The service is hosted at URL [secureholder.mobile.poste.it](https://secureholder.mobile.poste.it) and attached with multiple IP address. For that we have added in the URL based ACL.

Figure 6.1 shows in percentage the IdP used by users for authentication. 70% of the users used Poste for the identification of the electronic identity. Sielte, Aruba and InfoCert are among the other IdPs used by users to authenticate.

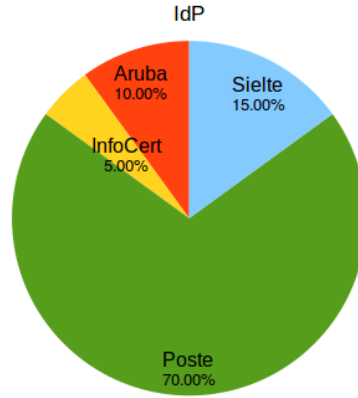
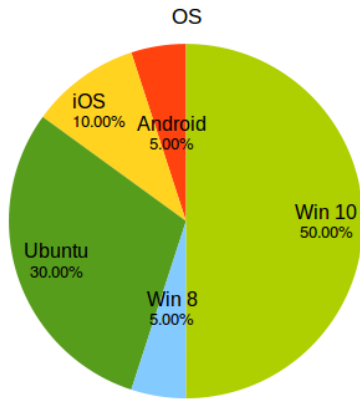
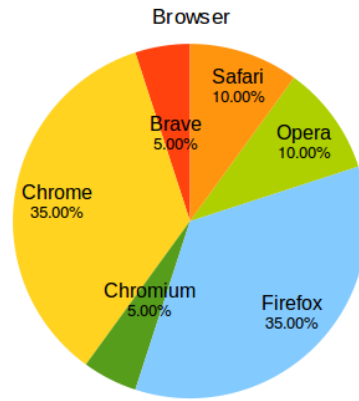


Figure 6.1: IdP used by number of users for authentication

Figure 6.2(a) shows the operating system used by the users. 55% of the users used windows operating system. A part from windows, others operating systems which were used are Ubuntu, iOS and Android.



(a) Operating system



(b) Browser

Figure 6.2: Operating system/browser used by users for authentication

Figure 6.2(b) shows the statistics of the browser used for testing the service. Chrome and Firefox were the browser used by most, 70% of the users one of them. A part from these two, the browser which were used includes Chromium, Brave, Safari and Opera.

We also asked users to fill a survey after testing the service. In Figure 6.3 it shows that more than 97% of the users says that they would want the inclusion of this initiative in other academic services from European universities.

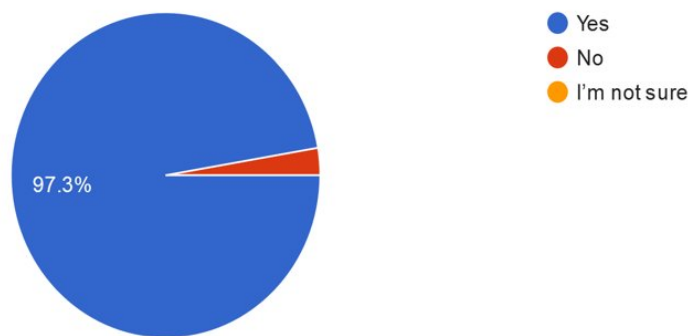


Figure 6.3: Survey Answer: After testing this pilot, I would like the inclusion of this initiative in the academic services of other European universities.

## Chapter 7

# Conclusion

The thesis proposed a solution to provide WiFi access services to European citizens. It was developed as a part of eID4u project: which wants to use the eIDAS infrastructure to provide advanced cross-border services to the European academic environment. We developed two test-beds for the WiFi access service, first using software based approach and second using eIDAS code with Polito wireless infrastructure.

In first approach we used Zeroshell, a Linux based distribution for the development of our service. We configured a WiFi access point to authenticate user's using eIDAS federated infrastructure. The eIDAS federated infrastructure specifies a list of cryptography algorithms for Signing and Encryption of SAML messages to ensure secure communication between endpoints in eIDAS network as described in section 2.5.3. The service developed was able to authenticate using eIDAS infrastructure but with the use of an older version of Encryption algorithm not specified by eIDAS.

In second approach we used eIDAS code to develop Wifi-Auth eIDAS-SP with Polito wireless infrastructure to provide WiFi access. We tested our service successfully using test credentials from various countries including Italy, Spain, Slovenia, Portugal and Austria.

For validation of the solution, we set-up a test-bed in Politecnico di Torino by deploying Wifi-Auth eIDAS-SP on a server and integrate it with the wireless infrastructure implemented in Polito. More than Twenty users tested our service with their national eID using various Identity Providers in the eIDAS infrastructure. After testing the pilot, more than 97 % of the users think that the inclusion of this initiative in academic services of other European universities will be useful.

The solution can be improved by providing a valid certificate for WLC (Wireless LAN Controller) to secure the communication between user's device and WLC after authentication. Also we are adding entities involved in the eIDAS network for authentication manually in the Access Control List (ACL), which also has a limit of maximum 64 IP based and 20 Domain Name based rules.

Finally this service is a validation of the eIDAS infrastructure to provide electronic identification across European countries with the same legal value as paper document. This infrastructure could be used by organisations to provide their services to European citizens. It would unifies the identification/verification process of the users for organisation.

# Bibliography

- [1] L.Florio, K.Wierenga, “Eduroam, providing mobility for roaming users”, Proceedings of the EUNIS 2005 Conference, Manchester (England), June 2005
- [2] eID4U, <https://ec.europa.eu/inea/en/connecting-europe-facility/cef-telecom/2017-eu-ia-0051>
- [3] D.Berbecaru, A.Lioy, C.Cameroni, “Electronic identification for universities: Building cross-border services based on the eIDAS infrastructure”, Information (Switzerland), Vol. 10, No. 210, June 2019, DOI [10.3390/info10060210](https://doi.org/10.3390/info10060210)
- [4] L.O’Gorman, “Comparing passwords, tokens, and biometrics for user authentication”, Proceedings of the IEEE, Vol. 91, No. 12, December 2003, pp. 2021-2040
- [5] J.Reschke, “The ‘Basic’ HTTP Authentication Scheme”, Internet Requests for Comments, RFC-7617, September 2015, DOI [10.17487/RFC7617](https://doi.org/10.17487/RFC7617)
- [6] S.M.Bellovin, M.Merritt, “Encrypted key exchange: Password-based protocols secure against dictionary attacks”, In Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy, May 1992, pp. 72-84
- [7] R.M.Needham, M.D.Schroeder, “Using encryption for authentication in large networks of computers”, Communications of the ACM, Vol. 21, No. 12, December 1978, pp. 993-999 DOI [10.1145/359657.359659](https://doi.org/10.1145/359657.359659)
- [8] J.Clark, J.L.Jacob, “Using encryption for authentication in large networks of computers”, December 1997
- [9] R.Housley, W.Polk, W.Ford, D.Solo, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, RFC-3280, April 2002, DOI [10.17487/RFC3280](https://doi.org/10.17487/RFC3280)
- [10] W.Diffie, M.E.Hellman, “New directions in cryptography”, IEEE transactions on Information Theory, Vol. 21, No. 6, November 1976, pp. 644-654
- [11] R.L.Rivest, A.Shamir, L.Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, Communications of the ACM, Vol. 21, No. 2, February 1978, pp. 120-126
- [12] S.Chokhani, W.Ford, R.Sabett, C.Merrill, S.Wu, “Public Key Infrastructure Certificate Policy and Certification Practices Framework”, Internet Engineering Task Force (IETF), RFC-2527, March 1999, DOI [10.17487/RFC2527](https://doi.org/10.17487/RFC2527)
- [13] J.L.Camp, “Digital identity”, IEEE Technology and society Magazine, October 2004, pp. 34-41
- [14] Y.Cao, L.Yang, “A survey of identity management technology”, 2010 IEEE International Conference on Information Theory and Information Security, December 2010, pp. 287-293
- [15] Security Assertion Markup Language (SAML) V2.0 Technical Overview, <http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html>
- [16] S.Cantor, J.Moreh, R.Philpott, E.Maler, “Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS Standard, March 2005, <http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf>
- [17] T.Bray, J.Paoli, CM.Sperberg-McQueen, E.Maler, F.Yergeau, “Extensible markup language (XML) 1.0”, February 1998, <http://www.renderx.com/~renderx/Demos/fo2html/xml.pdf>
- [18] D.Eastlake, J.Reagle, D.Solo, F.Hirsch, M.Nyström, T.Roessler, K.Yiu, “XML Signature Syntax and Processing”, W3C Recommendation, April 2013, <https://www.w3.org/TR/xmlsig-core/>
- [19] D.Eastlake, J.Reagle, F.Hirsch, T.Roessler, “XML Encryption Syntax and Processing”, W3C Recommendation, April 2013, <https://www.w3.org/TR/xmlenc-core1/>

- [20] R.Fielding, J.Gettys, J.Mogul, H.Frystyk, L.Masinter, P.Leach, T.Berners-Lee, "Hypertext transfer protocol – HTTP/1.1", RFC-2616, June 1999, DOI [10.17487/RFC2068](https://doi.org/10.17487/RFC2068)
- [21] D.Box, D.Ehnebuske, G.Kakivaya, A.Layman, N.Mendelsohn, HF.Nielsen, S.Thatte, D.Winer, "Simple object access protocol (SOAP) 1.1", W3C Note, May 2000, <https://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- [22] S.S.Y.Shim, G.Bhalla, V.Pendyala, "Federated identity management", Computer, Vol. 38, No. 12, December 2005, pp. 120-122
- [23] R.L.Morgan, S.Cantor, S.Carmody, W.Hoehn, K.Klingenstein, "Federated security: The shibboleth approach", Educause Quarterly, Vol. 27, No. 4, 2004, pp. 12-17
- [24] SPID - Sistema Pubblico di Identit  Digitale, <https://www.spid.gov.it/>
- [25] Chave M vel Digital, <https://www.autenticacao.gov.pt/a-chave-movel-digital>
- [26] DNIe - Electronic Identity Card <https://firmaelectronica.gob.es/Home/en/Ciudadanos/DNI-Electronico.html?idioma=en>
- [27] J.L.Hernandez-Ardieta, J.Heppe, J.F.Carvajal-Vion, "STORK: The European electronic identity interoperability platform", IEEE Latin America Transactions, vol. 8, No. 2, July 2010, pp. 190-193
- [28] H.Leitold, A.Lioy, C.Ribeiro, "Stork 2.0: Breaking new grounds on eid and mandates", In Proceedings of ID World International Congress, November 2014, pp. 1-8
- [29] D.Berbecaru, A.Lioy, C.Cameroni, "Providing digital identity and academic attributes through European eID infrastructures: Results achieved, limitations, and future steps", Software: Practice and Experience, Vol. 49, No. 11, November 2019, pp. 1643-1662, DOI [10.1002/spe.2738](https://doi.org/10.1002/spe.2738)
- [30] J.Dumortier, "Regulation (EU) No 910/2014 on Electronic Identification and Trust Services for Electronic Transactions in the Internal Market (eIDAS Regulation)", EU Regulation of E-Commerce, April 2017, pp. 256-289, DOI [10.4337/9781785369346.00017](https://doi.org/10.4337/9781785369346.00017)
- [31] "Electronic identification, signatures and trust services: Questions & Answers", European Commission MEMO, June 2012, [https://ec.europa.eu/commission/presscorner/detail/en/MEMO\\_12\\_403](https://ec.europa.eu/commission/presscorner/detail/en/MEMO_12_403)
- [32] J.Carretero, G.Izquierdo-Moreno, M.Vasile-Cabezas, J.Garcia-Blas, "Federated Identity Architecture of the European eID System", IEEE Access, November 2018, DOI [10.1109/ACCESS.2018.2882870](https://doi.org/10.1109/ACCESS.2018.2882870)
- [33] "Overview of pre-notified and notified eID schemes under eIDAS", <https://ec.europa.eu/cefdigital/wiki/display/EIDCOMMUNITY/Overview+of+pre-notified+and+notified+eID+schemes+under+eIDAS>
- [34] "ON THE APPLICATION OF EIDAS REGULATION", EUROSMART, October 2019
- [35] "eIDAS-Node and SAML 1.0", European Commission, October 2017, pp. 29-34, <https://e-gov.github.io/eIDAS-Connector/MetadataSeletus>
- [36] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC-5246, August 2008, DOI [10.17487/RFC5246](https://doi.org/10.17487/RFC5246)
- [37] "eIDAS - Cryptographic requirements for the Interoperability Framework Version 1.0", November 2015, [https://ec.europa.eu/cefdigital/wiki/download/attachments/82773108/eidas\\_-\\_crypto\\_requirements\\_for\\_the\\_eidas\\_interoperability\\_framework\\_v1.0.pdf?version=1&modificationDate=1497252920224&api=v2](https://ec.europa.eu/cefdigital/wiki/download/attachments/82773108/eidas_-_crypto_requirements_for_the_eidas_interoperability_framework_v1.0.pdf?version=1&modificationDate=1497252920224&api=v2)
- [38] R. Seggelmann, M. Tuexen, M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", Internet Engineering Task Force (IETF), RFC-6520, February 2012, DOI [10.17487/RFC6520](https://doi.org/10.17487/RFC6520)
- [39] D. Eastlake, "Transport Layer Security (TLS) Extensions: Extension Definitions", Internet Engineering Task Force (IETF), RFC-6066, January 2011, DOI [10.17487/RFC6066](https://doi.org/10.17487/RFC6066)
- [40] "Overview of available attributes of pre-notified and notified eID schemes", <https://ec.europa.eu/cefdigital/wiki/display/EIDCOMMUNITY/Overview+of+available+attributes+of+pre-notified+and+notified+eID+schemes>
- [41] "eIDAS: Standardising Digital Identity in the EU", Scribe, <https://www.scribe.com/eidas-electronic-identity-in-the-eu/>
- [42]  .Alonso, A.Pozo, J.Choque, G.Bueno, J.Salvach a, L.Diez, J.Mar n, P.L.C.Alonso, "An Identity Framework for Providing Access to FIWARE OAuth 2.0-Based Services According to the eIDAS European Regulation" IEEE Access 7, July 2019, DOI [10.1109/ACCESS.2019.2926556](https://doi.org/10.1109/ACCESS.2019.2926556)

- [43] TERENA Task Force on Mobility, <https://www.terena.org/activities/tf-mobility/>
- [44] L.Florio, K.Wierenga, “Eduroam, providing mobility for roaming users”, InProceedings of the EUNIS 2005 Conference, Manchester(), June 2005
- [45] M.Milinović, “eduroam Policy Service Definition”, Technischer Bericht, GEANT, July 2012
- [46] “European eduroam Confederation Policy Declaration”, GEANT, May 2012
- [47] eduroam in a nutshell (BEGINNER), <https://wiki.geant.org/pages/viewpage.action?pageId=121346286>
- [48] P.Congdon, B.Aboba, A.Smith, G.Zorn, J.Roose, “IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines”, RFC-3580, September 2003
- [49] C.Rigney, S.Willens, A.Rubens, W.Simpson, “Remote Authentication Dial In User Service (RADIUS)”, IETF, JUNE 2000, DOI [10.17487/RFC2865](https://doi.org/10.17487/RFC2865)
- [50] Govroam Explanation, <https://govroam.nl/english/>
- [51] Govroam NL Service Policy, <https://govroam.nl/wp-content/uploads/2015/02/govroam-NL-service-policy-jan2015.pdf>
- [52] PacketFence Overview, <https://packetfence.org/about.html>
- [53] H.Annuar, B.Shanmugam, A.Ahmad, N.B.Idris, S.H.AlBakri, G.N.Samy, “Enhancement of network access control architecture with virtualization”, International Conference on Informatics and Creative Multimedia (ICICM), September 2013, DOI [10.1109/ICICM.2013.68](https://doi.org/10.1109/ICICM.2013.68)
- [54] PacketFence Advanced Features, <https://packetfence.org/about.html#/features>
- [55] J.Dias, “A guide to microsoft active directory (ad) design”, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), April 2002, DOI [10.1109/SP.2006.4](https://doi.org/10.1109/SP.2006.4)
- [56] NoDogSplash Overview, <https://nodogsplashdocs.readthedocs.io/en/stable/overview.html>
- [57] Zeroshell Firewall Router, <http://www.zeroshell.net/>
- [58] Zeroshell distro, <https://distrowatch.com/table.php?distribution=zeroshell>
- [59] FortigateOS 6.0.0 Cookbook, <http://docs.fortinet.com/document/fortigate/6.0.0/cookbook/509275/getting-started>
- [60] Node.js, <https://nodejs.org/en/>
- [61] TestCafe, <https://devexpress.github.io/testcafe/>
- [62] Certbot, <https://hub.docker.com/r/certbot/certbot/>
- [63] Let’s Encrypt, <https://letsencrypt.org/>
- [64] Zeroshell Linux distribution, <https://zeroshell.org/>
- [65] D.Berbecaru, A.Lioy, M.D.Aime, “Exploiting Proxy-Based Federated Identity Management in Wireless Roaming Access”, In Proceeding of TrustBus 2011: 8th International Conference on Trust, Privacy and Security in Digital Business, Toulouse (France), August-September 2011, Vol. 6863, pp. 13-23, DOI [10.1007/978-3-642-22890-2\\_2](https://doi.org/10.1007/978-3-642-22890-2_2)
- [66] Get Docker Engine - Community for Ubuntu, <https://docs.docker.com/install/linux/docker-ce/ubuntu/>
- [67] J.Callas, L.Donnerhacke, H.Finney, D.Shaw, R.Thayer, “OpenPGP Message Format (RFC 4880)”, Informe técnico, Internet Engineering Task Force (IETF), November 2007, DOI [10.17487/RFC4880](https://doi.org/10.17487/RFC4880)

# Appendix A

## SAML message flow

### A.1 SAML message flow example with Zeroshell

#### A.1.1 Zeroshell to eIDAS-Connector

##### eIDAS Request

---

```
1 <saml2p:AuthnRequest
2   xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
3   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4   xmlns:idas="http://idas.europa.eu/saml-extensions"
5   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
6   AssertionConsumerServiceIndex="1"
7   Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
8   Destination="https://connector-test-eid4u.polito.it/EidasNode/
   ServiceProvider"
9   ForceAuthn="true" ID="_ca4107656fd18b527ae2ebefeeae4596" IsPassive="false"
10  IssueInstant="2019-12-05T11:59:34Z" ProviderName="zeroshell-SP" Version="2.0
   ">
11  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
12    https://zeroshell.example.com:12081/shibboleth</saml2:Issuer>
13  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
14    <ds:SignedInfo>
15      <ds:CanonicalizationMethod
16        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
17      <ds:SignatureMethod
18        Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512" />
19      <ds:Reference URI="#_ca4107656fd18b527ae2ebefeeae4596">
20        <ds:Transforms>
21          <ds:Transform
22            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
          >
23          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
24        </ds:Transforms>
25        <ds:DigestMethod
26          Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha384" />
27        <ds:DigestValue>
28          +3fzna29LgIJGdu7tNSRSbatyvFtZImrXecEiuTV0Apkw4lW3ltKqsy9QN77bjbI
29        </ds:DigestValue>
30      </ds:Reference>
```



---

```

31     </ds:SignedInfo>
32     <ds:SignatureValue>
33         oFK2UCgSCMH0w640Cn3frpIqjEJQN/T4a2QydHfDkD4P8u+OPQGwBvjEjOfA==
34     </ds:SignatureValue>
35     <ds:KeyInfo>
36         <ds:KeyName>SAML Signature</ds:KeyName>
37         <ds:X509Data>
38             <ds:X509SubjectName>CN=SAML Signature,OU=None,O=None,C=IT
39             </ds:X509SubjectName>
40             <ds:X509Certificate>
41                 MIIDKDCCAhCgAwIBAgIJAI57OE1syoDFMAOGCSqbhHSxZlwAzTyHoZfo=
42             </ds:X509Certificate>
43         </ds:X509Data>
44     </ds:KeyInfo>
45 </ds:Signature>
46 <saml2p:Extensions>
47     <eidas:SPTypexmlns:eidas="http://eidas.europa.eu/saml-extensions">public
48 </eidas:SPTypex>
49     <eidas:RequestedAttributes
50         xmlns:eidas="http://eidas.europa.eu/saml-extensions">
51         <eidas:RequestedAttribute FriendlyName="FamilyName"
52             Name="http://eidas.europa.eu/attributes/naturalperson/
53                 CurrentFamilyName"
54             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
55             isRequired="true" />
56         <eidas:RequestedAttribute FriendlyName="FirstName"
57             Name="http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName
58                 "
59             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
60             isRequired="true" />
61         <eidas:RequestedAttribute FriendlyName="DateOfBirth"
62             Name="http://eidas.europa.eu/attributes/naturalperson/DateOfBirth"
63             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
64             isRequired="true" />
65         <eidas:RequestedAttribute FriendlyName="PersonIdentifier"
66             Name="http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier
67                 "
68             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
69             isRequired="true" />
70     </eidas:RequestedAttributes>
71 </saml2p:Extensions>
72 <saml2p:NameIDPolicy AllowCreate="true"
73     Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" />
74 <saml2p:RequestedAuthnContext Comparison="minimum">
75     <saml2:AuthnContextClassRef>http://eidas.europa.eu/LoA/low
76 </saml2:AuthnContextClassRef>
77 </saml2p:RequestedAuthnContext>
78 </saml2p:AuthnRequest>

```

---

## A.1.2 eIDAS-Connector to eIDAS-Service

### eIDAS Request

```

1 <saml2p:AuthnRequest
2   xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
3   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4   xmlns:oidas="http://oidas.europa.eu/saml-extensions"
5   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
6   Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
7   Destination="https://service-test-oid4u.polito.it/EidasNode/ColleagueRequest
8   ForceAuthn="true"
9   ID="_FjbJwldU1NU1AhQy62czTmiZXGC4xveAXQrKxQA86i8bI60FoY2a02CAfmw_T8E"
10  IsPassive="false" IssueInstant="2019-12-05T11:59:34.520Z"
11  ProviderName="zeroshell-SP" Version="2.0">
12  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
13    https://connector-test-oid4u.polito.it/EidasNode/ConnectorMetadata
14  </saml2:Issuer>
15  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
16    <ds:SignedInfo>
17      <ds:CanonicalizationMethod
18        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
19      <ds:SignatureMethod
20        Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
21      <ds:Reference
22        URI="#_FjbJwldU1NU1AhQy62czTmiZXGC4xveAXQrKxQA86i8bI60FoY2a02CAfmw_T8E
23        ">
24        <ds:Transforms>
25          <ds:Transform
26            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
27          >
28            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
29          </ds:Transforms>
30          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" /
31          >
32            <ds:DigestValue>
33              n4yqJJ2dCk9gsIS90IHkYXVGjENpxsCxPtoDKFBjXy1lmNcTEtbgg==
34            </ds:DigestValue>
35          </ds:Reference>
36        </ds:SignedInfo>
37        <ds:SignatureValue>
38          qH1dNUk3y1YXQu8pqSI8DscUhtXv6qg7IrJsYT9eEPWqTwrv/ws0s59x6tIT
39        </ds:SignatureValue>
40        <ds:KeyInfo>
41          <ds:X509Data>
42            <ds:X509Certificate>
43              MIICkTCCAhegAwIBAgIJAMIXa5E22518LbVTpYnHEB8ORHag==
44            </ds:X509Certificate>
45          </ds:X509Data>
46        </ds:KeyInfo>
47      </ds:Signature>
48    <saml2p:Extensions>
49      <oidas:RequestedAttributes>
50        <oidas:RequestedAttribute FriendlyName="FamilyName"
51          Name="http://oidas.europa.eu/attributes/naturalperson/
52          CurrentFamilyName"
53          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
54          isRequired="true" />
55        <oidas:RequestedAttribute FriendlyName="FirstName"

```

---

```

52     Name="http://eidass.europa.eu/attributes/naturalperson/CurrentGivenName
53     "
54     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
55     isRequired="true" />
56     <eidass:RequestedAttribute FriendlyName="DateOfBirth"
57     Name="http://eidass.europa.eu/attributes/naturalperson/DateOfBirth"
58     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
59     isRequired="true" />
60     <eidass:RequestedAttribute FriendlyName="PersonIdentifier"
61     Name="http://eidass.europa.eu/attributes/naturalperson/PersonIdentifier
62     "
63     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
64     isRequired="true" />
65   </eidass:RequestedAttributes>
66 </saml2p:Extensions>
67 <saml2p:NameIDPolicy AllowCreate="true"
68   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" />
69 <saml2p:RequestedAuthnContext Comparison="minimum">
70   <saml2:AuthnContextClassRef>http://eidass.europa.eu/LoA/low
71   </saml2:AuthnContextClassRef>
72 </saml2p:RequestedAuthnContext>
73 </saml2p:AuthnRequest>

```

---

### A.1.3 eIDAS-Service to IdP-Proxy

#### eIDAS Request

---

```

1 <saml2p:AuthnRequest
2   xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
3   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4   xmlns:eidass="http://eidass.europa.eu/saml-extensions"
5   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
6   Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
7   Destination="https://idp-proxy-test-eid4u.polito.it/idproxy/idpeurequest"
8   ForceAuthn="true"
9   ID="_EZpVuMOZd7ZnGsMDB5TpyVfoCI5-tsP.Cylt-4gjHrE1gpMTnp6jpHB2M0zT5g"
10  IsPassive="false" IssueInstant="2019-12-05T11:59:40.046Z"
11  ProviderName="zeroshell-SP" Version="2.0">
12  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
13    https://service-test-eid4u.polito.it/EidasNode/ServiceRequesterMetadata
14  </saml2:Issuer>
15  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
16    <ds:SignedInfo>
17      <ds:CanonicalizationMethod
18        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
19      <ds:SignatureMethod
20        Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
21      <ds:Reference
22        URI="#_EZpVuMOZd7ZnGsMDB5TpyVfoCI5-tsP.Cylt-4gjHrE1gpMTnp6jpHB2M0zT5g"
23        ">
24      <ds:Transforms>
25        <ds:Transform

```

---

```

25         Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
26         >
27         <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
28         </ds:Transforms>
29         <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" /
30         >
31         <ds:DigestValue>
32         TJfCtQFcaBB2+VhTkk9g4L57fRHsB54qfP5pGAACWhZ+UboAmie5Jq8MMQ==
33         </ds:DigestValue>
34         </ds:Reference>
35         </ds:SignedInfo>
36         <ds:SignatureValue>
37         3817n/10nMJ0mETI5xoDtmkFGoibnRb5cSV40xNxj5rzYBKeEf4ntHa6VzWpAS0ThIYfGgK
38         </ds:SignatureValue>
39         <ds:KeyInfo>
40         <ds:X509Data>
41         <ds:X509Certificate>
42         MIICkjCCAhmGAWIBAgIJAPpljjIe0UbmhqZhWe5szLOE=</ds:X509Certificate>
43         </ds:X509Data>
44         </ds:KeyInfo>
45         </ds:Signature>
46         <saml2p:Extensions>
47         <eidas:SPTYPE>public</eidas:SPTYPE>
48         <eidas:RequestedAttributes>
49         <eidas:RequestedAttribute FriendlyName="FamilyName"
50         Name="http://eidas.europa.eu/attributes/naturalperson/
51         CurrentFamilyName"
52         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
53         isRequired="true" />
54         <eidas:RequestedAttribute FriendlyName="FirstName"
55         Name="http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName
56         "
57         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
58         isRequired="true" />
59         <eidas:RequestedAttribute FriendlyName="DateOfBirth"
60         Name="http://eidas.europa.eu/attributes/naturalperson/DateOfBirth"
61         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
62         isRequired="true" />
63         <eidas:RequestedAttribute FriendlyName="PersonIdentifier"
64         Name="http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier
65         "
66         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
67         isRequired="true" />
68         </eidas:RequestedAttributes>
69         </saml2p:Extensions>
70         <saml2p:NameIDPolicy AllowCreate="true"
71         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" />
72         <saml2p:RequestedAuthnContext Comparison="minimum">
73         <saml2:AuthnContextClassRef>http://eidas.europa.eu/LoA/low
74         </saml2:AuthnContextClassRef>
75         </saml2p:RequestedAuthnContext>
76         </saml2p:AuthnRequest>

```

---

### A.1.4 IdP-Proxy to IdP

#### eIDAS Request

---

```

1 <saml2p:AuthnRequest
2   xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
3   AssertionConsumerServiceIndex="0" AttributeConsumingServiceIndex="1"
4   Destination="https://identitycl.infocert.it" ForceAuthn="true"
5   ID="_8584766c-8a1a-4b6b-9e8c-9ef2ec7a0f69"
6   IssueInstant="2019-12-05T11:59:45.137Z" Version="2.0">
7   <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
8     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
9     NameQualifier="https://idp-proxy.test.eid.gov.it/idpproxy">
10    https://idp-proxy-test-eid4u.polito.it/idpproxy</saml2:Issuer>
11   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
12     <ds:SignedInfo>
13       <ds:CanonicalizationMethod
14         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
15       <ds:SignatureMethod
16         Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
17       <ds:Reference URI="#_8584766c-8a1a-4b6b-9e8c-9ef2ec7a0f69">
18         <ds:Transforms>
19           <ds:Transform
20             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
21             >
22             <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
23           </ds:Transforms>
24           <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha256" /
25             >
26             <ds:DigestValue>rFF8heVddt/9QPdycK77aFDpx0ZGslAgWXdrGj7ON+g=
27             </ds:DigestValue>
28           </ds:Reference>
29         </ds:SignedInfo>
30         <ds:SignatureValue>
31           Efltm08NV5/8gUrQPC7mqKUnYFMtBJ+RjkEpdsnKHq1ku39Z1HjaFeoDsNwp6yWcdCfiw==
32         </ds:SignatureValue>
33         <ds:KeyInfo>
34           <ds:X509Data>
35             <ds:X509Certificate>
36               MIIDcDCCAligAwIBAgIJAPTLDfHNBx15ZkMvG3fs/xk=
37             </ds:X509Certificate>
38           </ds:X509Data>
39         </ds:KeyInfo>
40       </ds:Signature>
41     <saml2p:NameIDPolicy
42       Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient" />
43     <saml2p:RequestedAuthnContext Comparison="minimum">
44       <saml2:AuthnContextClassRef
45         xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
46         https://www.spid.gov.it/SpidL1</saml2:AuthnContextClassRef>
47       </saml2:AuthnContextClassRef>
48     </saml2p:RequestedAuthnContext>
49   </ds:Signature>
50 </saml2p:AuthnRequest>

```

---

## A.1.5 IdP to IdP-Proxy

### eIDAS Response

---

```

1  <saml2p:Response
2    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
3    Destination="https://idp-proxy-test-eid4u.polito.it/idpproxy/spidresponse"
4    ID="_3c8e40824598082a3ebe4de11cff5dbf"
5    InResponseTo="_8584766c-8a1a-4b6b-9e8c-9ef2ec7a0f69"
6    IssueInstant="2019-12-05T11:59:51.529Z" Version="2.0">
7    <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
8      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
9      https://identitycl.infocert.it</saml2:Issuer>
10   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
11     <SignedInfo>
12       <CanonicalizationMethod
13         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
14       <SignatureMethod
15         Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
16       <Reference URI="#_3c8e40824598082a3ebe4de11cff5dbf">
17         <Transforms>
18           <Transform
19             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
20             >
21           <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
22         </Transforms>
23         <DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha256" />
24         <DigestValue>S6/v6uUp3vXajKbzKIqLMrAKkEh938ybfCUfd1VDG2Y=</DigestValue
25         >
26       </Reference>
27     </SignedInfo>
28     <SignatureValue>
29       ZSNMYM4gQzWbAHRisWhSiDLQswkX9X6bjnv8++6t0JVfapDizNy8zIVSH8UBtg==
30     </SignatureValue>
31     <KeyInfo>
32       <X509Data>
33         <X509Certificate>
34           MIIGbjCCBVagAwIBAgIDFI91MAOGCSqGSIb3DQEBCwUAMIGG3pdz+AzeNQ92mtk
35         </X509Certificate>
36       </X509Data>
37     </KeyInfo>
38   </Signature>
39   <saml2p:Status>
40     <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
41   </saml2p:Status>
42   <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
43     ID="_9c2c3e33ffaec0f91368fadae2b18286"
44     IssueInstant="2019-12-05T11:59:51.529Z" Version="2.0">
45     <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
46       https://identitycl.infocert.it</saml2:Issuer>
47     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
48       <SignedInfo>
49         <CanonicalizationMethod
50           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
51         <SignatureMethod
52           Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />

```

```

51     <Reference URI="#_9c2c3e33ffaec0f91368fadae2b18286">
52         <Transforms>
53             <Transform
54                 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"
55                 />
56             <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
57         </Transforms>
58         <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
59         <DigestValue>KYYVLv0NH16Sbdr5ma1QHt5pkZOHwFy6CERRM7oC5wI=
60         </DigestValue>
61     </Reference>
62 </SignedInfo>
63 <SignatureValue>
64     DLB1EU1apuSaktgCG4p1YUP6Hor8x72997KuVXEU1/c6stX9109I8hU1mZcD1ysw==
65 </SignatureValue>
66 <KeyInfo>
67     <X509Data>
68         <X509Certificate>
69             MIIGbjCCBVagAwIBAgIDFI91MAOGCSqGSIb3DQEBAzAeNQ92mtk
70         </X509Certificate>
71     </X509Data>
72 </KeyInfo>
73 </Signature>
74 <saml2:Subject>
75     <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
76         format:transient"
77         NameQualifier="https://identitycl.infocert.it">
78         _9bc06745b840b6844dc1567fa52dd4ca</saml2:NameID>
79     <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer
80         ">
81         <saml2:SubjectConfirmationData
82             InResponseTo="_8584766c-8a1a-4b6b-9e8c-9ef2ec7a0f69"
83             NotOnOrAfter="2019-12-05T12:00:21.529Z"
84             Recipient="https://idp-proxy-test-eid4u.polito.it/idpproxy/
85                 spidresponse" />
86     </saml2:SubjectConfirmation>
87 </saml2:Subject>
88 <saml2:Conditions NotBefore="2019-12-05T11:59:50.529Z"
89     NotOnOrAfter="2019-12-05T12:00:21.529Z">
90     <saml2:AudienceRestriction>
91         <saml2:Audience>https://idp-proxy-test-eid4u.polito.it/idpproxy
92         </saml2:Audience>
93     </saml2:AudienceRestriction>
94 </saml2:Conditions>
95 <saml2:AuthnStatement AuthnInstant="2019-12-05T11:59:51.529Z"
96     SessionIndex="_9e00984bef4aa4d02f8ba5f6687d3143"
97     SessionNotOnOrAfter="2019-12-05T12:29:51.529Z">
98     <saml2:AuthnContext>
99         <saml2:AuthnContextClassRef>https://www.spid.gov.it/SpidL1
100         </saml2:AuthnContextClassRef>
101     </saml2:AuthnContext>
102 </saml2:AuthnStatement>
103 <saml2:AttributeStatement>
104     <saml2:Attribute FriendlyName="Codice identificativo SPID"
105         Name="spidCode">
106     <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
107         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

```

```

104         xsi:type="xs:string">INFC0001TESTEU</saml2:AttributeValue>
105     </saml2:Attribute>
106     <saml2:Attribute FriendlyName="Nome" Name="name">
107         <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
108             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
109             xsi:type="xs:string">Arianna</saml2:AttributeValue>
110     </saml2:Attribute>
111     <saml2:Attribute FriendlyName="Cognome" Name="familyName">
112         <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
113             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
114             xsi:type="xs:string">Garbini</saml2:AttributeValue>
115     </saml2:Attribute>
116     <saml2:Attribute FriendlyName="Data di nascita" Name="dateOfBirth">
117         <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
118             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
119             xsi:type="xs:date">1968-05-22</saml2:AttributeValue>
120     </saml2:Attribute>
121 </saml2:AttributeStatement>
122 </saml2:Assertion>
123 </saml2p:Response>

```

### A.1.6 IdP-Proxy to eIDAS-Service

#### eIDAS Response

```

1 <saml2p:Response
2   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3   xmlns:eidas="http://eidas.europa.eu/attributes/naturalperson"
4   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
5   xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
6   Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
7   Destination="https://service-test-eid4u.polito.it/EidasNode/IdpResponse"
8   ID="_ZQzG0YhpTqMUJbWTxsQ33h8KuZHpL6oapTXjV0kh282Rb1vKAuiM1bcyp5ed2T8"
9   InResponseTo="_.EZpVuM0Zd7ZnGsMDB5TpyVfoCI5-tsP.Cylt-4
   gjHrE1gpMTnp6jpHB2M0zT5g"
10  IssueInstant="2019-12-05T11:59:52.038Z" Version="2.0">
11    <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
12      https://idp-proxy-test-eid4u.polito.it/idpproxy/idpeumetadata/</
        saml2:Issuer>
13    <ds:Signature>
14      <ds:SignedInfo>
15        <ds:CanonicalizationMethod
16          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
17        <ds:SignatureMethod
18          Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
19        <ds:Reference
20          URI="#_ZQzG0YhpTqMUJbWTxsQ33h8KuZHpL6oapTXjV0kh282Rb1vKAuiM1bcyp5ed2T8"
          >
21        <ds:Transforms>
22          <ds:Transform
23            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
24          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />

```



```

25     </ds:Transforms>
26     <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" /
27     >
28     <ds:DigestValue>
29         1RodIi87qEvmZRI0Ysd24hojAOFBM4Ki+DY6Y1o9RVKXerEHOPH0uUv3Lw==
30     </ds:DigestValue>
31 </ds:Reference>
32 <ds:SignatureValue>
33     UalcLx/Pw8BgbsZrM7m2Da+Cfedd7iS3ah2Y1Lq57NTF13/6gNgR4B09o
34 </ds:SignatureValue>
35 <ds:KeyInfo>
36     <ds:X509Data>
37         <ds:X509Certificate>
38             MIICKzCCAbKgAwIBAgIULN+1E5fV1h5DCYZIUA/X1jB04NKgd5LacxXfSAA==
39         </ds:X509Certificate>
40     </ds:X509Data>
41 </ds:KeyInfo>
42 </ds:Signature>
43 <saml2p:Status>
44     <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
45     <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success
46 </saml2p:StatusMessage>
47 </saml2p:Status>
48 <saml2:EncryptedAssertion>
49     <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
50     Id="_9edfcba8f58bfdc4113781a84f15cfc7"
51     Type="http://www.w3.org/2001/04/xmlenc#Element">
52     <xenc:EncryptionMethod
53     Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm" />
54     <ds:KeyInfo>
55         <xenc:EncryptedKey Id="_f8ee973cef7ae93d8ec4d44356891aba">
56             <xenc:EncryptionMethod
57             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
58             <ds:DigestMethod
59             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
60             </xenc:EncryptionMethod>
61             <ds:KeyInfo>
62                 <ds:X509Data>
63                     <ds:X509Certificate>
64                         MIIIF5DCCA8ygAwIBAgIJ/jp10A33sCi6SZKgr9Q2Fzp3HkLQF</
65                         ds:X509Certificate>
66                     </ds:X509Data>
67                 </ds:KeyInfo>
68             <xenc:CipherData>
69                 <xenc:CipherValue>
70                     oH3ZPDFzj2lY3m0adg5BhReSUCdEt2wXDuJ9lo9ArdNu3seemV3cQ=
71                 </xenc:CipherValue>
72             </xenc:CipherData>
73             </xenc:EncryptedKey>
74         </ds:KeyInfo>
75     <xenc:CipherData>
76         <xenc:CipherValue>
77             RTkNNq5nSxSttWv3mTNMcJYmWrj5PV9Tdkm4xbGESvGJYdkEz00
78         </xenc:CipherValue>
79     </xenc:CipherData>
80 </xenc:EncryptedData>

```

```

80   </saml2:EncryptedAssertion>
81 </saml2p:Response>

```

## A.1.7 eIDAS-Service to eIDAS-Connector

### eIDAS Response

```

1  <saml2p:Response
2    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
3    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4    xmlns:eidas="http://eidas.europa.eu/attributes/naturalperson"
5    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
6    Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
7    Destination="https://connector-test-eid4u.polito.it/EidasNode/
      ColleagueResponse"
8    ID="_QTFr0xaKSBn9GXDKsSbaeDN7uAHDgYsvxgusylX2YsP7bVsv0.xEk-5k6Uv5DvC"
9    InResponseTo="
      _FbjbwldU1NU1AhQy62czTmiZXGC4xveAXQrxxQA86i8bI60FoY2a02CAfmw_T8E"
10   IssueInstant="2019-12-05T11:59:53.014Z" Version="2.0">
11   <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
12     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
13     https://service-test-eid4u.polito.it/EidasNode/ServiceMetadata
14   </saml2:Issuer>
15   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
16     <ds:SignedInfo>
17       <ds:CanonicalizationMethod
18         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
19       <ds:SignatureMethod
20         Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
21       <ds:Reference
22         URI="#_QTFr0xaKSBn9GXDKsSbaeDN7uAHDgYsvxgusylX2YsP7bVsv0.xEk-5k6Uv5DvC
23         ">
24         <ds:Transforms>
25           <ds:Transform
26             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
27         </ds:Transforms>
28         <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" />
29         <ds:DigestValue>
30           DXXdUY2ulkd0HYH8gug5FgR+R80e1yEHuY44vm21i7+Qz3IQST9wzmA==
31         </ds:DigestValue>
32       </ds:Reference>
33     </ds:SignedInfo>
34     <ds:SignatureValue>
35       KOEyyjKl/YngVVZZLZ+wo12Uxn+9J7MAEWfdd4XOPVBikRFL/tIuVr8nLmRIi
36     </ds:SignatureValue>
37     <ds:KeyInfo>
38       <ds:X509Data>
39         <ds:X509Certificate>
40           MIICkjCCAhmGAwIBAgIJD/kZEHLTjIeOUbmhqZhWe5szL0E=</ds:X509Certificate>
41       </ds:X509Data>

```

---

```

42     </ds:KeyInfo>
43 </ds:Signature>
44 <saml2p:Status xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
45   <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
46   <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success
47 </saml2p:StatusMessage>
48 </saml2p:Status>
49 <saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion
50   ">
51   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
52     Id="_ca63412f5d01634d007beed3021b48e5"
53     Type="http://www.w3.org/2001/04/xmlenc#Element">
54     <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
55       Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm" />
56     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
57       <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
58         Id="_e2d5815d15fff5d0ba4726bffd9f326d">
59         <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
60           Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
61         <ds:DigestMethod xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
62           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
63         </xenc:EncryptionMethod>
64         <ds:KeyInfo>
65           <ds:X509Data>
66             <ds:X509Certificate>
67               MIIF4jCCA8qgAwIBAgIJAMlkjGOLNh9TEekfKjQTA==</ds:X509Certificate
68             >
69             </ds:X509Data>
70             </ds:KeyInfo>
71             <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
72               <xenc:CipherValue>
73                 Le4qm56nQ7vAFYalDCCQl+oSWte71RtcSkmRIe4/mEP8LjvvrnI=
74               </xenc:CipherValue>
75             </xenc:CipherData>
76             </xenc:EncryptedKey>
77           </ds:KeyInfo>
78           <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
79             <xenc:CipherValue>
80               wwdgMtOmV63VwnOCNsolufalAVt6ePHxBuQyKcsOVumD4m+etMIj93bs=
81             </xenc:CipherValue>
82           </xenc:CipherData>
83         </xenc:EncryptedData>
84       </saml2:EncryptedAssertion>
85     </saml2p:Response>

```

---

### A.1.8 eIDAS-Connector to Zeroshell

#### eIDAS Response

---

```

1 <saml2p:Response
2   xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
3   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4   xmlns:eidas="http://eidas.europa.eu/attributes/naturalperson"

```

```

5   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
6   Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
7   Destination="https://zeroshell.example.com:12081/Shibboleth.sso/SAML2/POST"
8   ID="_eU0wabdGEH00WQzrkxyQcBCYGiuKRGzLMh8eJPfRlZbH_dFACIoV3c3-00YcSq6"
9   InResponseTo="_ca4107656fd18b527ae2ebefeeae4596"
10  IssueInstant="2019-12-05T12:00:08.907Z" Version="2.0">
11  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
12    Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
13    https://connector-test-eid4u.polito.it/EidasNode/
14    ConnectorResponderMetadata
15  </saml2:Issuer>
16  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
17    <ds:SignedInfo>
18      <ds:CanonicalizationMethod
19        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
20      <ds:SignatureMethod
21        Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
22      <ds:Reference
23        URI="#_eU0wabdGEH00WQzrkxyQcBCYGiuKRGzLMh8eJPfRlZbH_dFACIoV3c3-00YcSq6"
24        ">
25      <ds:Transforms>
26        <ds:Transform
27          Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
28        >
29        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
30      </ds:Transforms>
31      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" /
32      >
33      <ds:DigestValue>
34        PaAqtga4al15ZLj3spWntEPiWPSi0hPKQPqaEkR1k6rcMDnV9bddzVt1DjcQ==
35      </ds:DigestValue>
36    </ds:Reference>
37  </ds:SignedInfo>
38  <ds:SignatureValue>
39    97SNue/47xd11dT/XmEQJuSENSFHWDSc4MbFjHBmA0HSg+MYTTboFZa/FK
40  </ds:SignatureValue>
41  <ds:KeyInfo>
42    <ds:X509Data>
43      <ds:X509Certificate>
44        MIICkTCCAhegAwIBAgIJ5xIKP225l8LbVTpYnHEB80RHag==</ds:X509Certificate>
45      </ds:X509Data>
46    </ds:KeyInfo>
47  </ds:Signature>
48  <saml2p:Status xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
49    <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
50    <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success
51  </saml2p:StatusMessage>
52  </saml2p:Status>
53  <saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
54    ">
55    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
56    Id="_20202c9bc18ec474dcb604bf966d6a3a"
57    Type="http://www.w3.org/2001/04/xmlenc#Element">
58      <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
59        Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
60      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
61        <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"

```

---

```

57     Id="_c84fc55617c1200daeabf0349bcd629">
58     <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
59       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
60       <ds:DigestMethod xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
61         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
62     </xenc:EncryptionMethod>
63     <ds:KeyInfo>
64       <ds:X509Data>
65         <ds:X509Certificate>
66           MIIFKjCCAxKgAwIBAgIJeXgwLVUVCTjEeNOGZ</ds:X509Certificate>
67       </ds:X509Data>
68     </ds:KeyInfo>
69     <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
70       <xenc:CipherValue>
71         QXsLaffQwx6M3JMLne94nFQb2GD+Cb5MGgj5EpRvmXQJBodG9yIEHmy5c=
72       </xenc:CipherValue>
73     </xenc:CipherData>
74   </xenc:EncryptedKey>
75 </ds:KeyInfo>
76 <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
77   <xenc:CipherValue>
78     IjZ0ZTpNBvV1aIHh/574jnsUJeu1gpYQUEYzlC2CRC0zmrJg/zkiMHC1eAMCn/Qa/7
79     Kt4mFFg/TA=
80   </xenc:CipherValue>
81 </xenc:CipherData>
82 </xenc:EncryptedData>
83 </saml2:EncryptedAssertion>
84 </saml2p:Response>

```

---

## A.2 SAML message flow example with Wifi-Auth eIDAS-SP

### A.2.1 Wifi-Auth eIDAS-SP to eIDAS-Connector

#### eIDAS Request

---

```

1 <saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
2   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3   xmlns:eidas="http://eidas.europa.eu/saml-extensions"
4   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
5   Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
6   Destination="https://connector-test-eid4u.polito.it/EidasNode/
7     ServiceProvider"
7   ForceAuthn="true"
8   ID="_sRmCiBCyS21tAgWrQeJcpWr.PefcyCiGQbsDvjHE5xn27Wxhh0uqF1.anRGgPbQ"
9   IsPassive="false" IssueInstant="2019-12-05T12:40:44.725Z"
10  ProviderName="WIFI-AUTH" Version="2.0">
11 <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
12   https://wifi-auth-eid4u.polito.it/SP/metadata</saml2:Issuer>
13 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
14   <ds:SignedInfo>
15     <ds:CanonicalizationMethod

```

```

16     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
17   <ds:SignatureMethod
18     Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
19   <ds:Reference
20     URI="#_sRmCiBCyS21tAgWrQeJcpWr.PefcyCiGQbsDvjHE5xn27Wxhh0uqF1.anRGGpBq
21     ">
22     <ds:Transforms>
23       <ds:Transform
24         Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
25       >
26       <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
27     </ds:Transforms>
28     <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" /
29     >
30     <ds:DigestValue>
31       wTkyL8TbnF32cn8pL7tVVuLjXLYV4iYP13rJJ
32     </ds:DigestValue>
33   </ds:Reference>
34 </ds:SignedInfo>
35 <ds:SignatureValue>
36   bCsn2tlq2bXDu9ElLd31ddxU8Dbexqv5aj9KBDOLY
37 </ds:SignatureValue>
38 <ds:KeyInfo>
39   <ds:X509Data>
40     <ds:X509Certificate>
41       MIICXzCCAeagAwIBAgIUNBxE8ZRQ/b0v83di7pXzA2IzaQxN92qYyC+o0Ikw=
42     </ds:X509Certificate>
43   </ds:X509Data>
44 </ds:KeyInfo>
45 </ds:Signature>
46 <saml2p:Extensions>
47   <eidas:SPTYPE>public</eidas:SPTYPE>
48   <eidas:RequestedAttributes>
49     <eidas:RequestedAttribute FriendlyName="FamilyName"
50       Name="http://eidas.europa.eu/attributes/naturalperson/
51       CurrentFamilyName"
52       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
53       isRequired="true" />
54     <eidas:RequestedAttribute FriendlyName="FirstName"
55       Name="http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName
56       "
57       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
58       isRequired="true" />
59     <eidas:RequestedAttribute FriendlyName="DateOfBirth"
60       Name="http://eidas.europa.eu/attributes/naturalperson/DateOfBirth"
61       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
62       isRequired="true" />
63     <eidas:RequestedAttribute FriendlyName="PersonIdentifier"
64       Name="http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier
65       "
66       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
67       isRequired="true" />
68   </eidas:RequestedAttributes>
69 </saml2p:Extensions>
70 <saml2p:NameIDPolicy AllowCreate="true"
71   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" />
72 <saml2p:RequestedAuthnContext Comparison="minimum">

```

---

```

67     <saml2:AuthnContextClassRef>http://eidas.europa.eu/LoA/low
68   </saml2:AuthnContextClassRef>
69   </saml2p:RequestedAuthnContext>
70 </saml2p:AuthnRequest>

```

---

## A.2.2 eIDAS-Connector to eIDAS-Service

### eIDAS Request

---

```

1  <saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
2    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3    xmlns:eidas="http://eidas.europa.eu/saml-extensions"
4    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
5    Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
6    Destination="https://service-test-eid4u.polito.it/EidasNode/ColleagueRequest
7      "
8    ForceAuthn="true"
9    ID="_882EW-EpjxyDF_YTH00wOKU1FYNPzhXmurz31yHJ6N0Cht.drOpZaMY03t1suh6"
10   IsPassive="false" IssueInstant="2019-12-05T12:40:46.796Z"
11   ProviderName="WIFI-AUTH" Version="2.0">
12   <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
13     https://connector-test-eid4u.polito.it/EidasNode/ConnectorMetadata
14   </saml2:Issuer>
15   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
16     <ds:SignedInfo>
17       <ds:CanonicalizationMethod
18         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
19       <ds:SignatureMethod
20         Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
21       <ds:Reference
22         URI="#_882EW-EpjxyDF_YTH00wOKU1FYNPzhXmurz31yHJ6N0Cht.drOpZaMY03t1suh6
23         ">
24         <ds:Transforms>
25           <ds:Transform
26             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
27           >
28             <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
29           </ds:Transforms>
30           <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" /
31           >
32             <ds:DigestValue>
33               UZ6y1xmE5UaAbjQLttZlu+ypdlMdXF7SYcFEkcbV2BZYh+
34             </ds:DigestValue>
35           </ds:Reference>
36         </ds:SignedInfo>
37         <ds:SignatureValue>
38           ugStb0y0s4yoMpbwsINurQnKaYMGsDzXC67sJSPUPjcX
39         </ds:SignatureValue>
40         <ds:KeyInfo>
41           <ds:X509Data>
42             <ds:X509Certificate>
43               MIICkTCCAhegAwIBAgIJAMIXa5EPQZdFMAoGbVTpYnHEB80RHag==
44             </ds:X509Certificate>

```

---

```

41     </ds:X509Data>
42     </ds:KeyInfo>
43 </ds:Signature>
44 <saml2p:Extensions>
45   <eidas:RequestedAttributes>
46     <eidas:RequestedAttribute FriendlyName="FamilyName"
47       Name="http://eidas.europa.eu/attributes/naturalperson/
         CurrentFamilyName"
48       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
49       isRequired="true" />
50     <eidas:RequestedAttribute FriendlyName="FirstName"
51       Name="http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName
         "
52       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
53       isRequired="true" />
54     <eidas:RequestedAttribute FriendlyName="DateOfBirth"
55       Name="http://eidas.europa.eu/attributes/naturalperson/DateOfBirth"
56       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
57       isRequired="true" />
58     <eidas:RequestedAttribute FriendlyName="PersonIdentifier"
59       Name="http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier
         "
60       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
61       isRequired="true" />
62   </eidas:RequestedAttributes>
63 </saml2p:Extensions>
64 <saml2p:NameIDPolicy AllowCreate="true"
65   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" />
66 <saml2p:RequestedAuthnContext Comparison="minimum">
67   <saml2:AuthnContextClassRef>http://eidas.europa.eu/LoA/low
68   </saml2:AuthnContextClassRef>
69 </saml2p:RequestedAuthnContext>
70 </saml2p:AuthnRequest>

```

---

### A.2.3 eIDAS-Service to IdP-Proxy

#### eIDAS Request

---

```

1 <saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
2   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3   xmlns:eidas="http://eidas.europa.eu/saml-extensions"
4   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
5   Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
6   Destination="https://idp-proxy-test-eid4u.polito.it/idproxy/idpeurequest"
7   ForceAuthn="true"
8   ID="_-7T6xlFmk0z71P03Kl9vKLYXl6ZVyaSw.QSHmLrmnFwKEJiUymd-HmX3ELnRBVL"
9   IsPassive="false" IssueInstant="2019-12-05T12:40:56.889Z"
10  ProviderName="WIFI-AUTH" Version="2.0">
11   <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
12     https://service-test-eid4u.polito.it/EidasNode/ServiceRequesterMetadata
13   </saml2:Issuer>
14   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
15     <ds:SignedInfo>

```



```

16     <ds:CanonicalizationMethod
17       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
18     <ds:SignatureMethod
19       Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
20     <ds:Reference
21       URI="#_ -7T6xlFmkoz71P03K19vKLYX16ZVyaSw.QSHmLrmnFwKEJiUymd-HmX3ELnRBVL
22         ">
23       <ds:Transforms>
24         <ds:Transform
25           Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
26         >
27         <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
28       </ds:Transforms>
29       <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" /
30       >
31       <ds:DigestValue>
32         DiiQHHPZemttkPc25niwtQ6YScGi/q8S8WadVAJH0hNe
33       </ds:DigestValue>
34     </ds:Reference>
35   </ds:SignedInfo>
36   <ds:SignatureValue>
37     z8y/gjXEOI4dutsqGjqVmX10nc1Ed3dv4+cqhA1kCptVOh1OCJgQ7
38   </ds:SignatureValue>
39   <ds:KeyInfo>
40     <ds:X509Data>
41       <ds:X509Certificate>
42         MIICKjCCAhmGAWIBAgIJAPpljANQoDs1MAoGCCqGSM49BbmhqZhWe5szLOE=
43       </ds:X509Certificate>
44     </ds:X509Data>
45   </ds:KeyInfo>
46 </ds:Signature>
47 <saml2p:Extensions>
48   <eidas:SPTYPE>public</eidas:SPTYPE>
49   <eidas:RequestedAttributes>
50     <eidas:RequestedAttribute FriendlyName="FamilyName"
51       Name="http://eidas.europa.eu/attributes/naturalperson/
52       CurrentFamilyName"
53       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
54       isRequired="true" />
55     <eidas:RequestedAttribute FriendlyName="FirstName"
56       Name="http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName
57       "
58       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
59       isRequired="true" />
60     <eidas:RequestedAttribute FriendlyName="DateOfBirth"
61       Name="http://eidas.europa.eu/attributes/naturalperson/DateOfBirth"
62       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
63       isRequired="true" />
64     <eidas:RequestedAttribute FriendlyName="PersonIdentifier"
65       Name="http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier
66       "
67       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
68       isRequired="true" />
69   </eidas:RequestedAttributes>
70 </saml2p:Extensions>
71 <saml2p:NameIDPolicy AllowCreate="true"
72   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" />

```

---

```

67   <saml2p:RequestedAuthnContext Comparison="minimum">
68     <saml2:AuthnContextClassRef>http://eid.as.europa.eu/LoA/low
69   </saml2:AuthnContextClassRef>
70 </saml2p:RequestedAuthnContext>
71 </saml2p:AuthnRequest>

```

---

## A.2.4 IdP-Proxy to IdP

### eIDAS Request

---

```

1  <saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
2    AssertionConsumerServiceIndex="0" AttributeConsumingServiceIndex="1"
3    Destination="https://identitycl.infocert.it" ForceAuthn="true"
4    ID="_7861263d-c009-4ea1-b39e-bf041148ed92"
5    IssueInstant="2019-12-05T12:41:04.283Z" Version="2.0">
6    <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
7      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
8      NameQualifier="https://idp-proxy.test.eid.gov.it/idpproxy">
9      https://idp-proxy-test-eid4u.polito.it/idpproxy</saml2:Issuer>
10   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
11     <ds:SignedInfo>
12       <ds:CanonicalizationMethod
13         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
14       <ds:SignatureMethod
15         Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
16       <ds:Reference URI="#_7861263d-c009-4ea1-b39e-bf041148ed92">
17         <ds:Transforms>
18           <ds:Transform
19             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
20             >
21             <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
22           </ds:Transforms>
23           <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha256" /
24             >
25             <ds:DigestValue>5oEsIZbMjN67wEAY3zNAaQ/oYaPFLef3Laih+Ac1GEI=
26             </ds:DigestValue>
27           </ds:Reference>
28         </ds:SignedInfo>
29         <ds:SignatureValue>
30           h0S1DDSmQ+rcYPcgz+z41NU1pGYQ3dj0cSovMzcoHOohdkrb
31         </ds:SignatureValue>
32         <ds:KeyInfo>
33           <ds:X509Data>
34             <ds:X509Certificate>
35               MIIDcDCCAligAwIBAgIJAPTLDfHNB0uRMAOGCSqGG3fs/xk=
36             </ds:X509Certificate>
37           </ds:X509Data>
38         </ds:KeyInfo>
39       </ds:Signature>
40     </saml2p:RequestedAuthnContext Comparison="minimum">
41     <saml2:AuthnContextClassRef

```

---

```

42     xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
43     https://www.spid.gov.it/SpidL1</saml2:AuthnContextClassRef>
44 </saml2p:RequestedAuthnContext>
45 </saml2p:AuthnRequest>

```

---

## A.2.5 IdP to IdP-Proxy

### eIDAS Response

---

```

1 <saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
2   Destination="https://idp-proxy-test-eid4u.polito.it/idpproxy/spidresponse"
3   ID="_cedfaf7594c9aef38cecafaf593cb125"
4   InResponseTo="7861263d-c009-4ea1-b39e-bf041148ed92"
5   IssueInstant="2019-12-05T12:41:11.698Z" Version="2.0">
6   <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
7     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
8     https://identitycl.infocert.it</saml2:Issuer>
9   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
10     <SignedInfo>
11       <CanonicalizationMethod
12         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
13       <SignatureMethod
14         Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
15       <Reference URI="#_cedfaf7594c9aef38cecafaf593cb125">
16         <Transforms>
17           <Transform
18             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
19             >
20             <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
21           </Transforms>
22           <DigestMethod Algorithm="http://www.w3.org/2001/04/xmenc#sha256" />
23           <DigestValue>sgYxg5mKk9XS44uyyQQGcXESxv2eMpnqN6RNb3Lh8RI=</DigestValue
24             >
25         </Reference>
26       </SignedInfo>
27       <SignatureValue>
28         JpUZ+lbSgcTpaVhHsz2h96DiG0ecq6c1tpa0QZwaoXpM7im
29       </SignatureValue>
30       <KeyInfo>
31         <X509Data>
32           <X509Certificate>
33             MIIGbjCCBVagAwIBAgIDFI91MAOGCSqGSIb3DQEBCwUAMI+AzeNQ92mtk
34           </X509Certificate>
35         </X509Data>
36       </KeyInfo>
37     </Signature>
38   <saml2p:Status>
39     <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
40   </saml2p:Status>
41   <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
42     ID="_98fda83df8ae11e7110bcd55616085a1"
43     IssueInstant="2019-12-05T12:41:11.698Z" Version="2.0">
44     <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">

```

```

43     https://identitycl.infocert.it</saml2:Issuer>
44 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
45   <SignedInfo>
46     <CanonicalizationMethod
47       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
48     <SignatureMethod
49       Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
50     <Reference URI="#_98fda83df8ae11e7110bcd55616085a1">
51       <Transforms>
52         <Transform
53           Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"
54           />
55         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
56       </Transforms>
57       <DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha256" />
58       <DigestValue>NlXYPeY2JqjdEWL0jYx/Unu1s3BafBBHtT4ikDjQ0XM=
59     </DigestValue>
60   </Reference>
61 </SignedInfo>
62 <SignatureValue>
63   pSmaQwbFC/VhOhkzNnbbPOmjjIn7T39GaFS6/6ujmfv2jTQlouFg6
64 </SignatureValue>
65 <KeyInfo>
66   <X509Data>
67     <X509Certificate>
68       MIIGbjCCBVagAwIBAgIDFI91MAOGCSqGSIb3DQEBCwUAMIGGMQ+AzeNQ92mtk
69     </X509Certificate>
70   </X509Data>
71 </KeyInfo>
72 </Signature>
73 <saml2:Subject>
74   <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
75     format:transient"
76     NameQualifier="https://identitycl.infocert.it">
77     _3bee572907e174f4b4e34e5dca106b61</saml2:NameID>
78   <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"
79     ">
80     <saml2:SubjectConfirmationData
81       InResponseTo="_7861263d-c009-4ea1-b39e-bf041148ed92"
82       NotOnOrAfter="2019-12-05T12:41:41.698Z"
83       Recipient="https://idp-proxy-test-eid4u.polito.it/idpproxy/
84         spidresponse" />
85   </saml2:SubjectConfirmation>
86 </saml2:Subject>
87 <saml2:Conditions NotBefore="2019-12-05T12:41:10.698Z"
88   NotOnOrAfter="2019-12-05T12:41:41.698Z">
89   <saml2:AudienceRestriction>
90     <saml2:Audience>https://idp-proxy-test-eid4u.polito.it/idpproxy
91   </saml2:Audience>
92 </saml2:AudienceRestriction>
93 </saml2:Conditions>
94 <saml2:AuthnStatement AuthnInstant="2019-12-05T12:41:11.698Z"
95   SessionIndex="_8f8cb76addf6c22d1ea84da9457a7ca3"
96   SessionNotOnOrAfter="2019-12-05T13:11:11.698Z">
97   <saml2:AuthnContext>
98     <saml2:AuthnContextClassRef>https://www.spid.gov.it/SpidL1
99   </saml2:AuthnContextClassRef>

```

```

96     </saml2:AuthnContext>
97 </saml2:AuthnStatement>
98 <saml2:AttributeStatement>
99     <saml2:Attribute FriendlyName="Codice identificativo SPID"
100       Name="spidCode">
101       <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
102         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
103         xsi:type="xs:string">INF00001TESTEU</saml2:AttributeValue>
104     </saml2:Attribute>
105     <saml2:Attribute FriendlyName="Nome" Name="name">
106       <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
107         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
108         xsi:type="xs:string">Arianna</saml2:AttributeValue>
109     </saml2:Attribute>
110     <saml2:Attribute FriendlyName="Cognome" Name="familyName">
111       <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
112         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
113         xsi:type="xs:string">Garbini</saml2:AttributeValue>
114     </saml2:Attribute>
115     <saml2:Attribute FriendlyName="Data di nascita" Name="dateOfBirth">
116       <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
117         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
118         xsi:type="xs:date">1968-05-22</saml2:AttributeValue>
119     </saml2:Attribute>
120   </saml2:AttributeStatement>
121 </saml2:Assertion>
122 </saml2p:Response>

```

## A.2.6 IdP-Proxy to eIDAS-Service

### eIDAS Response

```

1 <saml2p:Response xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2   xmlns:eidas="http://eidas.europa.eu/attributes/naturalperson"
3   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
4   xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
5   Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
6   Destination="https://service-test-eid4u.polito.it/EidasNode/IdpResponse"
7   ID="_I4HYGVHFIj1FEf4dgW7sW6TqXvrwBatziQiAUez3nS.j4QmyVPOHtFPFJRwBNdN"
8   InResponseTo="_-7T6xlFmkoZ71P03K19vKLYX16ZVyaSw.QSHmLrmnFwKEJiUymd-
   HmX3ELnRBVL"
9   IssueInstant="2019-12-05T12:41:12.706Z" Version="2.0">
10 <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
11   https://idp-proxy-test-eid4u.polito.it/idproxy/idpeumetadata/</
   saml2:Issuer>
12 <ds:Signature>
13   <ds:SignedInfo>
14     <ds:CanonicalizationMethod
15       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
16     <ds:SignatureMethod
17       Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
18     <ds:Reference

```

```

19      URI="#_I4HYGVHFIj1FEf4dgW7sW6TqXvrwBatziQiAUez3nS.j4QmyVPOHtFPFJRwBNdN
20      ">
21      <ds:Transforms>
22      <ds:Transform
23      Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
24      >
25      <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
26      </ds:Transforms>
27      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" /
28      >
29      <ds:DigestValue>
30      PaV/864iNZEjTUedrNmFsUXEpo1JLbmrB5iWABAR4o
31      </ds:DigestValue>
32      </ds:Reference>
33      </ds:SignedInfo>
34      <ds:SignatureValue>
35      OtBxYdJTYIG35sEwXnG3oQbmzYlHsfsPkpy0QPMrD1z4V1
36      </ds:SignatureValue>
37      <ds:KeyInfo>
38      <ds:X509Data>
39      <ds:X509Certificate>
40      MIICKzCCAbKgAwIBAgIULN+1E5fJs4K2d0EkpCYZIUUA/X1jB04NKgd5LacXfSAA==
41      </ds:X509Certificate>
42      </ds:X509Data>
43      </ds:KeyInfo>
44      </ds:Signature>
45      <saml2p:Status>
46      <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
47      <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success
48      </saml2p:StatusMessage>
49      </saml2p:Status>
50      <saml2:EncryptedAssertion>
51      <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
52      Id="_6103a746f246bbe2bd48905d3550cc6b"
53      Type="http://www.w3.org/2001/04/xmlenc#Element">
54      <xenc:EncryptionMethod
55      Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm" />
56      <ds:KeyInfo>
57      <xenc:EncryptedKey Id="_22ec0155b2f341ed6212dbdff9bac6fd">
58      <xenc:EncryptionMethod
59      Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
60      <ds:DigestMethod
61      Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
62      </xenc:EncryptionMethod>
63      <ds:KeyInfo>
64      <ds:X509Data>
65      <ds:X509Certificate>
66      MIIIF5DCCA8ygAwIBAgIJANMSzeK2MkKgR9Q2Fzp3HkLQF
67      </ds:X509Certificate>
68      </ds:X509Data>
69      </ds:KeyInfo>
70      <xenc:CipherData>
71      <xenc:CipherValue>
72      gCGKz2mmdykxE5/e5RsTZ7napK01jadr3l4FEEt24qXCKqX6BrL+
73      XVWgTsU7xxdBKD2BG4Q=
74      </xenc:CipherValue>
75      </xenc:CipherData>

```

---

```

72     </xenc:EncryptedKey>
73   </ds:KeyInfo>
74   <xenc:CipherData>
75     <xenc:CipherValue>
76       KvW5v9We8RpDrr0hHU48wc/5VqWJkXw8jtgzzNRzysscRIKd5admeTPMOLmpn
77     </xenc:CipherValue>
78   </xenc:CipherData>
79 </xenc:EncryptedData>
80 </saml2:EncryptedAssertion>
81 </saml2p:Response>

```

---

## A.2.7 eIDAS-Service to eIDAS-Connector

### eIDAS Response

---

```

1 <saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
2   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3   xmlns:eidas="http://eidas.europa.eu/attributes/naturalperson"
4   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
5   Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
6   Destination="https://connector-test-eid4u.polito.it/EidasNode/
7     ColleagueResponse"
8   ID="_s8zPRr1PcaF.mY0waaX8H82emMPCnZE8cqfF08l.xYyFXssDZmIQEL3wk.o8gt_"
9   InResponseTo="_882EW-EpjxyDF_YTH00w0KULFYNPzhXmurz31yHJ6N0Cht.
10     dr0pZaMY03t1suh6"
11   IssueInstant="2019-12-05T12:41:14.425Z" Version="2.0">
12   <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
13     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
14     https://service-test-eid4u.polito.it/EidasNode/ServiceMetadata
15   </saml2:Issuer>
16   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
17     <ds:SignedInfo>
18       <ds:CanonicalizationMethod
19         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
20       <ds:SignatureMethod
21         Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
22       <ds:Reference
23         URI="#_s8zPRr1PcaF.mY0waaX8H82emMPCnZE8cqfF08l.xYyFXssDZmIQEL3wk.o8gt_"
24         ">
25         <ds:Transforms>
26           <ds:Transform
27             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
28         </ds:Transforms>
29         <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512" />
30         <ds:DigestValue>
31           YjdkfGaHbqysyfc2w150zWaNrPnVJDn/nCB+GpTQDNXHYAvNle5qCN
32         </ds:DigestValue>
33       </ds:Reference>
34     </ds:SignedInfo>
35     <ds:SignatureValue>

```

---

```

34      T2ETPYZIf1BC+1DpbN/ACmwV3/V7s1wS51nhqX3+h46nY+pXsbKeB8k
35    </ds:SignatureValue>
36    <ds:KeyInfo>
37      <ds:X509Data>
38        <ds:X509Certificate>
39          MIICkjCCAhmGAWIBAgIJAPpljANQoDsIeOUbmhqZhWe5szLOE=
40        </ds:X509Certificate>
41      </ds:X509Data>
42    </ds:KeyInfo>
43  </ds:Signature>
44  <saml2p:Status xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
45    <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
46    <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success
47  </saml2p:StatusMessage>
48 </saml2p:Status>
49 <saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion
50   ">
51   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
52     Id="_e1a9908ba3027759a999a662551fce10"
53     Type="http://www.w3.org/2001/04/xmlenc#Element">
54     <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
55       Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm" />
56     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
57       <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
58         Id="_26d9ef9391adf41cf3ad3ab864b18e6f">
59       <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
60         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
61       <ds:DigestMethod xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
62         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
63     </xenc:EncryptionMethod>
64     <ds:KeyInfo>
65       <ds:X509Data>
66         <ds:X509Certificate>
67           MIIF4jCCA8qGAWIBAgIJAMlk7Nh9TEekfKjQTA==
68         </ds:X509Certificate>
69       </ds:X509Data>
70     </ds:KeyInfo>
71     <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
72       <xenc:CipherValue>
73         nRFeSg1gxkPaJ29CutFYhAPU0dmSfNJGQq6JvwRUf+nJGiR3THC9KA=
74       </xenc:CipherValue>
75     </xenc:CipherData>
76   </xenc:EncryptedKey>
77 </ds:KeyInfo>
78 <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
79   <xenc:CipherValue>
80     GehFeepq7gTv17EgVxDSUy6+XPCKvfr0jvt660T68qJni2l9wN06JKOZgHMo6vt7TE=
81   </xenc:CipherValue>
82 </xenc:CipherData>
83 </xenc:EncryptedData>
84 </saml2:EncryptedAssertion>
85 </saml2p:Response>

```

---



## A.2.8 eIDAS-Connector to Wifi-Auth eIDAS-SP

### eIDAS Response

```

1  <saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
2    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3    xmlns:eidas="http://eidas.europa.eu/attributes/naturalperson"
4    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
5    Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
6    Destination="https://wifi-auth-eid4u.polito.it/SP/ReturnPage"
7    ID="_A0tCg20zdDuQmbjqXowaZk0sh_zT89nwB.gQbCR5ni9yHvrjzkzyh0lGQL4Qxq"
8    InResponseTo="_sRmCiBCyS21tAgWrQeJcpWr.PefcyCiGQbsDvjHE5xn27Wxhh0uqF1.
      anRGGpBq"
9    IssueInstant="2019-12-05T12:41:19.478Z" Version="2.0">
10  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
11    Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
12    https://connector-test-eid4u.polito.it/EidasNode/
      ConnectorResponderMetadata
13  </saml2:Issuer>
14  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
15    <ds:SignedInfo>
16      <ds:CanonicalizationMethod
17        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
18      <ds:SignatureMethod
19        Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512" />
20      <ds:Reference
21        URI="#_A0tCg20zdDuQmbjqXowaZk0sh_zT89nwB.gQbCR5ni9yHvrjzkzyh0lGQL4Qxq
          ">
22      <ds:Transforms>
23        <ds:Transform
24          Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
          >
25        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
26      </ds:Transforms>
27      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmenc#sha512" /
          >
28      <ds:DigestValue>
29        ytkCbKhUrjNfUI6BDJ3Wqdu0qoSRUpf8gu6/8Za0EzJKsYqnrEw==
30      </ds:DigestValue>
31    </ds:Reference>
32  </ds:SignedInfo>
33  <ds:SignatureValue>
34    64sDgGmwG/atbLFEIk2Ui0lLr10ET9ErLnhoIka8YDl1aKJlGXwZ+PhLIBUw1ZcsYM
35  </ds:SignatureValue>
36  <ds:KeyInfo>
37    <ds:X509Data>
38      <ds:X509Certificate>
39        MIICkTCCAhegAwIBAgIJAMIXa5EPQZdFMAoGCCqertT5W5xIKP225l8LbVTpYnHEB8ORHag
          ==
40      </ds:X509Certificate>
41    </ds:X509Data>
42  </ds:KeyInfo>
43  </ds:Signature>
44  <saml2p:Status xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
45    <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
46    <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success

```

```
47     </saml2p:StatusMessage>
48 </saml2p:Status>
49 <saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion
">
50   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
51     Id="_69d27882cf8be2cb8a7aa62536215bef"
52     Type="http://www.w3.org/2001/04/xmlenc#Element">
53     <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
54       Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm" />
55     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
56       <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
57         Id="_e7e3d60e63c655de569e8a1ba008faa0">
58         <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
59           Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
60           <ds:DigestMethod xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
61             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
62           </ds:DigestMethod>
63         </ds:KeyInfo>
64         <ds:X509Data>
65           <ds:X509Certificate>
66             MIIDSTCCApmgAwIBAgIUOBopz8Pq+/fB6mN0t2Hv5Mxewgcs8dsMcOVM/
              ZeuOFnQ==
67           </ds:X509Certificate>
68         </ds:X509Data>
69       </ds:KeyInfo>
70       <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
71         <xenc:CipherValue>
72           EIUjJt1VjPT707LRsch7c1T+juVixoPKr72ybL5YVgP0sn8vPEwvRZQGMHA==
73         </xenc:CipherValue>
74       </xenc:CipherData>
75     </xenc:EncryptedKey>
76   </ds:KeyInfo>
77   <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
78     <xenc:CipherValue>
79       Ajx39Du2CKVo9nz9ViMp8E3PPD1SNXn0nCi6wgcfMcc5Cq6gd0oMzeoMFtvo=
80     </xenc:CipherValue>
81   </xenc:CipherData>
82 </xenc:EncryptedData>
83 </saml2:EncryptedAssertion>
84 </saml2p:Response>
```

---