F SRS EXTRACT

F.1 Overall Description

F.1.1 Product Perspectives

The system will be built according to the three-tiers architecture pattern and will consist of the following components :

Presentation

Front-End (**FE**) based on a web interface for consulting previous reports and for wizards according to roles

Application

Search Engine (**SE**) for retrieving metadata.
Automation system (**AS**) for installing test virtual environments relying on baselines.

Persistent

Back-End (**BE**) as a data-centric storage system.

The process of performing a software product assessment is driven in a role-based sequence of inputs lists. Once the sequence is complete, a report is automatically generated and stored on the BE and then available via the FE.

F.1.2 Product Functions

Through the FE, the user should be able to :

- **Search** for previous reports based on several criteria. The result of the search will be displayed as a list and it will be possible to view details for each report.
- According to its role, **submit** required information and file(s) about a software product to be assessed through a user-friendly wizard. This will trigger the computation of the related part of the report.

Moreover, the user with administration rights should be able to :

- **Determine roles** and related **sequences** of inputs.
- **Customize** the options of the **search** criteria.
- **Customize** the **tasks** suite for the assessment.
- **Install plugins** (facultative).

F.1.3 User Classes and Characteristics

There are three user classes (only interacting with the FE) :

Normal Users (NU)

They have access to dedicated wizards for filling in a predefined list of data items.

Security Users (SU)

They have access to dedicated wizards for filling in a predefined list of data items, running automated security assessment tasks and performing assessment tests.

Administrators (ADM)

They manage the system via a specific separated part of the FE. They are able to customize the lists and the data items of NU and SU wizards.

All users should have the ability to search for existing reports.

F.1.4 Operating Environment

The targeted environment is a (farm of) virtualization server rack(s). All the components will be running on **Linux VM**'s. **FE** will be running a web server, **SE** and **AS** could be installed on different VM's for distributing the computation load and **BE** could be installed on multiple VM's with a NoSQL database system for replication.

The components should reside in a same subnetwork, separated from the baseline management and VM testing subnetworks.

F.1.5 Design and Implementation Constraints

The system will be designed according to the following characteristics :

- **Extensibility** : It should be possible to easily adapt all available data schemes for tuning the reporting output, to associate features such as search criteria with new data fields and to add new basic search criteria.
- **Portability** : The implementation should be made compatible with Python 2 and 3 and should not be dependent on a specific platform.

- **Scalability** : System's components should be separated as to be able to scale to the operating environment.
- **Modularity** : The implementation should be sufficiently modularized in order to easily integrate new functionalities.

F.1.6 User Documentation

The system will be delivered with, at least, the following documentation :

- **Installation** Documentation : intended for system administrators, in order to be able to correctly install and configure the system.
- **User Documentation** : intended for managers, technicians and assessors, in order to be able to efficiently use the system.
- **Improvement Guide** : intended for administrators, in order to be able to easily create new plugins.

F.1.7 Assumptions and Dependencies

The system should be able to make use of or implement an interface with the following kinds of systems :

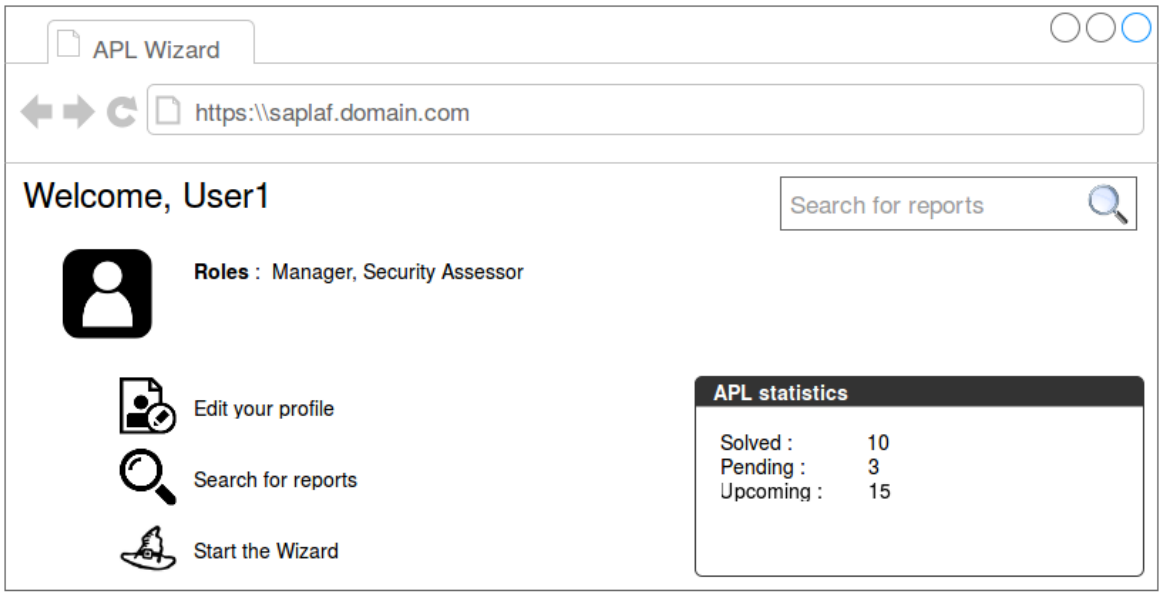
- **Vulnerability scanning** system : by deploying and interfacing an open-source project, e.g. OpenVAS.
- **Sandbox** : by providing an interface with an open-source project such as Cuckoo.
- **Pentesting** : by deploying a ready-to-use information security platform such as Kali Linux.

F.2 External Interface Requirements

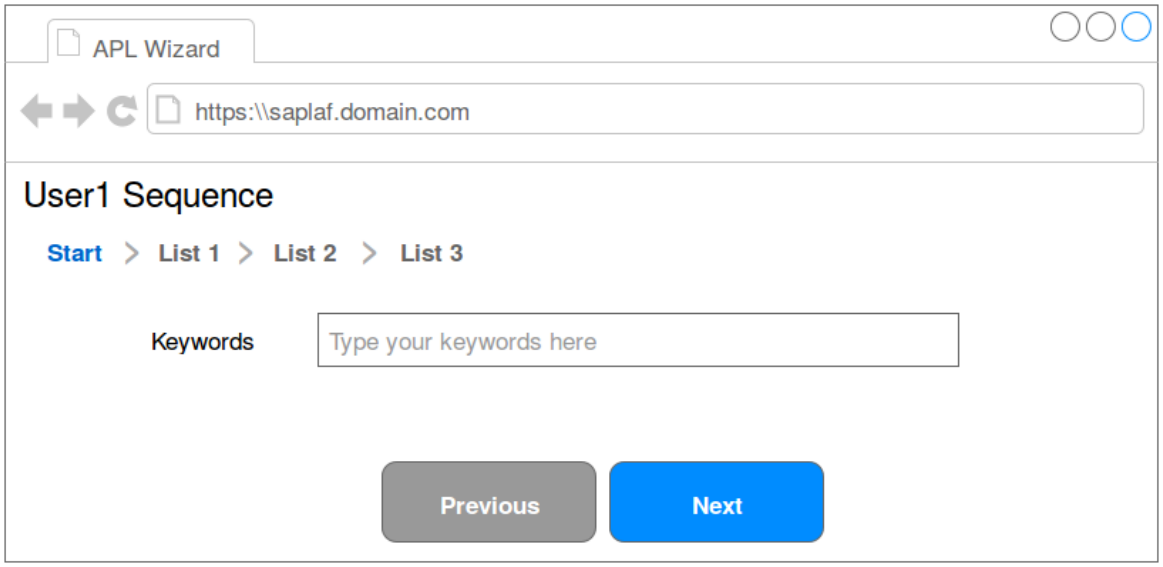
F.2.1 User Interfaces

When a NU/SU visits the homepage, he/she should see the **login page** which offers the possibility to sign in or up with a dedicated form.

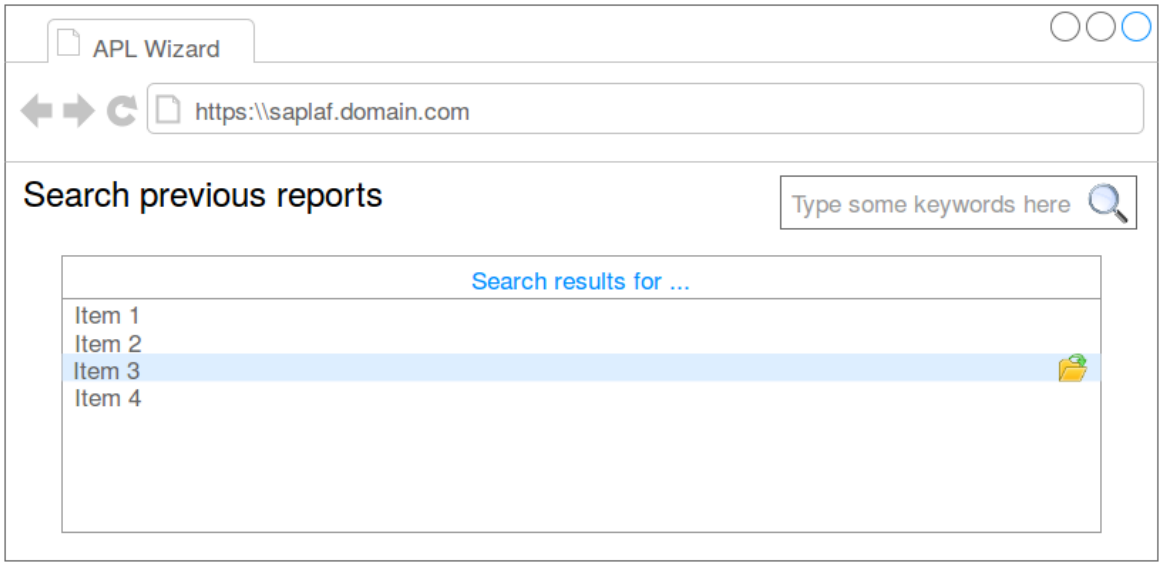
When a user is logged in, he/she should see the **homepage** with links to his/her **profile**, a **search** and his/her related **wizard** as illustrated in the following image.



The FE should use a page layout in **single column** as it should not handle a lot of user functionalities. The wizards should use multi-step forms to input required data in a predefined order relying on the sequence structure defined by an administrator (such as, for example, this of Figure 4.1). The following image illustrates what the wizard for a generic user should look like.



The wizards, according to the roles, should display a **sequence of lists**. Each list should contain data fields to fill in provided that information is automatically searched if applicable for the given data and that **results** of this **search** are displayed as drop-down lists of **suggestions**.



The **search page** should display a search field and a list of matching results with the links to the related reports.

Administrators should have access to a dedicated part of the FE available through an entry point different from the users' interface, thus with a different login page.

F.2.2 Hardware Interfaces

As the system's components will be virtualized, at least one **virtualization-capable server** is required for the prototype. The whole system will rely on this hardware but could also be scaled to several physical servers.

F.2.3 Software Interfaces

The **FE** should **communicate with** the **SE** and the **AS** through the local network **for** opening **asynchronous tasks**.

The **SE** should use **SCAP** for interacting through the network with external servers in order to retrieve security content.

It should also use **API**'s to public **search engines** (i.e. Google) to retrieve metadata.

The **AS** should **communicate with** its **child systems** and **security tools** through the local network for controlling the security assessment tasks.

F.2.4 Communication Interfaces

The system should use **classical network communication** protocols between its components.

F.3 System Features

This section lists system's main features to be implemented. More details are given in a separate document, the SDD. Priorities are rated on a relative scale from a high of 1 to a low of 3.

F.3.1 Front-End

ID	Title	Roles	Dependencies	Prio
FE01	Sign up	NU, SU		1
FE02	Registration confirmation	ADM	FE01	3
FE03	Sign in	NU, SU	FE01(, FE02)	1
FE04	Edit profile	NU, SU	FE03	2
FE05	Search for reports	NU, SU	FE03	1
FE06	Start the wizard	NU, SU	FE03	1
FE07	Sign in (admin)	ADM		1
FE08	Manage users	ADM	FE07	1
FE09	Manage roles	ADM	FE07	1
FE10	Manage process	ADM	FE07	1

F.3.2 Search Engine

ID	Title	Roles	Dependencies	Prio
SE01	Search for a data item	System		1
SE02	Search for SCAP content	System		2
SE03	Datamine	System	SE01	3
SE04	Mine SCAP content	System		3
SE05	Mention outdated version	System		3

F.3.3 Automation System

ID	Title	Roles	Dependencies	Prio
AS01	Create baseline	System		3
AS02	Manage baseline	System		1
AS03	Create a template from a baseline	System	AS01	3
AS04	Deploy a testbed from a template	System	(AS03)	1
AS05	Start application perimeter testing	System	AS04	1
AS06	Start system perimeter testing	System	AS04	1
AS07	Start network perimeter testing	System	AS04	1

F.3.4 Back-End Storage

ID	Title	Roles	Dependencies	Prio
BE01	Generate report	System		1
BE02	Manage database	System		2

F.4 Non-Functional Requirements

ID	Category	Requirement	Description
PERF01	Performance	Page load	Any web page has to be loaded in less than 5 seconds.
PERF02	Performance	Asynchronous Tasks Warnings	As lots of asynchronous tasks cannot provide results in a 5-seconds frame, all fields related to these tasks must clearly display a warning that these are not started, pending or aborted.
PERF03	Performance	Availability	Average FE availability should be at least 99%.

ERGO01	Ergonomics	User-friendliness	The web pages must be designed as user-friendly as possible in order to facilitate work. Main features (search and report view) should be easy to find/use.
ERGO02	Ergonomics	Error messages	Inconvenient pop-ups should be avoided in case of error and messages should be displayed inline and self-explantory.
SECU01	Security	Access control	Any application user or administrator should not have access to any other system's component than the FE via the web interface.
SECU02	Security	Remote administration	Access to any component via SSH is left for system administrators for management purpose.
SECU03	Security	Login	Login form should not be vulnerable to SQL Injections.
SECU04	Security	Website	FE's website should not be vulnerable to XSS and CSRF.
TEST01	Testing	Security	A simple pentest should be led on FE to assess Security Non-functional Requirements.
TEST02	Testing	Extensibility	At least one data scheme with its associated data items should be assessed for correct output and reporting.
TEST03	Testing	Portability	The components should be tested on both Python 2 and 3 and on Linux (and Windows if possible).
TEST04	Testing	Scalability	SE, AS and BE should be tested as distributed on two machines.