Software Requirements Specification

Prepared by CODEFELLAS

for the project LINUX PASSWORD VAULT

METU - Department of Computer Engineering
CENG 491 Senior Design Project I
Fall 2015-2016

Ali Can Oğul - 1746304
Gökhan Gurbetoğlu - 1828383
Mehmet Sait Gülmez - 1678986
Muhammet Kara - 1823509

Table of Contents

1. Introduction

This Software Requirements Specification provides a complete description of all the functions and specifications of the Linux Password Vault to its readers.

1.1 Problem Definition

This project aims to create a corporate password repository where all corporate passwords can be stored securely, accessed by the authorized personnel only on a need-to-know basis, and reset when needed.

1.2 System Overview

- The project consists of two main components: Server and Client.
- The server will be a service daemon running in the background on a unix/linux system.
- The server may also connect to external services for authentication purposes, and may use a DBMS to store corporate passwords as well as user credentials.
- The client will be a console based application which connects to the server and serves as a command line interface.
- The client will connect to the server via a secure communication channel, and facilitate 2-factor authentication while connecting the server.
- All activities will be logged on the server side.

1.3 Definitions, acronyms, and abbreviations

LPV	Linux Password Vault
SSL	Secure Sockets Layer
TLS	Transport Layer Security
PGP	Pretty Good Privacy
TOTP	Time Based One Time Password
GPG	GNU Privacy Guard
SRS	Software Requirement Specification
OTP	One Time Password
AES	Advanced Encryption Standard

RSA	Rivest-Shamir-Adleman Encryption
DBMS	Database Management System
ER Diagram	Entity Relationship Diagram

1.4 Assumptions and dependencies

This document assumes that the operation environment for this project uses a variant of a Linux operating system distribution.

Also the system is dependent to the users of the LPV to have a network connection.

2. Overall description

This section of the report will try to give a general understanding about the project by explaining the factors that affect the product and its requirements.

2.1 Product functions

This section will serve as a higher level specification of the main functionalities that the system will provide. Specific requirement section will give the detailed version of the material covered in this section. The product functions are as follows:

1. User registration
2. User authorization
3. Update user credentials
4. Deauthorize user
5. Remove user
6. Store password
7. Update stored password
8. Get stored password
9. Activity logging

2.1.1 Use-case model survey

2.1.1.1 "User registration" Use Case

Name	User registration
Description	New users are added to the system by administrators. User names are unique. User passwords are randomly generated.

	Public-private key pair of the users will be automatically generated at first login of the user.
Actors	Admin

2.1.1.2 “User authorization” Use Case

Name	User authorization
Description	Admin authorizes users that need access to a stored password or password group.
Actors	Admin

2.1.1.3 “Update user credentials” Use Case

Name	Update user credentials
Description	The user credentials can be updated by the user themselves or by an admin. A user can only update their own credentials whereas an admin can update any user’s credentials.
Actors	Admin, User

2.1.1.4 “Deauthorize user” Use Case

Name	Deauthorize user
Description	Admin deauthorizes users who should not be allowed to access a stored password or password group.
Actors	Admin

2.1.1.5 “Remove user” Use Case

Name	Remove user
Description	Admin removes a user from the system. This may be needed but not limited to when user leaves company or changes division.
Actors	Admin

2.1.1.6 "Store password" Use Case

Name	Store password
Description	A new password for a specific domain is securely stored on the server. Password may be specified by the admin or randomly generated by the system.
Actors	Admin

2.1.1.7 "Update stored password" Use Case

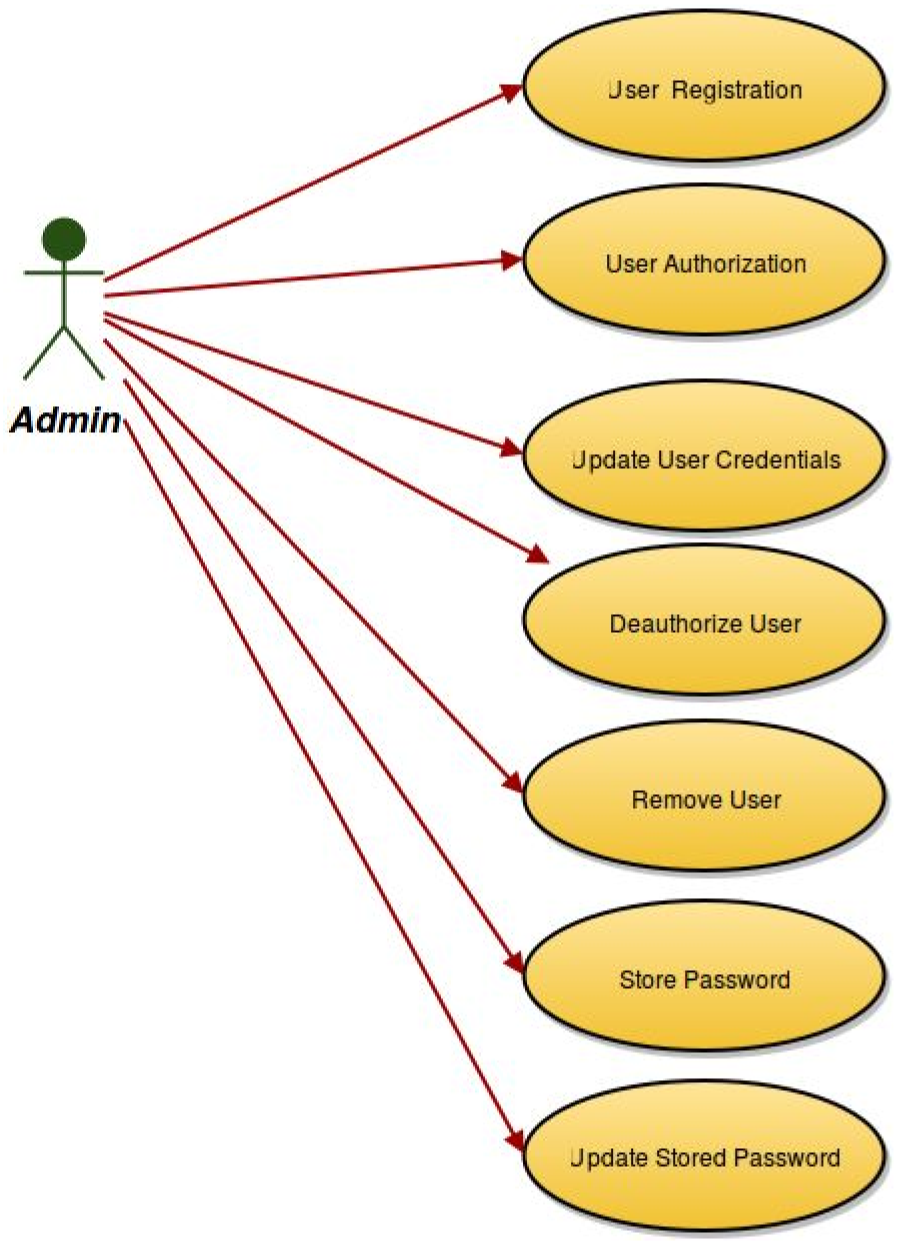
Name	Update stored password
Description	The stored password on the system is updated if it is required to. New password can be specified or randomly generated by the system.
Actors	Admin

2.1.1.8 "Get stored password" Use Case

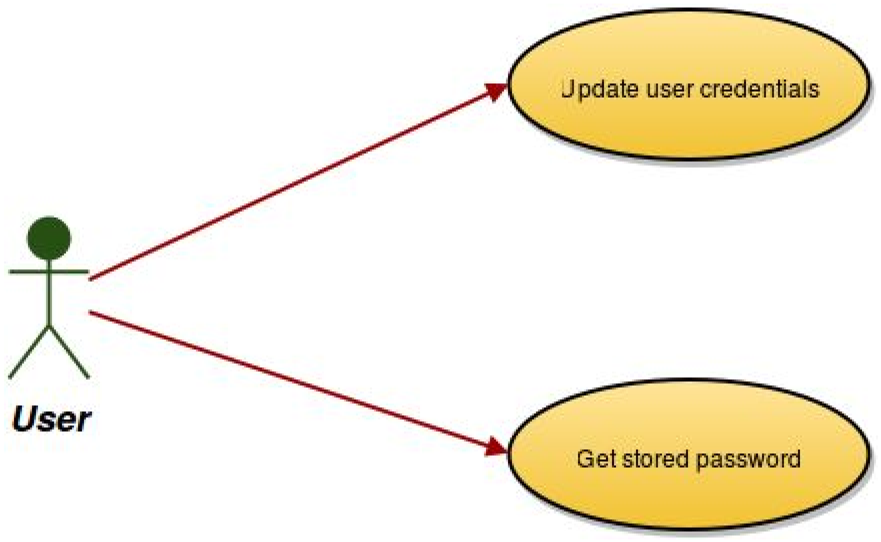
Name	Get stored password
Description	Authorized users shall be able to get stored password from the system.
Actors	User

2.1.1.9 "Activity logging" Use Case

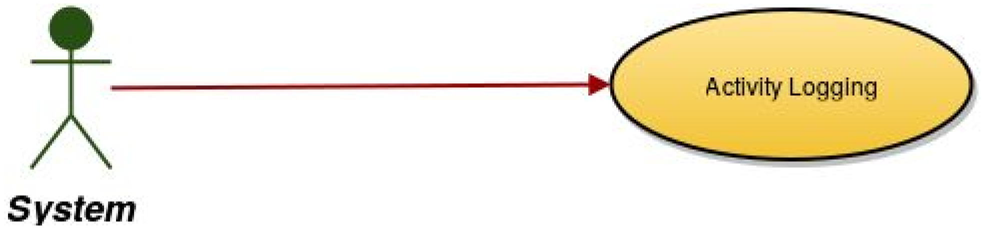
Name	Activity logging
Description	All user and administrative activities will be logged by the system. These include all of the use cases mentioned above but are not limited to them.
Actors	System



Use Case Diagram for Admin Actor



Use Case Diagram for User Actor



Use Case Diagram for System Actor

2.1.2 Actor survey

Actor's name	**Description of the actor**
User	All company employees that are registered on the system.
Admin	All company employees that have administrative rights on the system.
System	The Linux Password Vault itself, with both server and client components.

2.2 Interfaces

The Linux Password Vault is designed to provide a useful tool for storing and retrieving company passwords in a secure way. The product will be network connected application system which will run on Linux based consoles. The system will be platform dependent and can be accessed by Linux based systems.

2.2.1 User Interfaces

The project has only one interface which is Linux based console or any equivalent console emulators.

2.2.2 Hardware Interfaces

There is no hardware interface of the system.

2.2.3 Software Interfaces

The software interfaces that the system is going to use are:
MySQL: Is used to create and manage the database requirements of the project.
Ruby Programming Language: It is going to be used, since it includes lots of libraries and meets our needs during implementation part of the project.

2.2.4 Communications Interfaces

- Server and database will use TCP/IP protocol to communicate.

2.3 Constraints

- Ruby and MySQL is used in development of the software.
- The software product is aimed to run on only in Linux based operating systems.
- Security of user login authentication should be provided by main server. Also, sent and received data should be protected from any kind of third party user or software.
- Username and password is used to identify users.
- Server and database capacity is also a constraint. Under heavy network load the system may function abruptly and this should be avoided.

3. Specific requirements

Requirements can be analyzed in two parts as functional requirements and non-functional requirements.

3.1 Functional Requirements

3.1.1 Encryption / Password Storage

- All new passwords will be encrypted with a newly generated shared key on the client-side
- The shared key will also be encrypted with the public key of the authorized manager/administrator
- Then the cypher-text will be stored in the database along with the encrypted shared key

3.1.2 Authorizing A User for a Password / Group of Passwords

- Admin of the password/password group (let's call Owner) will retrieve the encrypted shared key of the password along with the public key of the new user to be authorized
- Decrypt the shared key by using his/her own private key, and re-encrypt with the new user's public key
- Send the encrypted shared key for the new user back to the server

3.1.3 Decryption / Password Retrieval

- There will be a shared key to decipher the ciphered-text (which contains the password to be retrieved), and the shared key itself will be stored as encrypted copies (encrypted with the public keys of authorized personnel) in the database.
- When an authorized personnel asks for a password, server will send the ciphered-text along with the encrypted shared-key to the client. Client will decrypt the encrypted key by using the personnels private key to get the shared key, then use this shared key to decipher the ciphered-text, which gives the desired password as plain text on the client-side.
- This way, all data will be secure in case of a compromised server.

3.2 Non functional Requirements

3.2.1 Usability

- The system will be in English language.
- The system should be available in working hours of the users.
- The system should be accessible from any computer who has network connection in the company.

3.2.2 Reliability

- System should be available at least in all the work hours.
- The database informations shall be kept by the system all the time.
- The system shall not fail on the application end. The failures on the database shall be kept at minimal. There shall be recovery mechanisms if any database failure occurs so that the information is not lost.
- When the system's components do not work correctly system should be able to show informative messages.

3.2.3 Performance

- LPV is designed for corporate use and will maintain many users. While there are numerous user entities, the number is not extremely high and can be assumed to be maximum at 5000.
- A user who does not have any experience with the system shall be able to learn the functionalities of the system in less than an hour.
- The system's use cases, such as user creation, credentials update and etc., shall be responsive and should not take more than a few seconds.
- Access to the system via networking depends on the performance of the connection speed and it still should not exceed 30 seconds for extended security purposes.
- The system shall be able to sustain the load and does not stall when at least ⅕ of the users are all connected to it simultaneously.
- If the system is degraded and unable to offer any service to the users, it shall still be accessible by the admins and it shall still carry on any password verification operation.
- The system shall not be very demanding on the hardware and shall require maximum 1 GB of disk space and minimum 256 MB of RAM on any number of cored x86 or x64 CPU with more than 1 GHz operation speed.
- Any speed of network connection shall be sufficient but at least a 128 Kbit connection shall be preferred.

3.2.4 Supportability

To ease the maintainability of the project's code-base, The Ruby Style Guide will be followed as reference of coding standards. The project's code-base will also be periodically analyzed by using an automated software for fitness to the aforementioned coding style guide.
Using external class libraries other than the ones within the scope of the Ruby Standard Library will be avoided when possible.

4 Data Model and Description

There will be six main objects in the software, namely User, Login, Admin, Registration, System, and Database classes.

4.1 User

The User object will carry the info about the users of the system, and holds the methods that allow performing operation on the user specific tasks.

4.2 Login

This object will be used by the software for logging the users and admins into the system. It will also handle session related operations.

4.3 Admin

This object hosts the data related to the Admin of the system. It will also carry the methods which will allow admins to perform management tasks on the users and the whole system

4.4. Registration

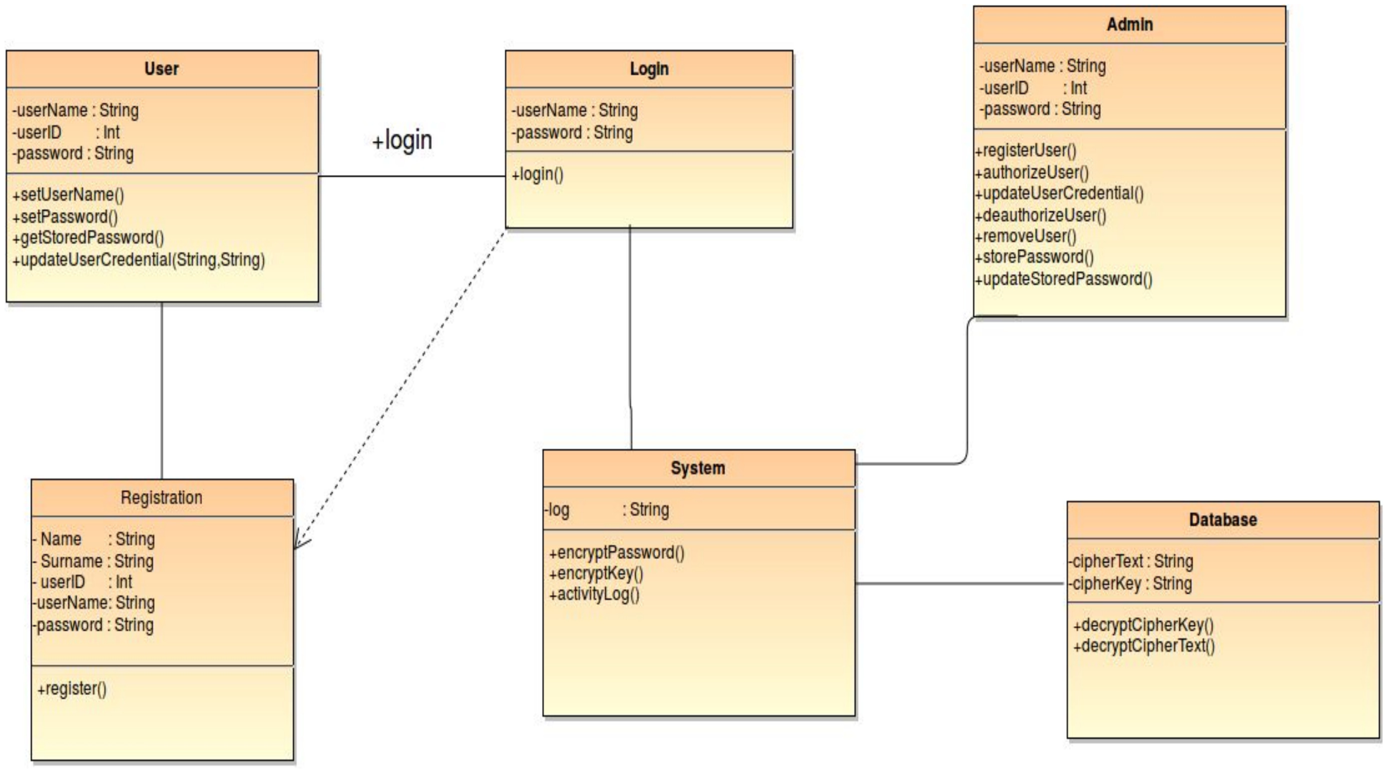
This object will carry registration info of users. It will also handle the tasks related to the user registration process.

4.5 System

This object is the main system object which handles system-wide operations, and puts all others together. This object will also handle daemonization, and logging of activities.

4.6 Database

This object will store the required information such as user credentials, passwords, roles, keys, and etc., throughout the system.



Class Diagram

About this template

This template was adapted by Emre Akbas from two sources: the IEEE 830 [1] and the "Modern SRS package" [2].

5 References

[1] IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984 , vol., no., pp.1-26, Feb. 10 1984, doi: 10.1109/IEEESTD.1984.119205,
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=278253&isnumber=6883

[2] Appendix C of Don Widrig, Dean Leffingwell, "Managing Software Requirements: A Unified Approach," Addison-Wesley Professional, Release Date: October 1999, ISBN: 0201615932.

[3] The Ruby Style Guide, URL: https://github.com/bbatsov/ruby-style-guide

[4] Ruby 2.3.0 Standard Library Documentation, URL:
http://ruby-doc.org/stdlib-2.3.0/