Requirements Specification

Captain CyBeard: Neil Before Us

Ryan Breitenfeldt | Noah Farris

Trevor Surface | Kyle Thomas

September 27, 2019

Washington State University Tri-Cities

CptS 421 Software Design Project 1

Contents

List of Figures

Revision History

Revision	Date	Author(s)	Description
1.0	11.15.2019	RB NF TS KT	First Final Doc
0.91	11.14.2019	KT	Spelling & grammar edits
0.8	11.12.2019	TS	Added more components to subsections for length
0.7	10.29.2019	NF	Edits to whole doc
0.6	10.29.2019	KT	Edits
0.5	10.28.2019	NF	Background
0.4	10.28.2019	RB	Introduction
0.3	10.15.2019	TS	Completed Overview
0.2	10.10.2019	RB NF TS KT	Filled in Environment & Operation sections
0.1	09.27.2019	KT	Document Creation

1 Introduction

This requirements specification document is to go over the requirements for the Virtual Machine file Downloader for Cypherpath's Resiliency Platform that was outlined in the `Project Plan` [1]. The document will cover how the application will provide a solution for users to easily upload their Virtual Machine files to the Resiliency Platform and also go in depth on how the users will interact with the application through a web interface. This document is intended to be understood and agreed upon by both the customer (Cypherpath) and the software development team (Captain CyBeard).

Subsequent sections will include some background information on Cypherpath and their Resiliency Platform, an overview of the application and what it is supposed to accomplish, the environment the application will execute in, and finally, how users will interact the application.

2 Background

The Cypherpath Resiliency Platform tool is a product that gives customers the ability to upload virtual machines, network configurations and more into self-contained digital environments, enabling customers to quickly recover from attacks such as ransomware. Right now, customers have to manually download their virtual disk images from the various platforms they are hosted on, and then upload those to the Resiliency Platform. Cypherpath would like a web application that will enable users to have their virtual disk images go directly from the cloud platform into the Resiliency Platform.

This project aims to give Cypherpath users a simple means of retrieving disk images from different cloud platforms, while making it easy to then upload those files to Cypherpath's Resiliency Platform, providing more value to the customer in the process.

3 Overview

The purpose of this application is to allow users to download virtual disk images from an online cloud platforms, to local file storage. The application itself will operate in the web browser using Python3 and Django. Upon starting the application the user will provide a URL to their account on the cloud platform. Processing the URL the user is then asked to provide credentials to validate they are the owner of the account. After authentication, the application will then use the API of the cloud platform and show the user their root directory on the account. The User will be able

to navigate their directory structure and choose which files and folders to download locally to their desktop. Upon accepting to download locally the application will use another API call to download the files onto their PC.

The application will be written with Django and Python3, including the requests libraries for API calls, Authentication libraries like SAML and standard libraries.

The application only allows users to view the files from VM's they own, and download those file locally to the machine they are running the application through. The users will have the ability to provide either a VMware URL to download from, or an Amazon Web Services (AWS) VM URL. The application will be designed with modularity to allow further development with different VM service providers such as Citrix, Google Drive and Dropbox.

4 Environment

The environment that the software will preside in consists of modules to interact with VMware, AWS, other virtual machine platforms and will also be interacting with authentication modules for those various platforms. The user will interact with the API's of these cloud services and authentication methods through a Django Web Application. Lastly, the environment will consist of the application storing the selected virtual machines onto the user's local machine.

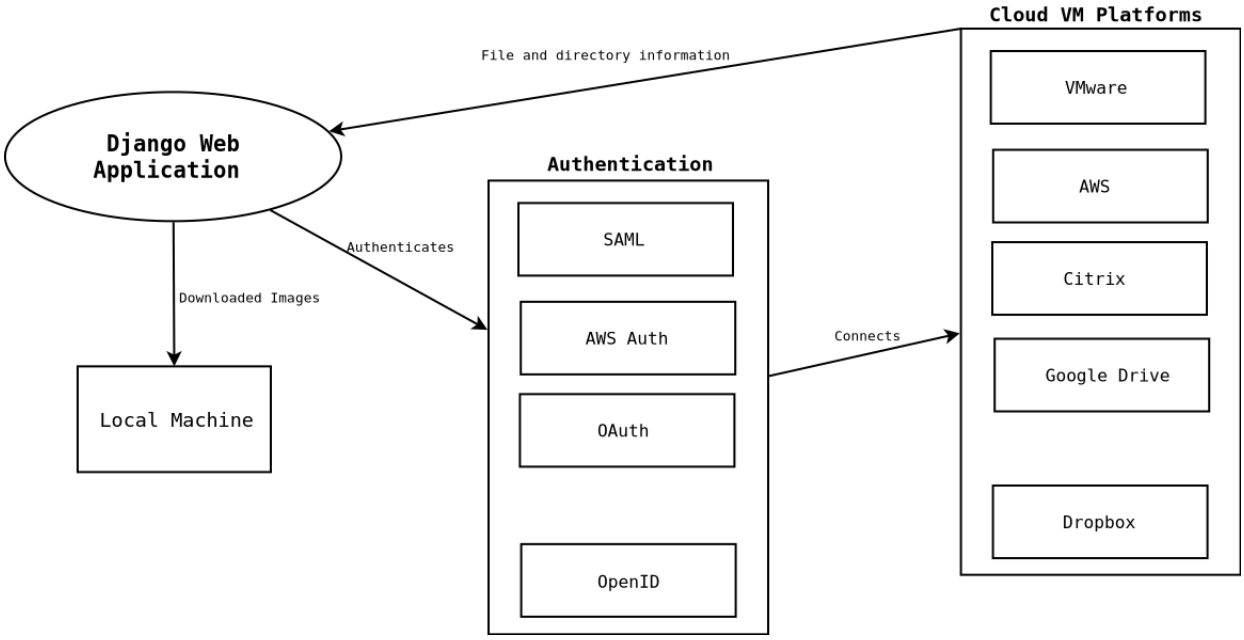

Figure 1: A visual representation of the applications environment.

4.1 Web Application

The web server that serves the application will be Django's built in development server and when the project is completed, Cypherpath will integrate the web application into their existing ecosystem using their webserver. The Django application will host the functionality of the application, including templates for the pages, where users provide Virtual Machine URLs and browse their directories. Additionally the Django framework will allow the usage of python classes that will enable the back-end operation of the application.

4.2 Authentication API's

The application will authenticate to the cloud platforms with various authentication methods and API's. To start with, the application will need to support SAML and AWS Authentication, but will need to be designed in a modular fashion so that other authentication methods can be added later on such as OpenID and OAuth. The authentication will provide a token to verify the users for their cloud platform, the application will not provide previous states for the use of the website, nor will it save previous URL's entered. The token will be passed to the request for access to the URL provided.

4.3 Cloud Storage API's

Once the application is authenticated to a cloud storage platform it will gather the directory structure for showing to the user and allowing them to select which of their files they would like to download. To start with the application will need to support interacting with VMware and Amazon Web Services. The API's that are necessary for the communication, will request the directory structure of their cloud storage and provide it to the user so they can select which virtual disk images they would like to download. The application will only interact with the Virtual Disk Image of the specified machines. The application will need to be designed so that these interactions are modular to enable adding support for other cloud platforms later. Other cloud platforms that will be included, time permitting, are Citrix, Google Drive and Dropbox.

4.4 Local Storage

Once the user has selected which files to download from their cloud platform, the application will download those files to the user's local storage. The user will not get the option to download specific files however. Unless time permitting allows implementation of the process.

The download will create copies of all user files from the cloud to local storage, The local storage location will not be able to be specified unless predefined by the download protocol of the user. Nothing else needs to be done with the files because after the application is finished and Cypherpath has integrated the web application into their platform, they will direct the downloads where they need them.

5 Operation

In the following sections the operation of the application will be described, including starting the application (invocation), the commands the application uses and finally, how to close or terminate the application.

5.1 Invocation

Since the application will be integrated into Cypherpath's platform upon completion, during development the application will be started using Django's built-in webserver and the user will not need to be authenticated to use the application. To invoke the application the user will simply enter the URL for the application (such as http://localhost:8080). They will then be presented with the screen of the application. The screen will have a single input text box for a user to insert a URL. Additionally a submit button will be provided for them to continue to step through the application.

The user interface and invocation of the machine are simple, to later be integrated into Cyperpaths resiliency tool, where invocation will by the user will be from within Cypherpath's platform. The main interface will remain the same by providing users with a single input box for URL's and a submission button. If there is an error in the initial URL provided the user will be prompted with an error that either the machine doesn't exist or cannot be reached, or if invalid credentials were provided, will be prompted as such.

5.2 User Actions

The User Actions subsection covers how the user will interact with the application and how entering the application, displaying the file structure, downloading virtual disk images and error catching will be implemented.

5.2.1 Enter URL

The user can enter a URL that goes to their cloud platform (VMware or AWS initially). The text box will be centrally located with a design to help users clearly see where to input the URL, the box will also have shadow text requesting input. The application will figure out which cloud platform the URL belongs to, through a GET request to the URL. If an error occurs with the GET request the user will be notified of the error. If the request is successful then the user will have to provide their login credentials, on a pop-up screen. The authentication will be verified with the platform they are using, (SAML for VMware and AWS Auth for AWS). Once authenticated the application will present the user with their files and folders that are on that platform.

5.2.2 Display File Structure

Once the user has selected a cloud platform and is authenticated, they will be presented with their directory structure on that platform. They will only see high level directories initially however will be able to open the directories to see files located within. They can use this information to verify all files were retrieved from the cloud. From here the user will be provided the option to download the files locally. Unless specified differently in their browser download protocol, the user can expect to find copies of the files in their Download folder When Cyperpath assimilates the application into their platform they will direct the download where they need it.

5.2.3 Download

The user will be provided with a interface that shows them the file structure of their cloud storage. A component of the display will be an optional download button. If the user wants to verify their files were downloaded properly the files will be neatly displayed to do so. Otherwise when the user is ready to download they will have to click the download button. The download itself will default to their predetermined browser download location. The user will see total download completed. The web browser's build in download manager will display the download progress.

5.2.4 Error Catching

Errors will be detected through the various aspects of the program. If the users provides an invalid URL, they will be prompted to provide a valid one. If the user does not

provide valid credentials they will be prompted to provide proper ones. There will be no limit to the amount of access attempts unless provided by the authentication package provided by Python3. If provided, then the max entry attempts will be set to 3 attempts as per standard. Additionally if the download fails then the user will have to press the download button again.

5.3 Termination

Since the user is not authenticated to the web application, to terminate their interaction with the application they will simply close the browser tab or window that has the application open. Use of the authentication token will be deleted as well per use, there will be no cookies to remember users every time they log into the accounts. Upon Termination of the application the user will have to go through the same steps every time they use the application.

References

[1] Ryan Breitenfeldt, Noah Farris, Trevor Surface, Kyle Thomas. *Project Plan.* [*Project Plan for Downloader Application*] 2019.