New Estimation for Project

Milestone 1. Web Interface (Azure WebApp)

Features

- Develop a login page.
- Implement **Azure Single Sign-On (SSO)** to secure the access to this web app.
- Integrate the **HaloPSA API** to retrieve and display a client list in a dropdown.
- *Question: Each client/company (that comes from HALO PSA) will have exactly one OAuth credential or potentially can have multiple?*
- Develop a **guided UI** that walks users through **creating Google OAuth Apps and Service Accounts** step by step.
- *Question: Is this UI just doc or interactive UI like with buttons to go to next step?*
- Enable a **drag-and-drop interface** for uploading Google OAuth JSON files.
- Before upload the credentials are encrypted using a public key that the server provides to enable end-to-end encryption.
- The backend receives the payload and decrypt before storing to Azure key-vault for easy retrieval by other applications.
- The UI should give a list of saved azure key-vault items for each client and let them delete from the key vault upon confirmation.
- The UI should show a confirmation pop up in case users tries to upload credential for a client/company that already have a stored credential in azure key-vault.

Integrations

- Azure key-vault integration
- HaloPSA API integration
- Azure SSO (Microsoft Entra ID)

Security Considerations

- Implement End-to-End encryption for JSON credential upload
- Azure SSO for secure login
- Standard security best practices for web applications i.e. use of proper security headers, content security policies, secure cookies, protection against CSRF, XSS etc.

Deployment

- App will be deployed to **Azure App Service**
- No CI/CD and deployment pipeline will be used since the scope of the project is very small, if needed will be discussed

Deliverables

- Well commented clean source code *(No integration or unit tests will be written, if needed will be discussed)*
- Tested thoroughly by developer *(No automated tests will be written, if needed will be discussed)*
- Polished UI/UX
- Deployed to **Azure App Service** environment
- A Readme.md/pdf file explaining how to run & deploy the project with different connected parts, possible troubleshooting steps.

Technology Stack & Architecture

Asp.NET Core 8(LTS version)

Razor Pages

Vanilla JS/Ajax for interactivity in the UI

It will an MVC application

Estimated Time & Cost

4 weeks, Price: $2400 ($15/hour)

Milestone 2. Key Rotation

Features

- **Azure Function / Runbook (Python)** that runs every specified amount of time (90 days)
- Pulls all saved credentials from Azure Key-vault
- Rotate the secretes
- Saves back to the azure key vault
- In case of failure for any of the keys sends email to a specified email address from a specific address

Integrations

- Azure key-vault
- Google Cloud APIs (IAM and other relevant APIs)
- Azure Graph API for emailing

Security Considerations

- Azure Managed Identity to access key-vault
- Google Service Account with proper API permissions to access the cloud
- Should not be publicly accessible

Deployment

- Deployed to Azure Functions environment or Runbook environment
- No CI/CD and deployment pipeline will be used since the scope of the project is very small, if needed will be discussed

Deliverables

- Clean and well commented code *(No unit or integration tests, if needed will be discussed)*
- Fully tested by developer *(No automated tests, if needed will be discussed)*
- Deployed on Azure Function environment or Azure Automation in case of Runbook
- A Readme.md/pdf file explaining the deployment guide, setup and configurations troubleshooting steps

Technology Stack

- In case of Azure Function Asp.Net Core 8
- In case of Runbook Python 3.8

Estimated Time & Cost

2 weeks, Price: $1200 ($15/hour)

Total estimated time **6 weeks** and the price is **$3600**, considering the hourly rate of **$15/hour**. Again this is an estimation, so actual time can be a little off, but I hope it won't be anything out of ordinary.