# Course Project Proposal

## CPEN 442

October 8th 2019

Adriano Sela Aviles (31063150)
Christian Neufeld (36719152)
Israel Trujillo Quintero (16055155)
Felipe Ballesteros (10703156)

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada

### I. What is the problem we are going to address?

Security is a hard thing to do right, and neglecting it can have severe consequences. It is well known that infrastructure engineers are often not security experts, yet they are tasked with maintaining passwords, encryption / signing keys, and runtime secrets secure throughout the build and release process. Not only are they concerned with giving team members and application deployments access to these secrets, but also that said access can be revoked. They are also concerned with knowing who accessed secrets, when, and from where, i.e. auditing. Our project addresses the issue of securing secrets with common rules -- including MFA, in a way that it simplifies securing secrets for infrastructure engineering teams.

### II. Why is this problem important?

Most large companies have dedicated Security Engineering teams embedded in their infrastructure organizations. This is seldom the case when it comes to start-ups and smaller teams. A recent study from researchers at NC State University found that over 100,000 public GitHub repositories are leaking secret keys, with thousands of new, unique secret keys being leaked every day[1]. So secrets management is a real issue affecting hundreds of thousands across the planet. This problem exists because the productivity cost of securing secrets is too high. With this project we aim to significantly reduce that cost, encouraging smaller teams to secure their secrets.

### III. How is this problem currently addressed?

Currently, secrets between team members are shared in a variety of ways. Mostly through unencrypted email or some chat platform; or even via a USB key. Others leverage services like LastPass Vault, which does not have programmatic access to secrets and requires copying and pasting onto a local machine, thus creating locally insecure copies of a secret. Finally, some larger companies leverage Open-Source APIs such as Hashicorp Vault, which is ran on premises by a security/infrastructure team. This large effort is not feasible for most small teams.

### IV. How are we proposing to address the problem?

To address this problem we propose to build a microservice that abstracts away the complexities of common secret-management rules. The service will be in the form of an Open-Source API, which we will host and provide as-a-service, as well as make it available to be run on premises by infrastructure teams for internal use. We propose improvements over existing solutions by providing secret accessing audits, multi-factor authentication, and supporting common secret management use cases:

- Decrypt project secrets with a read-only deploy key (for applications, automated processes, or service accounts)
- Decrypt project secrets with team member's personal PGP key plus an additional shared team key in AWS KMS
    - Optionally enforce MFA on users every X minutes

Example use cases:

- I want anyone on my team to be able to decrypt shared application runtime secrets with their own PGP key locally and a shared key elsewhere as a second factor
- I want to have my deployments be able to decrypt the same secrets by fetching a decryption key from AWS KMS

We build these complex rules with the help of Shamir's Secret Sharing algorithm, with which we split secrets into shards and manage those without the knowledge of our clients. Shamir's algorithm allows us to use strong encryption to enforce rules, rather than enforcing them only with our API's logic.

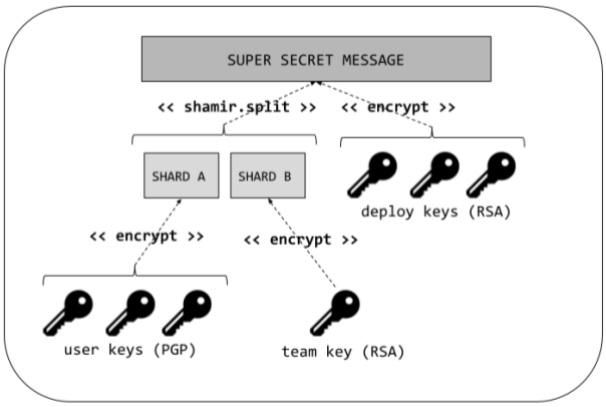


FIGURE I: Proposed Encryption Scheme

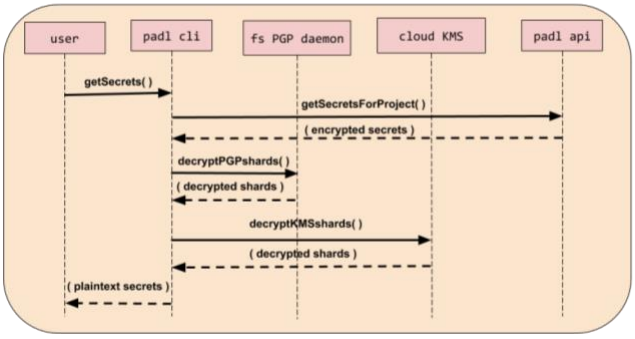


FIGURE II: User Secrets Access

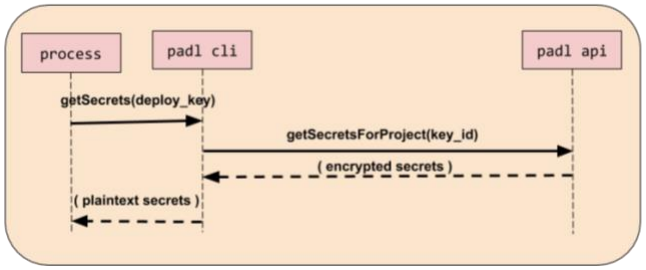


FIGURE III: Service/Process Secrets Access

## V. How is our solution better than others?

Some solutions such as Mozilla/SOPS, a command line tool for encryption of secrets at rest, require the user to have a significantly deep understanding of secrets and encryption to be able to use the tool effectively. Others, such as HashiCorp Vault, take a long time to set up, and also require some knowledge of encryption to run. We propose to build something which abstracts how secrets are stored away from the user, allowing small, security-oblivious, teams to set up secrets management for their project quickly without them needing to understand what goes on under the hood. Our main non-functional requirement is that our tool is perceived as enjoyable to use by students and engineers.

## VI. Project Completion Plan:

In order to successfully complete our project and the corresponding report that will explain our implementation and design, we will have weekly meetings and "daily stand-ups" where we will explain the tasks that we have done and the current tasks that are needed in order to move forward. For this process to work, it is required that we have a clear idea of what is being implemented and strictly follow the self-set deadlines associated with each task. A list of the main deliverables that we have planned for this project and the potential timeline is outlined below:

**Deliverables:**

- **Encryption Library** - October 25
  - **RSA & PGP Integrations** - October 10
  - **AWS KMS Integration** - October 15
  - **MFA Integration** - October 25
- **Database** - October 30
  - **Hosting set up**
- **API** - October 30
  - **Endpoint Router** - October 25
  - **Handlers (Logic, DB I/O)** - October 30
- **CLI** - November 20
  - **Network manager (Requests to API)** - November 12
  - **Command handler (Parse/Validate/Execute commands)** - November 18
  - **User's manual** - November 20
- **Demo (Video)** - November 25

## References

[1] S. Tolts,"Why (and how) you should manage secrets outside version control," *datree.io*, 15-May-2019. [Online]. Available: https://datree.io/secrets-management-aws/.