Vimcryption: Vim Encryption Plugin

Authors: Thomas Manner, Miguel Nistal

For our final project we plan to implement a Vim plugin that will allow users to apply encryption to files in the editor. In order for the project to have real-world value the plugin has to be aware of and encrypt both the file being editted when it's saved as well as any intermediate swap files created in the process of editing the file. This plugin will support encrypting new files, safely interacting and editing already encrypted files, and saving existing files as encrypted. The plugin will be broken down into several components in order to handle both the encryption code and the Vim api.

Motivation

Vim is one of the most commonly used editors in the world with thousands of plugins available to it. In a world where keeping data secure is rapidly becoming a requirement, we noticed that there was not a readily available plugin for Vim to handle this. Using the popular plugin repository Vimawesome (https://vimawesome.com), we queried "encryption" and "crypto" and came up with nothing that allowed you to encrypt your files or to read files that you have encrypted using Vim. This is where "Vimcryption" comes in!

The Vimscript Plugin Interface

All Vim plugins require some Vimscript to interact with the editor itself and load the backend code. This plugin will include the bare minimum Vimscript code to be compliant with modern Vim plugin managers (such as Vundle or Pathogen), allow users to interact with the plugin and bind commands, and intercept buffers before they're flushed to disk. The plugin will load on the opening of the application so that it's available at

all times. The Vimscript will tie Python to only necessary callbacks to ensure we're not wasting users resources by being always on.

The Python Encryption Backend

Since Vim natively supports Python based plugins, we'll be writing the heavy lifting code in Python. This is where we'll implement the encryption, reading and storing of the keys, and the decryption/file loading. The core Python code will be implemented in a file(s) seperate from the main Vimscript using only some glue code inline with Vimscript to call out to the core code. Since we have acces to the buffers in Python via the Vim module, we can disable Vim's buffer swap files in Vimscript and create encrypted swap files ourself in Python to protect users from losing information, or having plaintext information exposed, in the event of a crash.

Keys and Authentication

Since most users of Vim already have SSH configured in their home directory, we can use SSH RSA keys to do the encryption. It also means we don't have to implement key generation code and directory protections like SSH already does for us. We can also add the ability to add other users public keys (similar to how GitHub keys work) to allow your friends to work on files that you share together.

Plugin Layout and Repository

We're going to build the plugin to be Pathogen-compatible, so we'll be using the standard vim plugin layout inside of a git repository that can be used in Vundle or Pathogen calls. To work together on this we've created a private repository on GitHub to host the plugin while we work on it and for deployment testing: https://github.com/tsmanner/vimcryption.

```
vimcryption/
    LICENSE
    README
    docs/
        vimcryption.txt
    plugin/
        vimcryption.vim
        vimcryption.py
```