Master Password Erasure Code

Team members:

Huong (Penny) Vo – UIN: 728009697

Scott Allen – UIN: 125009775

***Abstract*—The team use standard AES encryption to encrypt the file storing all of users' password. Erasure Code (Reed-Solomon) method is used to divide the encryption file into small pieces. These pieces will be stored in different virtual machines to provide extra security and fault tolerance.**

Keywords—AES encryption, Erasure Code, Reed-Solomon, virtual machine

I. Introduction

The purpose of this project is helping people with managing multiple passwords. Instead of remembering 10 or 20 passwords, they can put all these passwords into a csv file. Our team will encrypt this file by AES encryption key to get a cipher text. This file is very important, so we need an extra secured method to protect it. To protect this encrypted file, we use Erasure Code to divide the key into N pieces and store them into different virtual machine. Therefore, even though an attacker finds a piece and do the decryption, one cannot get any knowledge from it. When the user wants to recover the file, he/she needs K pieces (K <= N) than get the whole file.

Improvement

The large improvement changed form presentation to submission was the consolidation. The original project worked however it had to be done over multiple command lines and changing the code to reflect what IP you wanted to send your file to. We changed this to automate using a for loop and a set array of IP addresses.

II. Related Work

We divided this project into 5 steps:

1. Create a CSV file to store all the user's passwords.

2. AES encryption:
   We write an AES encryption program by using Cryptography Library. A key (32 bits) is hardcoded because it is not the important part of the method. The IV (16 bits) is randomize every time a user do the encryption. This IV will be written in the output file for the decryption purpose.

3. Erasure Coding:
   The code is written in Java and Python by using ReedSolomon Library.
   The purpose of this step is encoding the encrypted file into N pieces. If the user be able to get more than K pieces with K lesser than N, the user will get the original file.

4. Store N pieces in different virtual machine:
   This is the most challenging step. Since our laptop runs very slow while several virtual machine open, we will divide the file into less than 4 pieces.
   In real life, user can use cloud to store their encrypted pieces.
   For simplicity and ease of use we used paramiko which is a python library for SSH connection and file transfer. SSH was chosen over raw socket programing for security.
   In our function we have a known list of IP addresses that the files need to be transferred to. Using the same VM from homework 1 we were able to use the same log in information, so we did not need to change this parameter.
   We went through the list of IP addresses and would store our fragments on different IP according to their index in the array.

III. Evaluation and results

We get into so many unpredicted errors.
program correctly.

1. Erasure Coding

   Penny takes care of this step. This step takes a long time for the team to debug. By following the encode instruction, there are N pieces created and stored in a folder called fragment. When we do the decode instruction, the program crashed.

   At first, we thought the program crashes because we didn't clean up our code properly after encoding. We tried to delete file and folders, but nothing happened. We finally realized the problem.

   Besides these N files, we have a hidden file called .DS which messed up our code. The program read all files

as parts to decode and the .DS file is not part of the 10 pieces. Because we name our pieces by counting number (ex 0-9 for 10 pieces), we say if the file number is not a number, skip the file. Even though it seemed straightforward, it was hidden and took us a long time to recognize the issue.

However, the issue was found and fixed.

2. Library Compatible

At first, I divided the project into three small steps and took care each of the individually. When I connected all of them together, they were not compatible because of different Python versions.

At the beginning, we used Crypto Cipher Library to do the AES Encryption. The library is only available in Python2, but not Python3; while the ReedSolomon Library only runs on Python3.7 or above.

It took me 15 hours to try 3 different libraries for the AES Encryption: Crypto Cipher, Cryptodome Cipher, Pyaes, and Cryptography. Cryptography is the only library I completely understand its functionality, and it runs on Python3.

3. Virtual Machine

We were able to remote in and store files on different virtual machines.

IV. Depth

This method can be used not only for password, but for any type of file.

If the attacker finds one of the pieces, after doing encryption, the attacker cannot decrypt the piece because the decryption requires

V. Correctness

After the presentation, the team touch up with some final issues. When we compile our tasks together, there were many issues related to multiple files integration.

VI. Future Improvement

In theory, the program should work with any pair of n and k. However, there are still some flaws in the Main.java; therefore, it works correctly with certain pair of n and k. For example, (5,2), (6,2), (5,4), (6,4)

VII. Live Demo

https://www.youtube.com/watch?v=OhiGJTXS4sA&ab_channel=HuongVo

VIII. Instruction

Follow the instruction in the “readme” file.

IX. effort

Work	Hour
Understand ReedSolomon Library	10
Connect Java with Python	1
Try 4 different encryption	10
Try Raw Sockets Programming	7
File Transfer to VM	3
Combining	10

X. Conclusion/summary

Even though there are many flaws, our team are proud of our project. We have learnt so many new things and spent a lot of time to fix errors.

Both of us agree that although the project takes a lot of time more than studying for the final, it is worth it.