TorComm - Secure P2P Communication

Documentation

Taha Canturk
kibnakanoto@protonmail.com

2024-05-20

Contents

1 Key Protector

1.1 What Is It

The Key protector app in security folder is used to secure a 32-byte symmetric key, 2-byte port key, 32-byte pepper. The output is in a file named *keys*. The data in this file is used for securing the local data. It needs a 4-32 byte password generated and stored by you.

To set the password, execute the *key* file which would generate the *get_keys* executable which is the key protector program. Store a copy of *get_keys* in somewhere secure if you don't want to lose it. If you lose the *get_keys* and don't have the *keys* file, then your key is forever lost.

1.2 Algorithm

The C++ code is in *security/key.cpp*, but the basic idea is as following:

```
1
2  Generate key, pepper, iv
3  Ask user for 4-32 byte password
4  result = pepper ⊕ password
5  Use sha256(result) as symmetric key to encrypt key using chacha
6  Store sha256(result) as sha256(sha256(result))
7  Generate exe for getting key (get_keys):
8      Store sha256(sha256(result)), iv, encrypted key, pepper (excluding 3-bytes)
9
10     Ask user for password:
11         Guess 3 bytes of password of unknown pepper
12         result = pepper ⊕ password
13         Compute sha256(sha256(result)) and compare with stored sha256(sha256(
       result)).
14         if no match:
15             Continue guessing all possible 3-bytes. Once done, let user guess
       again
16
17         if user guessed more than once:
18             If guessed 3 or 6 times and while guess count is smaller than 7:
19                 Pause for 10s
20             Else if Every 5 guesses:
21                 Pause for 30s
22         Sleep(random(1s,5s)) # make it a random range so that timing attacks aren
       't possible
23
24         If not valid match:
25             If more than 10 password inputs made:
26                 Delete everything in current directory
27         Else:
28             Decrypt encrypted key using sha256(result)
29             Write decrypted key to file
30
31
```

Figure 1: Key Protector

1.3 Security

since 2/3-bytes of the pepper is not stored in the *get_keys* file, they need to be guessed with every password that is entered. If we say 3 bytes of the data needs to be guessed. then the number of combinations in password is multiplied with 256^3.

e.g. if you have a 4-digit pin as your password, then there are 10^4 combinations in your password. Then the total number of combinations in password is $(256^3)(10^4) = 167772160000$.

This doesn't mean that your password needs to be smaller, it should still be 6-16 characters of numbers, small/capital letters, and symbols.