Vulnerability Report TTLock Password Reset Mechanism Attack

Arvid Viderberg

June 2019

1 Introduction

For the TT Lock, the password reset is ordered from the app. The user launches the app and then presses the *Forgot Password* button. To reset the password, the user must provide an e-mail address or a phone number associated with the account. The user then provides a new password, two times to confirm it. The user then needs to order a verification code, which triggers an email sent to the users e-mail. The verification code is a six digit number and can only ordered each 60 seconds in the smartphone app. Once the verification code is entered, the user can request to reset the password, see figure 1.

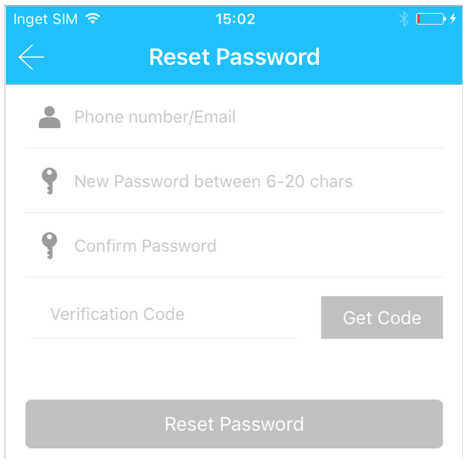

Figure 1: Password reset confirmation in the TT Lock app.

2 Execution

Firstly a test was conducted in order to try and order multiple password resets. In the smartphone app it was only possible to do so each 60 seconds. Next step was to use the man-in-the-middle proxy lab environment to view the HTTPS traffic between the smartphone app and the cloud API. This showed that the request sent to the cloud API to request a verification code didn't require authorization. A small Pyhton script was written in order to try how many password resets could be requested at the same time. The script sent the same parameters in the request as the smartphone app did. The outcome of the test showed that it was possible to order nine verification codes before the server returned an error.

Nine password reset requests meant that nine e-mails with password reset codes was sent to the account owner. In order to try password reset code invalidation, the reset code of the most recent password reset mail was used to reset the password of the user. When the user password had been reset, the second most recent password reset code was used to try and reset the password again. It was possible to use the second password reset code to reset the password. The remaining 7 reset codes were also used to try and reset the password, which were also successfully used to reset the password. The conclusion is that when a new password reset code is requested, previous codes does not get invalidated. The codes also doesn't get invalidated upon re-use. The request sent when trying to do password reset with a password reset code didn't require authentication, since the user is requesting a forgotten password and doesn't have access to his/her account.

When requesting a password reset code, the e-mail which contains the code claims that the reset code is valid for one hour. By trying to reset the account with the code after certain time intervals, it was found that the code was in fact valid for one hour and 40 minutes. This was tried with multiple codes with the same result.

If the user enters an valid e-mail och phone number, the app will start a timer of 60 seconds until the user can order a new code. If a user enters an invalid e-mail or phone number, the app will tell the user that the account doesn't exists, see figure 2. This makes it possible to gain knowledge regarding if an account exists or not.

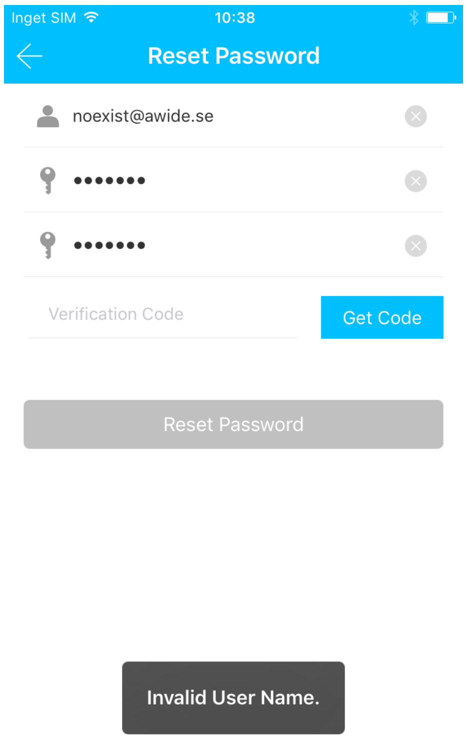

Figure 2: Entering a non-existing account in the TT Lock app password reset.

The result of the password reset for the TT Lock can be summarized as follows:

- Password reset codes are six digits, meaning there is 1 000 000 different combinations for one reset code.
- Password reset codes can only be ordered every 1 minute in the app, but are not limited via the API, making it possible to order nine reset codes at the same time via the API.
- Password reset codes are valid for one hour and 40 minutes.
- When ordering a new password reset code, the previous ordered codes are not invalidated.
- There is no (obvious) limitation of sending requests to try if the password reset code is valid.
- It is possible to gain knowledge if an account exists or not.

In order to test if it was possible brute force the password reset code, a Pyhton script was written that could send multiple password reset requests simultaneously. To not flood the server and cause a denial of service, a small test was conducted. Eight virtual machines was setup through Google Cloud in different locations around the world. Each and every machine used the Python script to do 40 requests to reset the password simultaneously. This was done in three iterations. In the slowest case, the machines had an average time of 10.7 requests per second per machine. That means 642 requests per minute and 38520 hourly requests for one machine. A reset code is active for 100 minutes, so one machine can make 64200 requests in 100 minutes. If there was only one recovery code, 16 machines would be neccessary to test all combinations within the range 0-999999 in 100 minutes. To avoid flooding the server, it is worth waiting three seconds every hundred request to avoid being blocked by the API server. In that case, the average time is 8 requests per second, which implies 48000 requests per 100 minutes, per machine. Then 21 machines would be needed to cover the entire range 0-999999 within the time frame of 100 minutes. However, it is possible to request 9 recovery codes instead of one, which significantly improves the ability of finding the right code faster, provided the codes are fairly evenly spread. The conclusion is that it is theoretically possible to brute force the password reset code. The attacker needs to know the e-mail or phone number associated with the account in order to request password reset codes and then brute force the password reset. By doing so, the attacker can gain access to the users account. This attack was not limited to the app itself but exists within the platform which multiple app developers use. This test was never conducted in full scale to not violate Swedish law, but instead a smaller theoretical test was conducted. The findings have been responsible disclosed to the owners of the platform, which have been given a 90 days grace period to patch the vulnerabilities.

It should be noted that the first time a user logins in to his/her account on a new smartphone, they need to enter a verification code very similar to the password reset code. This would prevent an attacker to login in to an account of which the attacker has reset the password and where the account hasn't been used on the attackers phone before. However, the procedure for the verification code is the same as with the password reset code, meaning that the attack for brute forcing the password reset code is also applicable for brute forcing the verification code. Hence, it is still possible for an attacker to gain access to the account, but the attacker needs to do the brute force two codes: one for the password reset and one for the sign in on a new device.