Replication of CVE to bypass password authentication for unlocking vault on LastPass

# Setting up Last Pass

LastPass is a freemium password manager that stores user’s encrypted passwords online. The standard version of LastPass comes with a web interface, but also includes plugins for various web browsers and apps for many smartphones. This is a popular password manager which enterprises use and I even know of people in CSG who use LastPass as their password manager.

I signed up an account with LastPass and used the default security settings as shown.

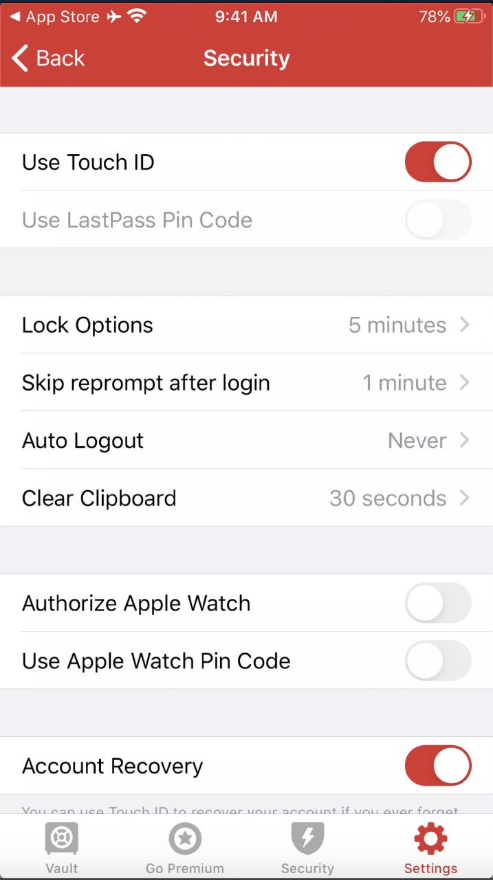


Figure 1. Default security settings for LastPass

As you can see, LastPass is set to never logout. Additionally, they enable TouchID verification by default and only locks the vault after 5 minutes of inactivity.

When LastPass is locked, users have an option to select TouchID or enter the master password.

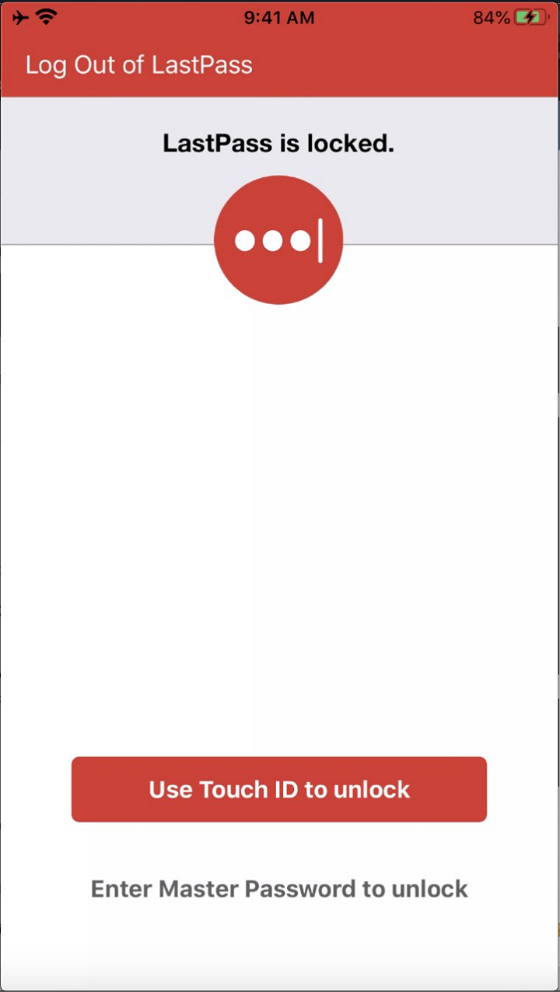


Figure 2.LastPass Lock Screen

# Vulnerable function #1- [lputil checkpw]

For this vulnerability, I will be using a wrong master password to unlock the vault.

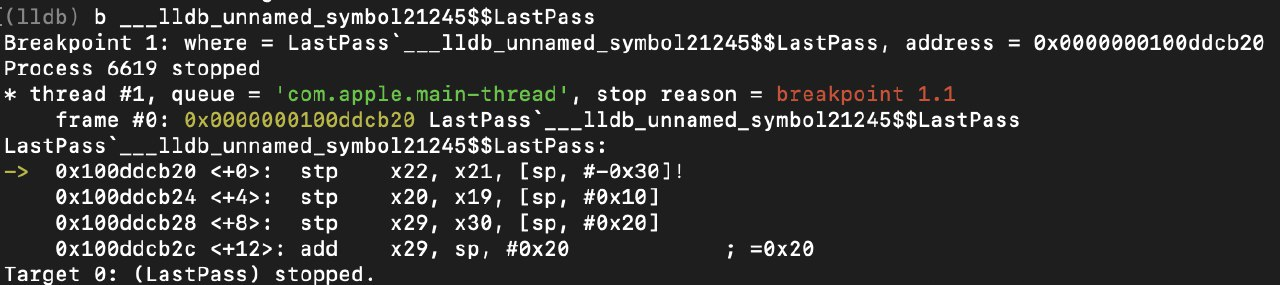
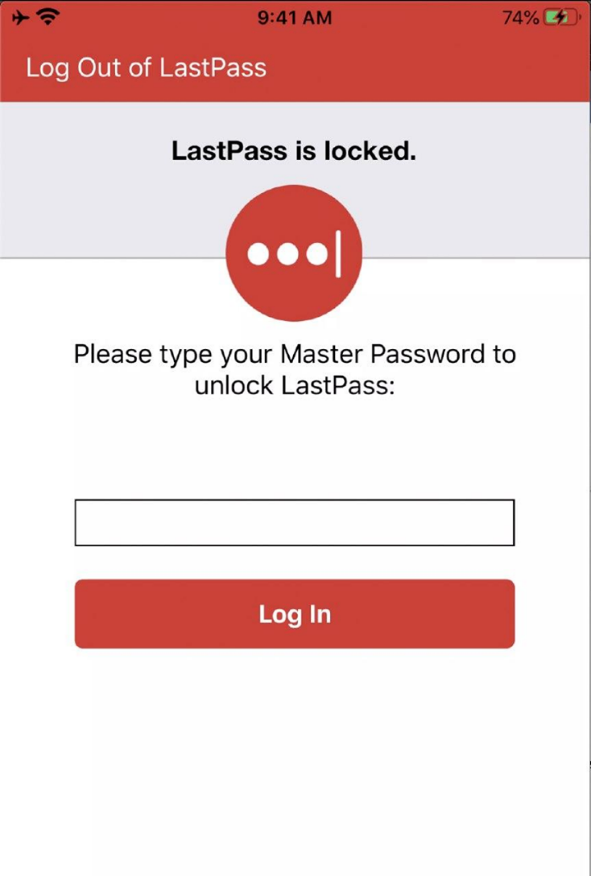
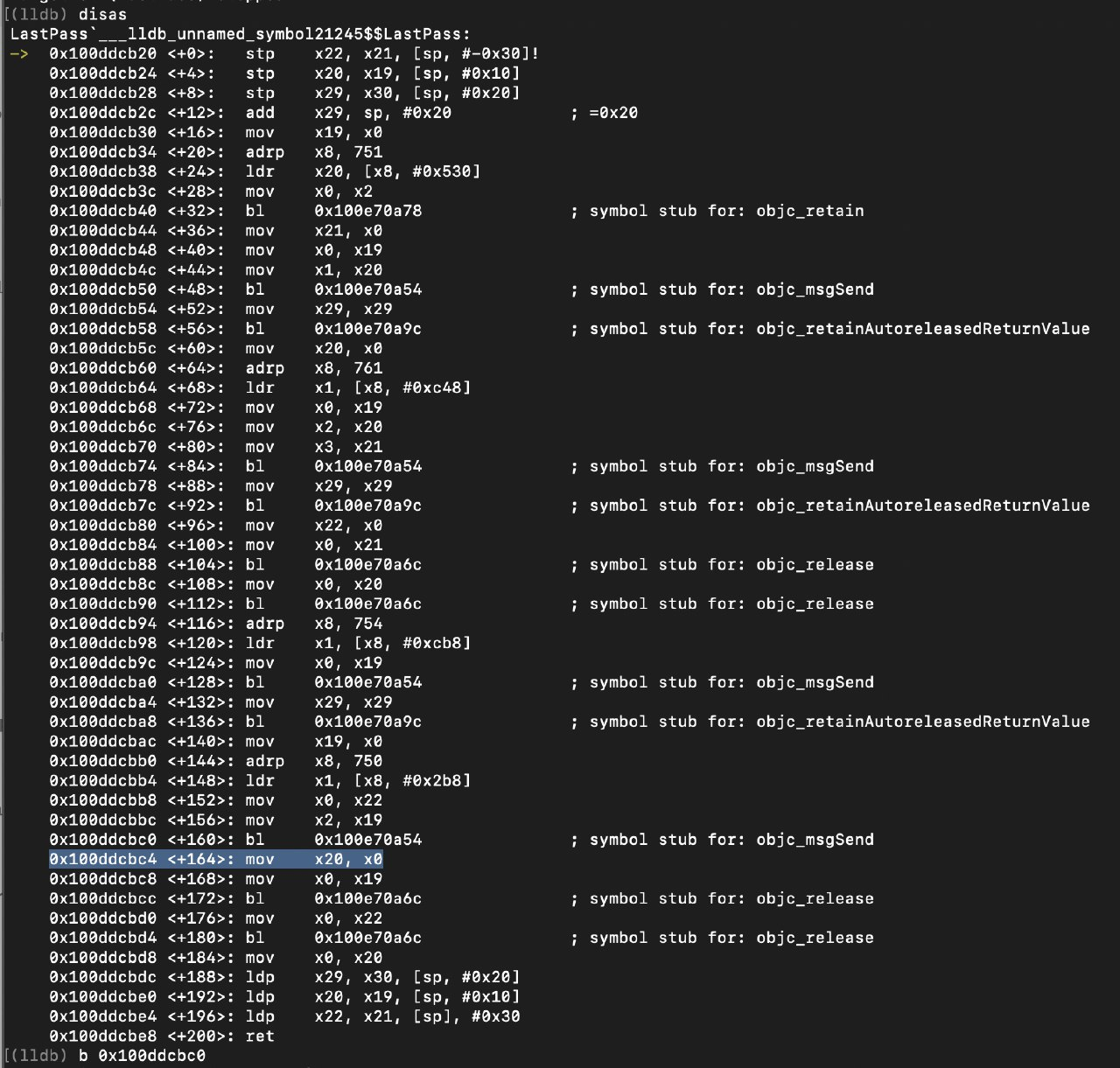


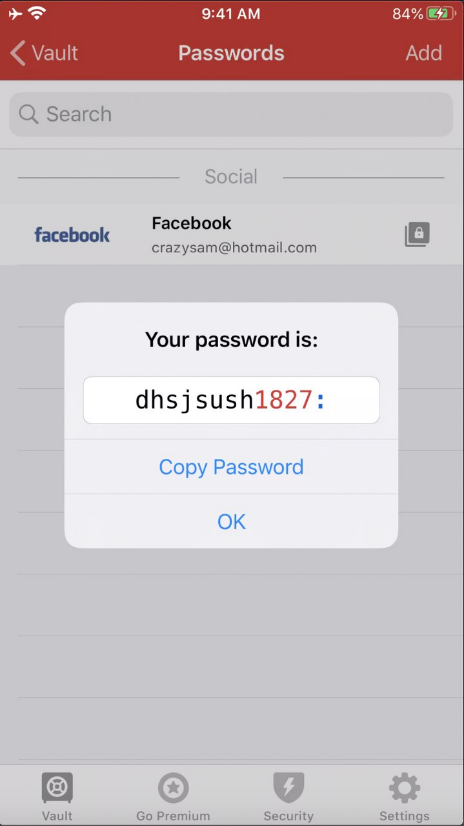
Figure 3. CheckPW is called during master password validation in lock screen

As you can see from the pseudo code above, the lock screen is basically checking if the input password is equal to the master password set. **The input password is NEVER USED to decrypt the vault. This is an insecure design.**

This also meant that if I were to debug to this assembly code and change the assembly registers where the assembly registers are compared, I will be able to bypass the lock screen.

**Proof of Attack**

1. First, I have to jailbreak the “clean device” and setup my debug server on my iPhone.
2. I then SSH into my iOS device and debug on LastPass by typing
   1. “debugserver localhost:6666 -waitfor "LastPass"
3. I then setup my debugger to be ready to debug this application
   1. iproxy 6666 6666
   2. lldb
   3. platform select remote-ios
   4. process connect connect://localhost:6666
4. Once the application is being debugged, I set a breakpoint on the function
   1. b \_\_\_lldb\_unnamed\_symbol21245$$LastPass
5. Next, I enter any wrong master password into the password field. This will cause the debugger to hit the breakpoint which was just set. 
6. Do a disas to find the relevant instruction where it is doing the isEqual comparison. **The result of the password comparison can be bypassed by modifying the return result from isEqual comparison**. In this case, I have identified the address to be 0x100a211bc during my runtime and have set a breakpoint there. 
7. Continue the app. As we have entered a wrong master password, isEqual comparison will always return 0x0. Now do a register write to x0, writing 0x1 will let the app believe that the attacker has entered the right password. 
8. Continue the app and the vault would be completely unlocked. As you can see from the screenshot, I was able to access all the saved passwords in the vault.



# Vulnerable function #2- [LPLockScreenViewController checkPin:]

LastPass has a few options to unlock the password manager. Another option of unlocking LastPass would be through LastPass Pin Code. This is a 6-digit pin code which can be used to unlock LastPass. For this vulnerability, I have enabled authentication through LastPass Pin Code.



Figure 4. Authentication through Last Pass Pin Code enabled

The new lock screen would change once LastPass Pin Code is enabled. Users are now presented with a screen to enter their pin code instead of the master password.

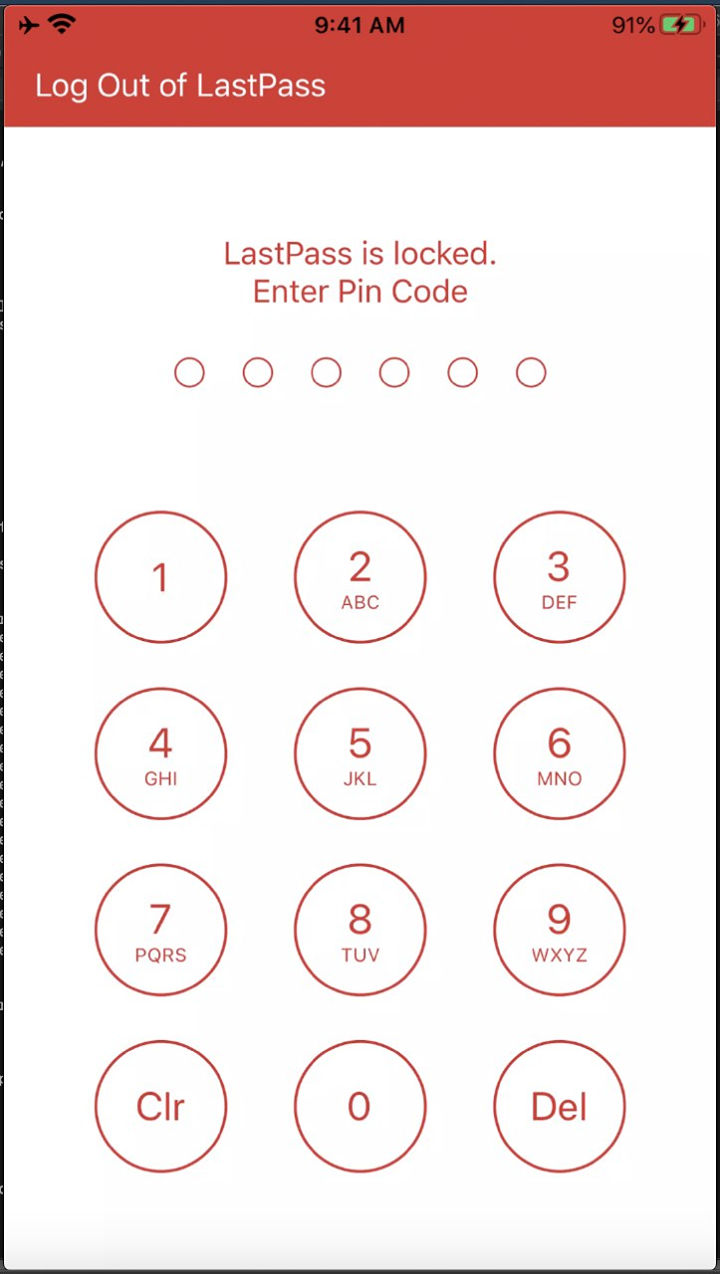


Figure 5. Pin authentication used to unlock LastPass

Similarly, the **implementation for Pin Code verification is done through [LPLockScreenViewController checkPin:] and is insecure**. This function merely does a simple string comparison checks which be easily bypassed. **The input pin code is NEVER USED to decrypt the vault**.

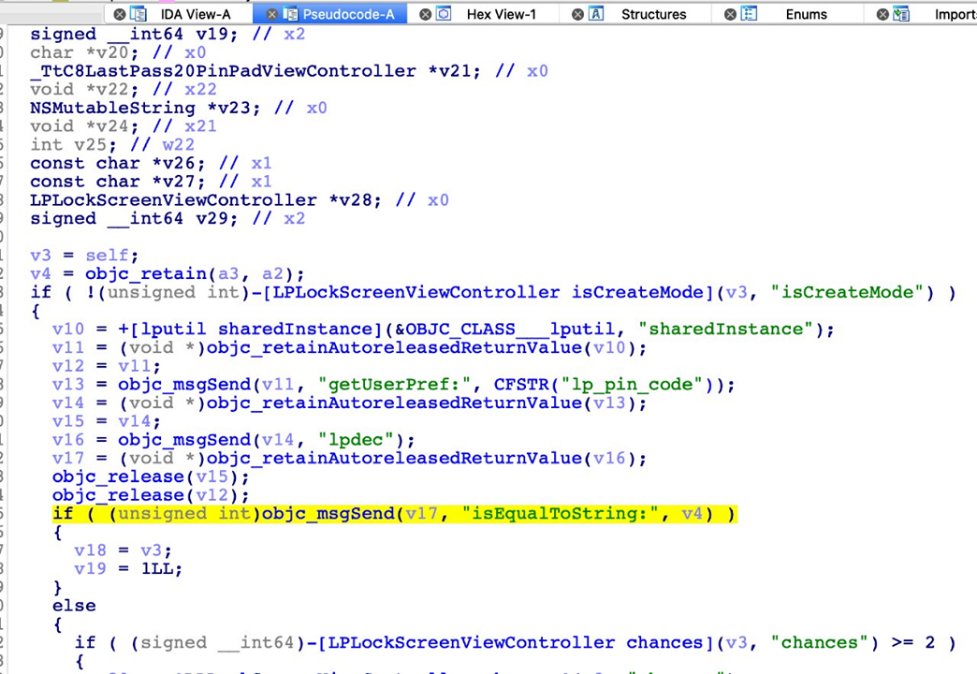
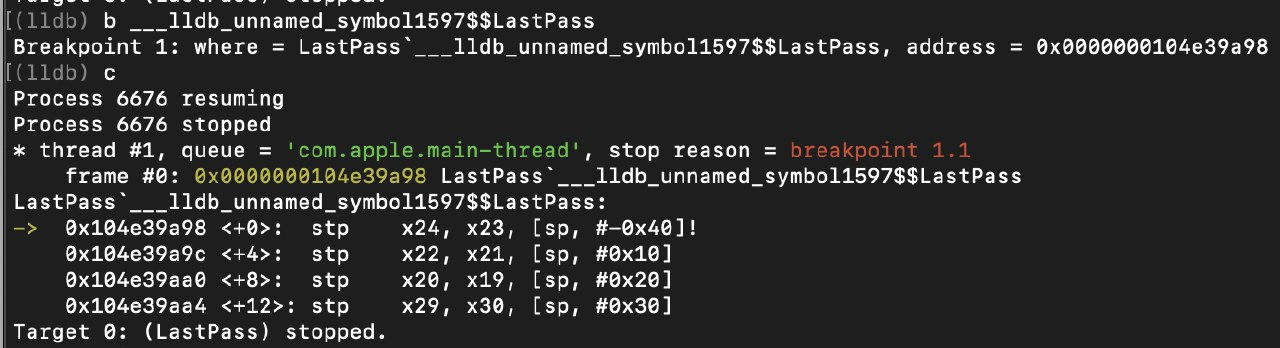
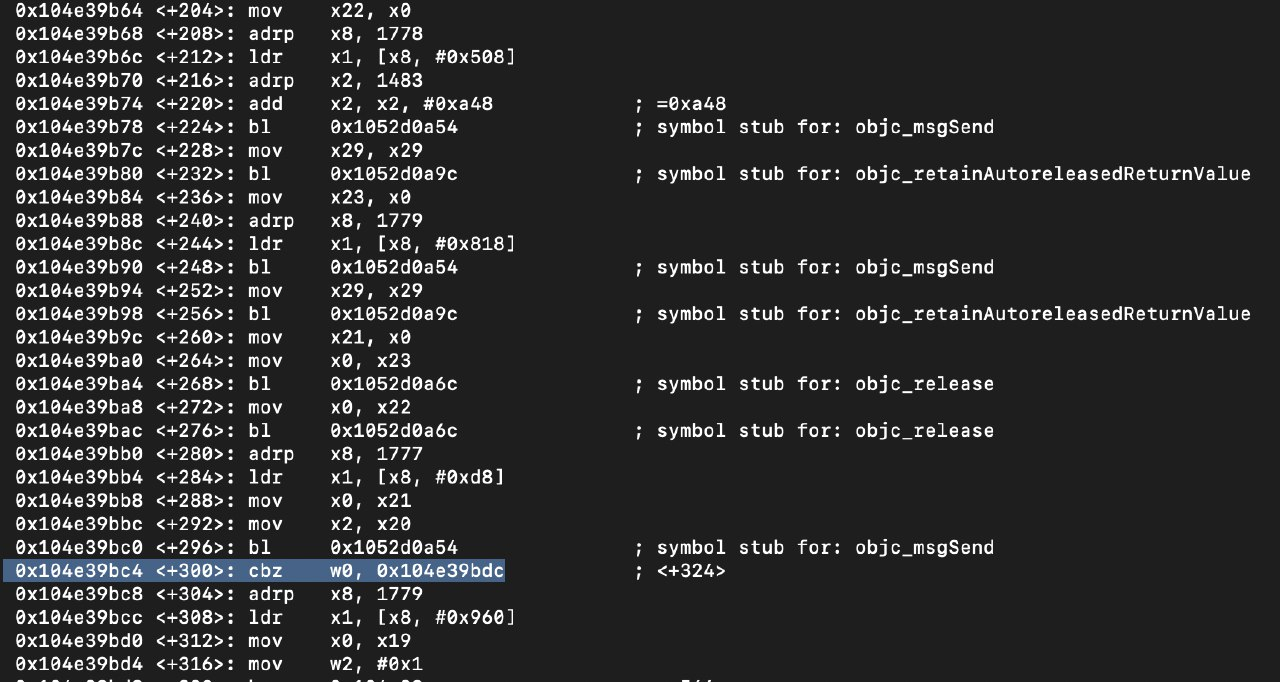


Figure 6. Last Pass Pin comparison through isEqualToString

**Proof of Attack**

1. First, I have to jailbreak the “clean device” and setup my debug server on my iPhone.
2. I then SSH into my iOS device and debug on LastPass by typing
   1. “debugserver localhost:6666 -waitfor "LastPass"
3. I then setup my debugger to be ready to debug this application
   1. iproxy 6666 6666
   2. lldb
   3. platform select remote-ios
   4. process connect connect://localhost:6666
4. Once the application is being debugged, I set a breakpoint on the function
   1. b \_\_\_lldb\_unnamed\_symbol1597$$LastPass



1. Enter a wrong pin code and the breakpoint would hit. Next, do a disassembly and identify the return result for the string comparison isEqualTo. 
2. Register W0 would hold the return result of isEqualTo comparison. Since we entered a wrong pin code, the result will always be 0x0. Do a register write of 0x1 to w0. 
3. Continue the application and you will be able to access all the saved passwords in the vault.

# Mitigation against security vulnerabilities discovered

Today, I have demonstrated how LastPass can be broken on the default settings. For the users of LastPass, how could they better protect themselves?

1. Use the log out feature on LastPass. I have tested this functionality and it seems secure enough for now.
2. Setting a pin on your mobile device. Before an attacker can actually launch any attacks on your stolen iPhone, he would need to have physical access to an unlocked phone.