Fraunhofer
SIT

iOS Keychain Weakness FAQ

Further Information on iOS Password Protection

Jens Heider, Matthias Boll

Fraunhofer Institute for
Secure Information Technology (SIT)

May 6, 2011

Revision history

1.1 2011-05-06

updated: 2.1 Which versions of iOS are affected by the attack method?, p. 3
(iOS Firmware 4.3.3 added to affected versions)

1.0 2011-04-20 (First version)

Abstract

This paper summarizes answers to frequently asked questions about the iOS keychain weakness described in [HB11]. It provides further information for security evaluators on the impact of the findings.

1 Introduction

The *keychain* in Apple iOS devices is intended as secure storage for sensitive user data that should be protected even if an attacker has physical access to the device. Therefore, the keychain stores secrets for services used by the operating system (e.g., WiFi passwords, VPN credentials, etc.) but also credentials stored by 3rd party apps.

The protection of the keychain is of vital concern for device users, as an unauthorized access to contained passwords, keys and digital identities may be used to access sensitive data, to cause loss of data and may inflict financial loss.

With the previous publication on *Practical Consideration of iOS Device Encryption Security* the risks should be highlighted that accompany losing a locked iOS device regarding confidentiality of passwords stored in the keychain [HB11]. The paper presented results of hands-on tests that showed the possibility for attackers to reveal some of the keychain entries. For the described approach, the knowledge of the user's secret passcode was not needed, as the protection provided by the passcode was bypassed.

As intended, the publication of our findings has provided the possibility for a public discussion on the iOS keychain security. Also others have found weaknesses in the keychain concept, but seem to have chosen to keep this for their own [For11]. However, the general reaction we observed as the result of the publication was very positive. Many requests on further details and other aspects have provided the opportunity to increase the knowledge about the provided protection.

This paper should provide answers from our perspective to many important questions arisen from the initial publication. As for the initial publication, the intention is to provide as much insights as possible to enable others to understand and evaluate the impact of the observed security design, but without causing too many benefits for the wrong hands.

2 FAQ

2.1 Which versions of iOS are affected by the attack method?

As of this writing, all versions of iOS 4.0 up to (and including) iOS 4.3.3 are affected by the described attack method. For these versions at least the entries

marked "w/o passcode" in table 1 of [HB11] are vulnerable in case of an attacker with physical access to the device.

However, in iOS 3.x and prior versions *all* entries of the keychain can be accessed by attackers. In these versions all keychain entries are encrypted *only* with the device key accessible on the device. Therefore, for those devices, the knowledge of the passcode is not needed to decrypt all keychain entries.

2.2 Are also the credentials of App XY affected?

This depends on how the app stores the credentials and how it uses the keychain. If the developer chose an alternative way of storing the user's credentials (instead of using the keychain), it depends on the strength of this proprietary protection.

If the developer chose to store the data in the keychain and instructed the system to make the stored credentials available even when the device is locked by setting the accessibility to *kSecAttrAccessibleAlways* (cp. [App10a]), then the app is affected. Otherwise the credentials of the app are *not* affected by the attack method.

2.3 Are X.509 certificates also affected?

Yes, certificates stored in the keychain by the OS or apps can be affected by the weakness as well. Like for passwords the accessibility depends on the applied protection class (see section 2.14).

2.4 Do iOS configuration profiles provide a better protection than manual settings?

No. After reception of the configuration profile, the device stores the credentials in the data structures as if entered manually in the setup dialogs.

2.5 Are credentials in encrypted iOS configuration profiles protected?

No. Configuration profiles of the iPhone Configuration Utility (iPCU) can be exported with the setting "signed and encrypted". This is meant purely to secure the channel between iPCU and the device. If the device has the matching CA certificate to verify the signature it decrypts the configuration file and installs it. The settings and passwords from the configuration profile are now stored like any other in the keychain. This means the export settings of the iPCU do not provide any additional security on the device and against the described attack method.

2.6 Do you provide the scripts used in the demo?

No, please note that we do not publish our scripts, parts of it or further details on the used system functions. The only exception is made for law enforcement agencies with direct authorization of the judiciary.

2.7 What data may get lost with the described attack method?

We used a so-called *custom ramdisk* for the jailbreak. This technique leaves everything in place on the iOS device, so no user data gets lost. After mounting the filesystem of the device, all files are accessible. Some entries in these files are encrypted additionally. In our publication we showed that it is possible to decrypt some of these encrypted items of the keychain without knowing the passcode.

2.8 Can changing the root password prevent the attack?

No. See also Section 2.9.

However, if the device is already jailbroken, it is of course important to change the passwords for account *root* and *mobile*. Otherwise attackers can also perform serious remote attacks via wireless networks.

2.9 Does jailbreaking / modifying a device prevent an attacker from accessing secrets in the keychain?

No, the access possibilities to the iOS via a custom boot are that low level that an attacker can revert all changes to the device during the jailbreak step without loosing user data.

In the case of a changed root password from the default "alpine" to something unknown to the attacker, simply installing an SSH certificate (together with the SSH server) during the jailbreak will do. Also other device modifications (e.g., changed SSH configuration, changed file permissions, etc.) can not prevent an attacker from reverting these changes during the jailbreak step as the attacker possesses in this step already root privileges.

In the meantime we have changed our scripts to work also on already jailbroken devices to demonstrate that this is no protection. Even worse: on a jailbroken device a trojan app downloaded from e.g. Cydia store can read *all* keychain entries (and sent it to the attacker) when the user has started the app (not only those listed vulnerable in the table of our publication).

2.10 Did you inform Apple Inc. prior publication?

Yes. Our mission is helping industry to understand security, and work with industry in developing security solutions. Our goal is not publishing hacks, but rather to be on the constructive side. We are well aware that the world benefits more from a constructive approach to security than from quickly publishing hacks.

This work came out of client requests to analyze the security of the Apple iOS, which is a major concern for many enterprises. As pointed out in our report, the problem is caused as a direct consequence of a trade-of between functionality and security.

We're in contact with Apple regarding this issue since September 2010 and of course they were aware of the possibilities an attacker has after jailbreaking a device.

Prior publication we contacted Apple again, but did not receive a comment on how or if this will be addressed in the future.

2.11 Did Apple Inc. announced a timeline for mitigations?

As of this writing we're not aware of planned changes.

2.12 Is there a patch available?

No, as of this writing we're not aware of a patch or configuration option that changes the observed behavior.

2.13 What are mitigation options from your experience?

The possibilities for protection depends on the use cases, the installed apps and the environment in which the iOS device should be used and protected. When accepting to reduce functionality, possible mitigation options are for example to switch from MS Exchange to IMAP / SMTP / ICAL servers (to prevent the vulnerable push passwords) and refrain from using affected VPN and WLAN configurations by using other protection mechanisms. However, this may require major changes to the infrastructure that should be done carefully to prevent other flaws.

For organizational measures please also refer to the conclusion section in [HB11].

2.14 I'm an app developer. How can I protect the credentials of my apps?

Set the keychain item accessibility value to *kSecAttrAccessibleWhenUnlocked* or kSecAttrAccessibleAfterFirstUnlock. This prevents keychain access without knowledge of the passcode. If you do not specify the setting, the current default is kSecAttrAccessibleWhenUnlocked, but to prevent being affected by future changes of this default you should explicitly define your protection class. Refer to Apple developer portal [App10a] for further details.

2.15 Is the iOS keychain in general insecure?

No, the iOS keychain can provide sufficient security for stored items, if *both* of the following conditions are met:

- Items are stored with a protection class that makes them only available when the device is unlocked. (see Section 2.14)
- A strong passcode of 6 alphanumeric digits is enforced (reduces the risk[1] of brute-force attacks).

2.16 Some passwords are not accessible with the shown method. Doesn't this prove the general security of the used concept?

Well, though it is indeed good news to find passwords e.g. stored by the Safari browser to be encrypted with the passcode[2], in the field the threat to a user is exposed by any available stored secret. Especially the available passwords for email accounts can be exploited to gain access to other accounts via common password reset functionality. Additionally, it is also debatable how many different passwords really are used by common users. Of course, users are instructed to refrain from using the same passwords over and over again, but this advice cannot be enforced.

Therefore, from our perspective, the problems with the concept that threatens users with higher security demands are

- the non-configurable trade-off between functionality and security,
- the unavailability of security informations about the actual provided protection for specific keychain items (caused by the differences of the protection classes),

1 according to [App10b]: ≈50ms per password try; → ≈20 tries per second; → ≈1.7 years for a 50% change of guessing the correct passcode for a 6-digit alphanumeric code with base 36. The standard simple code of 4 numeric digits would be brute-forced in less than 9 minutes. Based on the assumption that the counter for wrong tries in the iOS can be bypassed, as it is not hardware-based.

2 See table 1 in [HB11]

- and the non-disclosure of the actual protection scheme preventing efficient in-depth evaluation of the protection strength.

References

[App10a] Apple Inc. Keychain item accessibility constants. http://developer.apple.com/library/ios/documentation/Security/Reference/keychainservices/Reference/reference.html#//apple_ref/doc/constant_group/Keychain_Item_Accessibility_Constants, September 2010. 4, 7

[App10b] Apple Inc. WWDC 2010, Core OS, Session 209 "Securing Application Data", Slide 24. http://developer.apple.com/videos/wwdc/2010/, 2010. 7

[For11] Chris Foresman. Security expert: iPhone password hack shows flawed security model. http://arstechnica.com/apple/news/2011/02/six-minute-keychain-hack-highlights-busted-iphone-security-model.ars, February 2011. 3

[HB11] Jens Heider and Matthias Boll. Lost iphone? lost passwords! practical consideration of ios device encryption security. http://www.sit.fraunhofer.de/Images/sc_iPhone%20Passwords_tcm501-80443.pdf, February 9 2011. 3, 4, 6, 7