Fraunhofer
SIT

Lost iPhone? Lost Passwords!

Practical Consideration of iOS Device Encryption Security

Jens Heider, Matthias Boll

Fraunhofer Institute for
Secure Information Technology (SIT)

February 9, 2011

Abstract

The paper highlights risks that accompany losing a locked iOS device regarding confidentiality of passwords stored in the keychain. It presents results of hands-on tests that show the possibility for attackers to reveal some of the keychain entries. For the described approach, the knowledge of the user's secret passcode is not needed, as the protection provided by the passcode is bypassed.

1 Introduction

During many discussions about security of iOS devices (iPhone, iPad) we have recognized that the public perception of protection strength provided by the *iOS device encryption* does not reflect all aspects of the security for stored passwords, which we consider important. Therefore, this paper is intended to point out related facts, which were elaborated by hands-on tests to clarify the strength of the protection in a lost device scenario.

When an iOS device with hardware encryption capabilities is lost or stolen, many users believe that there is no way for a new owner to access the stored data — at least if a strong *passcode*[1] is in place. This estimation is comprehensible, since in theory the cryptographic strength of the AES-256 algorithm used for iOS device encryption should prevent even well equipped attackers. However, it was already shown[2] that it is possible to access great portions of the stored data without knowing the passcode. Tools are available for this tasks that require only small effort. This is done by tricking the operating system to decrypt the file system on behalf of the attacker. This decryption is possible, since on current[3] iOS devices the required cryptographic key does not depend on the user's secret passcode. Instead the required key material is completely created from data available within the device and therefore is also in the possession of a possible attacker.

Less considered is the aspect that, as an extension to the ability to decrypt the file system, an attacker may aim at gaining access to stored secrets kept in the *keychain*. Therefore, the impact of extending the known iOS weaknesses by targeting the keychain security should be shown in this paper.

In current versions of iOS, the keychain contains user accounts including passwords such as email, groupware, VPN, WiFi, websites and often also passwords and certificates used in 3rd party apps. As these secrets are stored *encrypted* in the keychain, the questions is: Which key is used for the encryption and which practical barrier does it create for an attacker with access to the device.

1 A user-defined secret that has to be entered to unlock the device
2 See e.g., `http://www.zdziarski.com/blog/?p=516`
3 As of this writing this refers to iOS firmware 4.2.1

2 Scenario Setup

For evaluating the practical strength of iOS device encryption security, we assume an attacker with physical access to the device, e.g. accomplished by theft or when finding a lost device. The assumed device is protected with a strong passcode, which is unknown to the attacker. The complexity of the passcode does not play a role for this evaluation, but is assumed to prevent an attacker from gaining access by simply guessing. Also, it is assumed that the device has *not* been jailbroken and so all original iOS protection mechanisms are in place.

When the device is found, it is assumed to be in the locked[4] state with activated *data protection*[5]. An unlocked device would provide the possibilities for user space exploits and could reveal more secrets. However, this leakage could not be accounted to the protection mechanism we wanted to evaluate.

The attacker's PC used to gain access to passwords has *not* been synchronized with the attacked device before. Therefore no secrets can be used by the attacker that are created between the owner's PC and his device.

In the described situation, device encryption commonly should provide protection against attacks from the outside. If the device is still turned on — e.g., not run out of battery meanwhile —, we assume that no remote wipe[6] command was received in the meantime (e.g, theft remained unnoticed, no network connection, etc.). In any case, the attacker turns off the device and removes the SIM card to prevent a further remote control.

In this described state, we have conducted our tests with iPhone 4 and iPad Wi-Fi+3G hardware with the latest firmware 4.2.1.

3 Approach

We have conducted various hands-on tests to evaluate which secrets can be revealed from the keychain and how much effort this might take an attacker. We focused on gaining access to the key used for the keychain encryption without having to open the device and without having to know the user's passcode. As we assumed that the operating system does have access to this key, we have chosen to access the operating system via an alternative boot procedure.

4 The enter password dialog is shown when trying to wake or power up device.

5 A feature of iOS 4 to put an extra layer of security on some stored data. See `http://support.apple.com/kb/HT4175`

6 A feature of iOS 4 to remotly remove the cryptographic key via *MS Exchange* or with the service *mobile me* from a device. See `http://manuals.info.apple.com/en_US/Enterprise_Deployment_Guide.pdf`

In the following, we describe the general approach we use to access the keychain. An additional video[7] illustrates the necessary interaction between PC and iPhone. The intention is to provide enough insides to enable others to understand and evaluate the impact of the observed security design. On the other hand, we will not disclose the created script nor other details to prevent too much benefits for the wrong hands.

The general approach we take can be summarized in three steps:

1. Getting access to the file system.
2. Coping keychain access script to file system.
3. Execute script which reveals stored accounts and secrets

The first step is needed to be able to access the keychain database. This step depends on the device's iOS version and hardware but in general can be achieved with a jailbreaking tool and by installing an SSH server on the device without overwriting user data. Now software can be launched unrestricted on the device. This way the software can access all files including the keychain database. Secrets in this database are encrypted with the device's key, which could not be extracted from the device. However, the key can be used from software within the device.

In the second step, we copy our keychain access script to the device via the SSH connection. It uses system functions to access the keychain entries, which made it not necessary to reverse engineer the encryption mechanism of the keychain items.

The last step executes the script, which outputs the found accounts to the shell screen. A sample output of the script is shown in Figure 1. Secrets that can not be decrypted by the system would be reported as *(null)*.

4 Results

After using a jailbreaking tool, to get access to a command shell, we run a small script to access and decrypt the passwords found in the keychain. The decryption is done with the help of functions provided by the operating system itself.

Our script reveals the always unencrypted settings (e.g., user name, server, etc.) for all stored accounts. For the account types marked "w/o passcode" in Table 1, also the account's *cleartext* secrets are revealed. This indicates, that an attacker would *not* need to know the user's passcode nor does he would need to exploit new vulnerabilities to reveal these secrets. The results were taken from a passcode protected and locked iPhone 4 with current firmware 4.2.1. The

7 See `http://www.sit.fraunhofer.de/forschungsbereiche/projekte/Lost_iPhone.jsp`

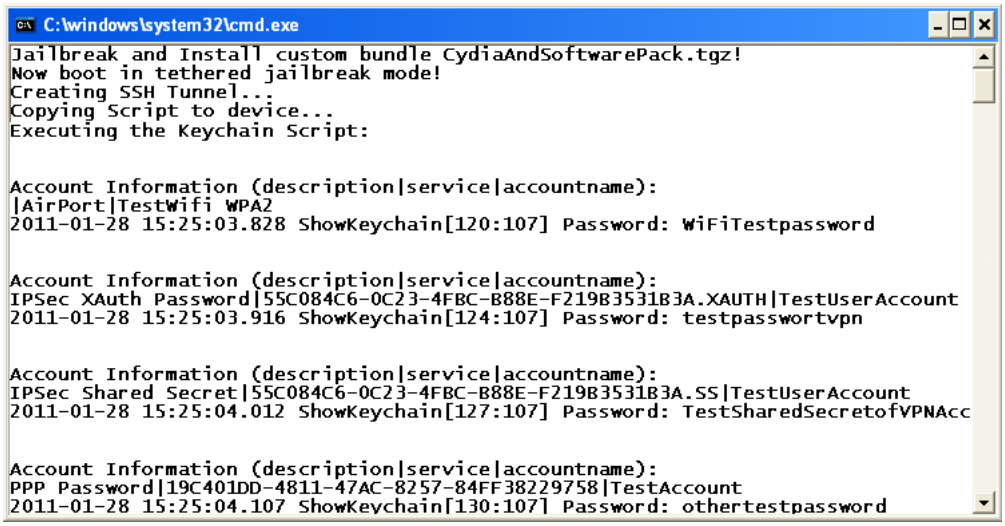

Figure 1:
Screenshot of Proof of Concept approach with truncated Output of revealed Passwords

overall approach takes six minutes, which might provide an additional opportunity for an attacker to return the device to the owner to cover the revealing of passwords.

Secrets within other protection classes, such as passwords for websites, could not be revealed in our lost device scenario. In our proof of concept implementation, these secrets — marked "protected" in Table 1 — were available to the script only after entering the passcode to unlock the device, which by assumption should not be possible for an attacker. However, note that in many situations it is sufficient for an attacker to gain access to the user's email account for abusing the password recovery processes of many other accounts.

The accessibility of keychain secrets without requiring the passcode is considered a result of a trade-off between system security and usage convenience: The passwords for network related services should be available directly from device startup, without having to enter the passcode first. In consequence the knowledge of the user's passcode is also not necessary for an attacker with access to the file system.

Further considerations are needed for the revealed service passwords such as Voicemail, iChat or Mobile me. The impact on these services and the possibilities that can be gained by attackers will be addressed in future research.

5 Conclusion

The results show that a lost iOS device may endanger also the confidentiality of data that is not stored on the device, but which is accessible for an attacker via the revealed stored secrets. This is not specifically a problem only to iOS devices, as other smartphone operating systems may also have circumventable password

Table 1: Test Results regarding Availability of Secrets to Attackers in the Lost Device Scenario

Tested Account Types	Secret Type	Accessibility
AOL Email	Password	protected
Apple Push	Certificate + Token	w/o passcode
Apps using keychain with default protection	depends on App	protected
Apple-token-sync (mobile me)	Token	w/o passcode
CalDav	Password	w/o passcode
Generic IMAP	Password	protected
Generic SMTP server	Password	protected
Google Mail	Password	protected
Google Mail as MS Exchange Account	Password	w/o passcode
iChat.VeniceRegistrationAgent	Token	w/o passcode
iOS Backup Password	Password	protected
LDAP	Password	w/o passcode
Lockdown Daemon	Certificate	w/o passcode
MS Exchange	Password	w/o passcode
Voicemail	Password	w/o passcode
VPN IPsec Shared Secret	Password	w/o passcode
VPN XAuth Password	Password	w/o passcode
VPN PPP Password	Password	w/o passcode
Website Account from Safari	Password	protected
WiFi (Company WPA with LEAP)	Password	w/o passcode
WiFi WPA	Password	w/o passcode
Yahoo Email	Token + Cookie	protected

protection mechanisms. However, iOS devices with device encryption may keep users in false believe that these devices have in general a strong password protection in place.

Regrading the iOS compliance to individual enterprise security policies, especially the sometimes applied comparison to fully encrypted notebook harddisks with pre-boot authenticaion is not valid, since these systems use the user's secret for the device encryption. Instead for iOS devices, the protection of vulnerable account types listed in Table 1 — and great portions of other stored data — are not bound to the knowledge of the user's passcode.

We judge the effort for the shown attack method as low, since the used jailbreaking tools are freely available and the additional steps to decrypt the keychain requires only moderate programming skills.

Owner's of a lost or stolen iOS device should therefore instantly initiate a change of all stored passwords. Additionally, this should be also done for accounts not stored on the device but which might have equal or similar passwords, as an attacker might try out revealed passwords against the full list of known accounts.

Enterprises should create efficient processes for lost device incidents to shorten the time during which their accounts may be vulnerable. Especially the change of group passwords like sometimes used for VPN and WiFi may require an additional effort but should be taken seriously.