WHITE PAPER:
APPLE IOS SECURITY IN THE ENTERPRISE

Apple iOS Security in the Enterprise

Who should read this paper

The whitepaper is aimed at IT professionals wanting to understand the Apple iOS security framework and technical capabilities more in detail. The content also provides guidance on how to integrate your existing infrastructure with functionality built in to iOS devices. It will also provide an overview of the mobile threat landscape.

written by Oliver Karow, Symantec Germany GmbH

Content

Introduction 1

The iOS Security Model 1

Mobile Device Management 6

Top threats targeting mobile devices 8

Security Vulnerabilities & iPhone Attacks 10

Jailbreak 12

Summary 13

Sources 13

Introduction

What you will learn

- Details of the iOS security model
- Overview about today's mobile security threats
- The capabilities of Mobile Device Management
- Common business models of a malware developer
- Information about the latest attacks against iPhones

What you should know

- Basic knowledge about Apples iPhone and iPad
- Basic knowledge about encryption and usage of certificates
- Familiar with it security terms

Whether supplied by the company or owned by employees, smartphones and tablet PC's are making their way into the enterprise. One of the important vendors to consider when talking about mobile devices is Apple, with its iOS based iPhone and iPad platforms. Their early versions were mainly designed for private usage, and therefore missing fundamental security mechanisms which are required for use within enterprises. Understanding these requirements, Apple did some homework and added a lot of functionality, helping companies using Apple devices in a secure fashion. This article describes the security mechanisms available on iOS including strengths and weaknesses, and demonstrates how a company can adopt these mechanisms to keep up with the latest security threats targeting mobile devices.

Enterprise Requirements

Following the trend of mobility, companies have to think about a solid mobile strategy, including security and device management. Companies usually spend a lot of resources to protect their valuable assets. On one hand, they would like to prevent putting assets at risk by introducing new technology like Smartphones and Tablet PC's, and on the other hand, they strive to enable their mobile users. In order to maintain or even improve their level of security, a mobile device platform has to meet the companies' security requirements and should be compliant to security standards. If a company has to ensure, that its mobile devices are secure and compliant over time, they need a way to manage hundreds or even thousands of devices in a uniform way. This is where Mobile Device Management (MDM) comes into play. While it seems to be mainly a cost consideration to have a capability of managing large numbers of devices from a centralized management platform, a solid MDM solution is key to achieve mobile device security.

The iOS Security Model

In comparison with other mobile device platforms like Android or Windows Mobile, Apple designed their operating system to provide security without the need of (or in many cases even the possibility to use) 3rd party security products like antivirus- or encryption software. In order to realize this approach, the underlying security model is based on the four pillars; *Device Security*, *Data Security*, *Network Security* and *App Security*, which are intended to protect against all known mobile security threats. Below I will describe the basic concepts behind each pillar. Throughout this document I will use the term iPhone, for both iPhone and iPad, as both device types are built upon the iOS operating system, where security mechanisms are embedded.

Device Security

When talking about device security, the first line of defense against unauthorized access to an iPhone is the device pass code, which can be used to lock and unlock the phone. Unfortunately, a user is by default neither forced to activate the pass code lock of the device, nor

to configure any other pass code related settings. To increase the physical access security, Apple implement pass code policy capabilities, containing features like activation of pass code lock and locking of the device after a predefined period of inactivity. Further pass code related settings, like minimum length, usage of a complex character set, aging and history are as well available, as a maximum number of failed login attempts, which will lead to a local wipe of the device, if exceeded.

Furthermore, with device restrictions it is possible to define which resources of the iPhone a user can access. This enables a company to align the device usage to its business needs, where one example could be disabling the camera for employees in the Research & Development department. Or by preventing the installation of additional apps or the usage of potentially dangerous services like Apple's iCloud that stores backup data in the cloud. For a list of possible features which can be enabled or disabled via a device security policy.

Feature (true/false)	**Comment**
Install or Update Apps	Disables Appstore. Users are unable to install and update Apps.
Enable or disable Siri	Disables Siri
Allow camera	Disables camera. Users are unable to take photographs
Allow Explicit Content	Hides explicit music or video content purchased from the iTunes Store.
Allow Screenshot	Users are unable to take screenshot of the display
Allow Youtube	Disables Youtube app
Allow iTunes	Disables iTunes Music Store
Force iTunes Store Password	Forces user to enter iTunes password for each transaction (iOS 5)
Allow Safari	Disables Safari web browser.
Allow untrusted TLS Prompt	Automatically rejects untrusted HTTPS certificates without prompting the user (iOS5)
Allow Cloud Backup	Disables backing up the device to iCloud
Allow Cloud Document Sync	Disables document syncing to iCloud
Allow Photo Stream	Disables Photo Stream (iOS5)
Force encrypted Backup	Forcing encrypted backup with iTunes

Table 1 - Device restrictions

The pass code policy, the device restrictions, as well as other settings described within this article can either be configured directly on the iPhone, or be included in a configuration profile and then deployed to a device. Configuration profiles are xml-based files, which can be created with either the *iPhone Configuration Utility*(iPCU), a free tool provided by Apple, or via any commercial mobile device management solution. I will describe policy deployment and mobile device management in more detail in a dedicated section of this article.

Data Security

If a mobile device gets lost or stolen, the financial effort to replace it is a few hundred dollars and would in most cases not even be considered in a companies' risk evaluation. The real damage is the potential leakage of confidential data, which is stored on or accessible via the mobile device.

To protect the data stored on the device iPhone has a built-in hardware based 256-bit AES encryption, which is running in background, encrypting and decrypting the complete flash drive real time. The original purpose of this device encryption was to enable the local and remote wipe functionality. Because the encryption key, which is unique for each device is stored on the device itself, it is sufficient to "throw away" this key, in order to make all stored information inaccessible. Actually, this is how the wipe mechanism is implemented on iOS devices.

In addition to encryption of the drive, Apple provides two further encryption mechanisms; the keychain and file encryption, which are accessible to app developers via the API.

The keychain is an encrypted container, using a 128bit AES algorithm, and can be used from apps to securely store data like passwords and certificates. From the user's point of view, a keychain provides transparent authentication. After unlocking the device with the passcode, the user does not have to log in separately to any app or service whose password is stored in the keychain (see - Keychain). For security reasons, each app only has access to its own keychain entries. This access is controlled by the operating system via entitlements, ensuring that an app does not have access to other apps confidential data.

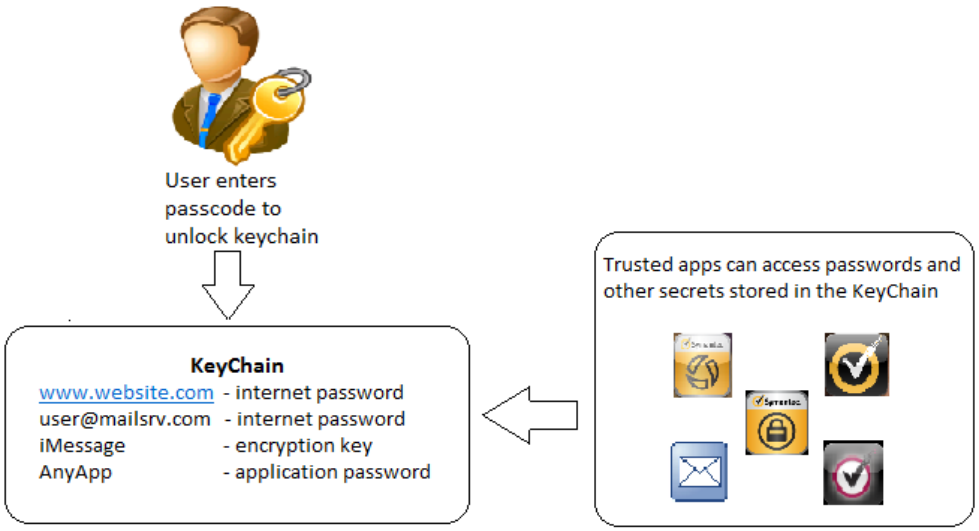

Figure 1. KeyChain

Another level of protection is the possibility to encrypt files on the iPhone. There is no configuration option available on the iPhone to enable or disable file encryption. Instead, Apple provides an API for app developers, which allows them to create and work with encrypted files, using the extended attribute key *NSFileProtectionKey*.

App Security

Working with an iPhone becomes interesting by the usage of apps. While Apple ships its hardware with some preinstalled apps like the Safari web browser and an email client, there are 3rd party apps available on the Appstore for almost every need. The various apps are offering attractive functionality or entertainment and are widely installed and used.

To avoid compromised iPhones caused by installation of vulnerable or malicious apps, a sandboxed approach and mandatory code signing are the core elements of iOS' app security. There is a vertical isolation between apps and a horizontal isolation between apps and the operating system (as illustrated in Figure 2). This means that apps do not have direct access to other apps data or to operating system resources like file system and kernel. If an app wants to communicate with other apps or the operating system, it can only do so by using the APIs and services provided by iOS. This architecture makes it virtually impossible for example to install a kernel mode malware from within the user space, as it would generally be possible on common PC based operating systems. As a result of this architecture, also 3rd party security applications are limited to work only in their own sandbox. If we would apply this to an antivirus solution, it would mean that it can only scan and protect its own environment, as it has no access to other apps nor their data. It is the main reason for no such 3rd party apps being available for iOS.

It is important to note, that sandboxing does not prevent an application or service from being hacked, but it prevents a hacked application from "breaking out" of its environment. However, there is a vast amount of information, like unique ID and phone number linked to the device, address book and other information, which can freely be accessed by any application. A team of researchers figured out, that 55% out of 1407 tested apps were accessing some of this information and sending it either to the app developer or an advertising company [1].

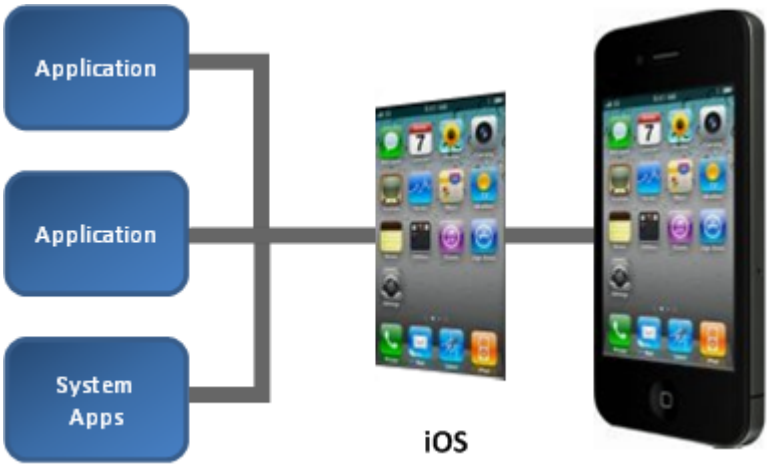

Figure 2. Sandbox

In order to ensure that only trustworthy applications can be installed and executed on an iPhone, each application is required to be digitally signed. Because each certificate is linked to a personal developer account, this potentially makes it possible to track back the developer of a malicious app. But more importantly, once an attacker is able to successfully exploit a vulnerability within an app, it is impossible to infect the binary of an application, as the signature test would fail when starting the app.

Also Apples strict process for publishing apps on their Appstore, and the fact that this is basically the only source for downloading apps, enhances the application security of the iPhone. Some MDM solutions also have app store capabilities and allow companies to install their apps without the usage of Apples AppStore. But for this article, this will be considered as a secure store, as well.

Network Security

Mobile users must be able to access their corporate data and network from anywhere in the world. Therefore it is very important, that users are authorized and the data is protected during transmission over Wi-Fi or cellular data network connections. As most companies already have a virtual private network (VPN) infrastructure in place, iOS supports the most common VPN protocols. In order to connect iPhones securely to the corporate network, secure tunnels can be established via IPSec, using the built-in Cisco client, the Layer-2-Tunneling Protocol (L2TP) and the Point-to-Point-Tunneling Protocol (PPTP). Also the use of generic SSL-VPN's is supported, including solutions from Juniper, F5 and Cisco. Depending on the VPN protocol used, iOS supports authentication via x.509 certificates or hard and software based tokens like *Symantec's VeriSign VIP Access* to allow strong authentication of users and devices.

To protect wireless access iOS supports the Wi-Fi Protected Access 2 (WPA2) and the IEEE 802.1x standards. While WPA2 uses 128-bit AES encryption to protect transmitted data, 802.1x provides an authentication mechanism to devices requesting to connect to a WLAN, in example by leveraging the RADIUS protocol, which is widely used within enterprises.

For Safari Web browser, Calendar, email Client and other apps communicating via the Internet, iOS provides the Secure Socket Layer Protocol v3 (SSL) as well as Transport Layer Security v1.2 (TLS) to securely transmit their data. In addition the S/MIME protocol is implemented to view and send encrypted email messages.

In case companies are using Microsoft Exchange ActiveSync for email and calendar synchronization or policy updates, the communication can be configured to use a 128-bit-SSL encryption and the client authentication can either be HTTP basic authentication, certificate based or both.

With iMessage a new messaging service introduced with iOS 5, users can exchange text messages, videos, contacts and other information between each other. If the communication counterpart does have an iPhone or iPad, the messages will be exchanged via Apple's Push Notification Service, which is using a 2048-bit TLS/SLL certificate to encrypt and authenticate. In case the receiver of a message does not have an iOS based device, the message will be send via Short Message Service (SMS), which by its nature is not encrypted and may pass different insecure provider networks before it arrives at the recipient.

Policy Enforcement & Security Policy

As mentioned, multiple iPhones can be configured via xml based configuration profiles. Within the configuration profile the administrator can remove the permission for the user to uninstall the profile. This way a company can enforce the configuration settings and the company device security policies.

Configuration profiles are usually deployed over-the-air. In most cases, this means that a user will get the profile either provided via email or for download from a web portal. In order to give the user the possibility to determine if the configuration profile was really issued by the corporate IT and configuration profiles can be digitally signed (see Figure 3. Profile install dialog).

As a profile can also contain confidential data, like VPN or Wi-Fi credentials, Apple also offers the possibility to encrypt the profile.

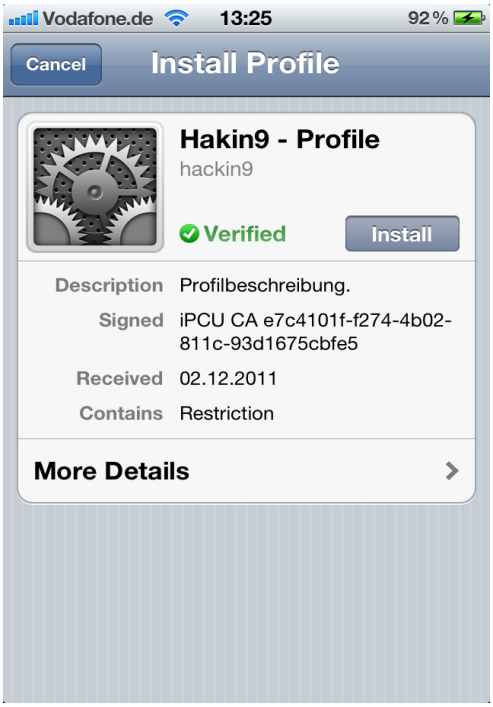

Figure 3. Profile install dialog.

Mobile Device Management

In order to manage a large number of iPhones, iOS supports Mobile Device Management (MDM) and therefore allows companies to centrally enroll and activate new iPhones, to manage their configuration and to deploy Apps and documents. Apples MDM functionality is based on *Configuration Profiles*, *Over-The-Air Enrollment* and the *Apple Push Notification Service*. While Apple offers the iPCU utility and the Apple Profile Manager to manage iPhones and iPads, there are different vendors on the market, offering MDM solutions, based on the interface provided by Apple. But since Apple only allows defined and limited access to the iOS platform, all MDM providers will provide merely the same features.

While MDM by itself is a huge topic and could fill a separate article, we will focus on the security related topics of device management.

Apple Push Notification Service

The management of iPhones takes place via a connection between the iPhone and the MDM server. A core mechanism for the communication between an MDM server and the iPhone is the Apple Push Notification Service (APNS). When the server wants to communicate with the device, for example in order to update or query the configuration, he will send a silent notification via the APNS. The iPhone will then communicate directly with the MDM server, to fetch the new commands, app or data. Because the APNS is running on servers, operated by Apple, a company has to open the port TCP/2195 to the APNS server “gateway.push.apple.com” and port TCP/2196 to the server “feedback.push.apple.com” on their corporate firewall.

From a security standpoint it is important to have static TCP ports and defined destination addresses, in order to be able configure narrow firewall rules. But surely more important is to protect the communication over the public Internet. To accomplish transport security and to reduce the risk of spoofing and man-in-the middle attacks, the communication between a company’s’ MDM server and Apples PNS Server is based on the SSL/TLS protocol, using a 2048-bit SSL certificate. This certificate must be requested and downloaded

from the Apple Push Certificate Portal. The communication between Apples Notification Server and the iPhone is protected via SSL/TSL, as well. *Figure 4. APNS communication scheme* illustrates the components within an APNS communication and the ports and protocols used.

The APNS is not only used for mobile device management. Notifications are the main way for apps to provide alerts and related information, like new messages on Facebook or Twitter.

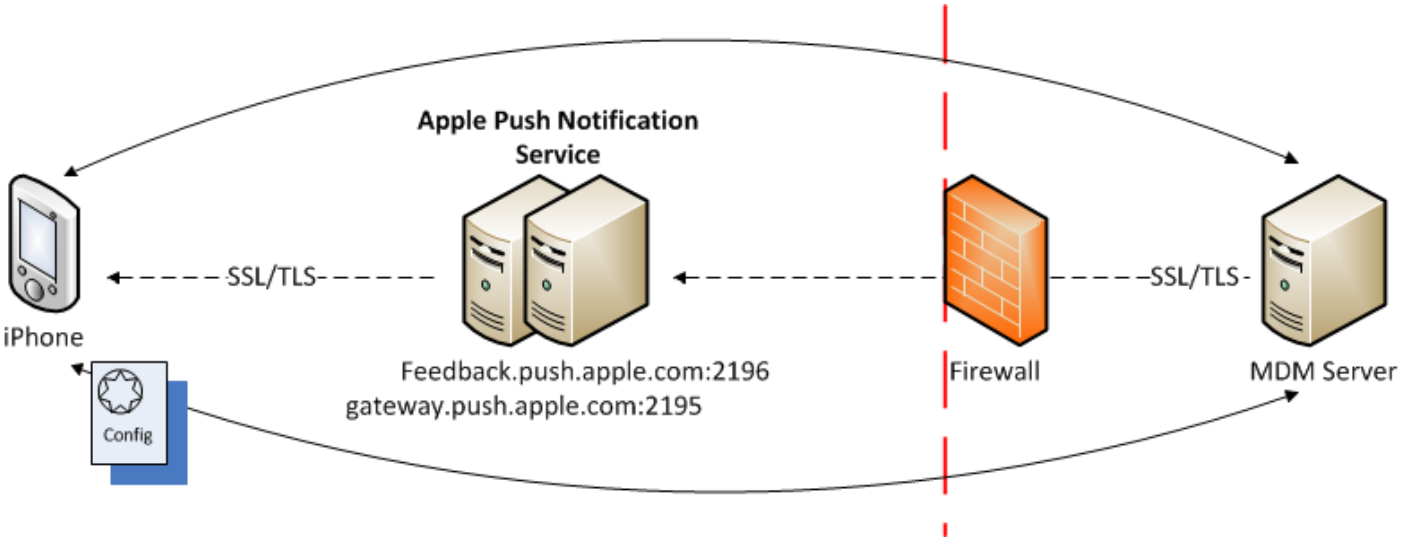

Figure 4. APNS communication scheme

iPhone Enrollment

Assuming that a MDM solution is in place, the first step a company has to take to integrate a new iPhone into its mobile management is to enroll it with the MDM server. The enrollment process creates a relationship between the device and the server, allowing it to be managed without further interactions of the user.

The enrollment of a new device will usually be done by delivering the enrollment profile via network connection. Most MDM vendors manage this by either making a web portal or a dedicated mobile management agent app available. In the first case, the user has to use the iPhone web browser and link it to the MDM Servers Web portal to start the process. In the second case, the user has to install and run the MDM vendors' app, which then will handle the communication with the MDM server. For example Symantec provides its Mobile Management Agent for download via Apple's Appstore.

The process of Over-The-Air Enrollment includes three phases; *UserAuthentication, Certificate Enrollment* and *Device Configuration*. In order to ensure that only authorized users can connect to the MDM server and enroll their device the User Authentication is an important part of the process. Therefore a user has to provide his credentials; either via the web portal or the mobile device agent (see Figure 5).

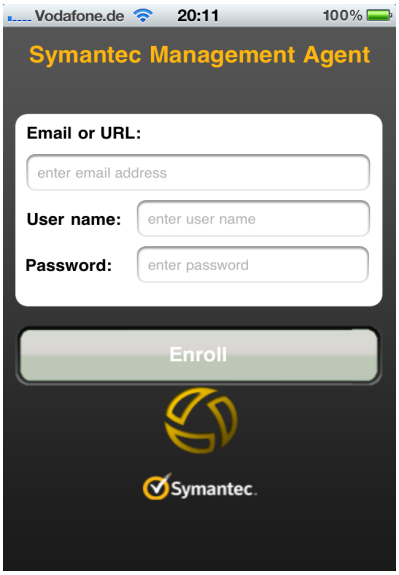

Figure 5. Authentication for enrollment.

After having successfully authenticated, the device obtains a signed x.509 identity certificate, which is used for encryption. To obtain the certificate the iPhone creates an asymmetric Public- and Private Key pair, sends the public part of the key to the enterprise Certificate Authority (CA) and in return gets the signed x.509 certificate. For this Certificate Enrollment process, the *simple certificate enrollment protocol* (SCEP) is used.

The last phase in the iPhone enrollment is the configuration. When an identity certificate is installed on the device can receive encrypted configuration information over the air. Configuration profiles and configuration updates can be delivered to the iPhone via email, web link for download, an MDM app or via Exchange ActiveSync.

Top threats targeting mobile devices

In order to understand how effective the security capabilities of a mobile device platform, a good starting point is to have a look at its resilience against today's mayor mobile threats [2]. When going through a complete threat modeling process would blast the article I will write about some of the major threats for mobile devices, independent of the devices platform or vendor.

Data Loss

For most companies, the loss of critical information, unintentional or malicious, is nowadays considered to be the biggest threat related to mobile devices. If a company provides iPhones to their employees, the intention is to enable them to work anywhere at any time. Therefore, users will store the information they need to do their work on their devices. In addition they will have VPN access to the corporate network. The information accessible on a lost iPhone may include customer data, financial statements, proposals and other information, depending on the business the user is in. Having up to 64 GB of disc space allows an employee to carry a lot of this data and by the nature of mobile devices, this information is now outside the well protected company network with all its preventive and detective security mechanisms.

For sure, there is some information a company does not want to be stored on mobile devices. But how can they prevent this information from being transferred to the iPhone, or even worse, from the iPhone to any other data repository? How can they prevent that a user is "backing up" this information by using cloud services like iCloud or Dropbox? And last but not least, what happens to the stored information, if the device gets lost or stolen?

Let's have a look at each of these use cases. We know that the physical access to an iPhone can be restricted with a pass code via configuration profile and even force a local wipe after a maximum number of failed logins. So a person who gets access to a company owned iPhone cannot simply access its content without knowledge of the pass code. In addition, the complete flash drive is encrypted, using a hardware based 256-bit AES encryption. Each app can store its files encrypted and the confidential data within the keychain is also encrypted. Finally, the encryption keys used are derived from the pass code, if data protection is enabled, making it harder to crack them. And last but not least, the company can initiate a remote wipe of the device. Those mechanisms, if implemented properly, can give sufficient protection against data loss on lost or stolen iPhones.

Using configuration profiles, a company can disable the usage of Apples iCloud service for data backup. They also can prevent the installation of additional apps like Dropbox, which offer cloud based data storage functionality.

It is more difficult for a company to prevent users from uploading confidential information to the iPhone, or sending such information from the iPhone to an external location or recipient. Installing a Data Loss Prevention (DLP) product on the iPhone would be useless, as this product would only be able to monitor its own communication, due to the sandboxing mechanism.

A working approach would be to force the iPhone, via configuration profile, to route all network traffic through the VPN tunnel into the corporate network, where a network Data Loss Prevention solution is monitoring all communication, including web up- and download and email traffic. This approach is used by *Symantec DLP for Tablets*.

Having the above mechanisms in place, can help a company to minimize the risk of data loss via iPhones.

Malware

Malware authors are mainly motivated by making money and therefore very innovative when developing new business models. Beside understanding the technology they are using to bring their malware onto the users devices, it is important to have an idea about what these business models look like, in order to successfully fight them. An analysis of the latest mobile malware [3] identified the following motivations:

- *Premium Rate Number Billings*: Attackers set up and register a premium-rate number. Infected mobile phones will call or send a SMS to this number and the caller is billed a premium rate.
- *Spyware*: Malicious apps that allow someone to track and monitor a user of a mobile phone. Such malware may record and export all SMS messages, emails, call logs, GPS locations or turn on the microphone.
- *Search Engine Poisoning (SEO)*: Such malware manipulates or "poison" search engines in order to display search results. Raising their search rank allows attackers to increase visits by prospective customers or generate revenue through pay-per-view or pay-per-click advertisements shown on the site.
- *Pay-Per-Click (PPC)*: Based on the internet advertising model, where advertisers pay a website owner, when clicking on a banner, a malware will initiate a huge number of clicks on such advertisements.
- *mTAN Stealing*: Online bank account hacking. Malware will intercept SMS from the bank, containing mTAN, in order to manipulate transactions. This also includes stealing of other confidential information like passwords and PIN's.

Malware can be split up into the three categories; Viruses, Worms and Trojan Horses. Classical viruses are "infecting" legitimate program files by attaching themselves. Once an infected program file is executed, it looks for further files on the system to infect. Computer worms spread themselves over the network, for example by sending a malicious file via email to exploit client site applications or network based services like Secure Shell (SSH) or Microsoft RPC. Trojan horse programs do not use spreading techniques, but instead perform malicious actions on the target system, like logging keystrokes or installing remote control functionality in order to grant access to attackers. Today's malware often uses combined techniques of Viruses, Worms and Trojan horses. This kind of malware is called Blended Threat.

A quick search at Symantec's Threat Explorer[4]shows that there are currently around 70 malware threats known for Android, about 220 for Symbian and mainly two which are targeting the iPhone platform. This two computer worms, named iPhoneOS.Ikee and iPhoneOS.Ikee.B were discovered in 2009 and spread over-the-air on jailbroken iPhones with a SSH default password attack. While iPhoneOS.Ikee changed the devices background wallpaper to display a picture of 80's pop-star Rick Astley, iPhoneOS.Ikee.B locked the screen of the device and displayed the text *"Your iPhone's been hacked because it's really insecure! Please visit doiop.com/iHacked and secure your iPhone right now!."* In order to unlock the infected iPhone, the user was required to pay a €5 ransom to the attacker's PayPal account.

Figure 6. iPhone.Ikee.B Message

Taking into account that the iKee worm only affects jail broken devices, we can say that iOS seems to be largely resistant against malware threats. This is obviously a result of the security architecture of iOS. Since apps are running in a sandbox and are digitally signed, a virus can neither infect other apps on the device, nor attach itself to an app, as this would change the apps signature, preventing it from starting. The isolation from apps against the operating system also limits most network based attacks, such as buffer overflows, from taking control of the iPhone. Also, it is by design not allowed for apps to send SMS or to initiate phone calls without user's acknowledgement, which will prevent the malicious use of premium phone numbers. Furthermore, Apples strict process for publishing apps on their App store, prevents spreading of malicious apps, as each app will be reviewed and app developers have to be registered, supplying their name and credit card data. While this is not bullet proof it sets another barrier for malware authors who normally want to stay anonymous.

Web based attacks

Web based attacks are usually launched by malicious or hacked legitimate websites. Once a user is accessing such a site, the web server is determining the version of the browser used and sending appropriate malicious content to the web browser, causing the browser to execute malicious instructions. If the web browser has been exploited successfully, it tries to install malware on the system, or steal information like passwords and TAN's, entered by the user or available through the browser.

Again, since iOS isolates each app from each other, if an attacker successfully compromises a browser app he might not be able to attack other apps on the device or the operating system itself. Instead, the attacker will be limited within the exploited web browser app. Having access to the browser app may give the attacker access to confidential information as a he can monitor all data sent and received through it, including passwords, credit card numbers and other confidential information. In addition, every app on an iPhone can access the system wide available unique device identifier, the address book, the safari browser search history and other information.

For web based attacks, iOS provides a good level of protection against web based attacks. However, an attack against a sandboxed application could provide access to confidential data and thus still cause significant harm.

Security Vulnerabilities & iPhone Attacks

As of the time this article was written, there were round about 200 different vulnerabilities discovered in various versions of the iOS operating system. One example of vulnerability potentially allows attackers to execute arbitrary code due to multiple memory corruptions in Apple's iOS FreeType implementation [5].

This is indeed not a small number of vulnerabilities and only verifies what most of you already know: There is no bug free software. The larger and more complex the code base of an application or operating system is, the higher the probability of programming mistakes, which may lead to security vulnerabilities. Interestingly enough, these vulnerabilities seem to be exploited by users rather for jail breaking their devices than for maliciously compromising other devices.

Considering this, it is more important for a company that the vendor is reacting promptly and is supplying patches for discovered vulnerabilities within an acceptable time frame. Currently Apple has needed an average of 12 days to patch each vulnerability once it was discovered. A company has to ensure that it gets informed about new vulnerabilities and available patches in order to ensure, via their MDM solution, that those patches get installed on the affected devices.

Apple iOS Code Signing Security Bypass Vulnerability

Let's have a look at a sample vulnerability, which was discovered by the well known security researcher Charlie Miller and published on the 7th of November 2011. Charlie discovered a flaw within the JavaScript engine "Nitro" which allowed him to download and execute additional code from within a signed app. Nitro has special permissions when run by Safari, that allow it to perform just-in-time (JIT) compilation of JavaScript code to native instructions and then execute them. The vulnerability affects all iOS versions beginning with v4.3, where Nitro was introduced.

As a proof of concept he developed a harmless looking app called "InstaStock" and submitted it to the Apple's App store review process, which it passed successfully. Once installed and run on an iPhone, InstaStock downloaded and executed additional code like playing a video from Rick Astley or opening a reverse shell, giving him access to the operating system. A video from Miller, showing his attack in action is available on Youtube[6]. The InstaStock app was meanwhile removed from the AppStore by Apple. Apple also revoked Millers developer account as a result of his research activities.

Because application signing is one of the most important cornerstones of iOS security, vulnerabilities like this could potentially have very serious impact to the security of the iPhone platform.

Bypassing hardware Encryption

In the part about Data Security within this article, I described the encryption mechanisms implemented on the iPhone, which should protect against unauthorized access to the information stored on the device. Unfortunately there is a way to access the data stored on an iPhone, even without knowing the pass code. If an attacker has physical access to an iPhone, he can start a Boot-ROM attack against it (see paragraph covering Jailbreak for details) and install a SSH daemon.

Because the key for the hardware based encryption is located in the memory and the device is still continuing to decrypt the data on the flash drive transparently, all files on the device are accessible via the SSH login and can be copied from the device.

Bypassing KeyChain encryption

But as we have learned before the hardware based encryption is only one layer of defense of the iPhone, another one is the keychain. While an attacker can access and copy the KeyChain's SQLitedatabase file, he can't simply access its content because it is encrypted with a different key than the one used for the drive encryption.

The way that researchers [7]figured out how to access the encrypted content of the KeyChain is also based on a precondition given by a jailbreak. To understand the attack we need a little bit more background on how access to the KeyChain is granted by the operating system.

Each application on an iOS device contains a unique application-identifier entitlement, which is added and digitally signed, before being submitted to the Apple App store. The keychain service controls the access of an application to the keychain based on this unique

entitlement and by default allows each app to only access its own data. In order to allow apps to share information between each other, Apple introduced the keychain-access-groups entitlement. If a developer for example has developed App-A and App-B and wants both apps to have access to each other's keychain content, he has to assign App-A and App-B a shared keychain-access-group. When accessing the keychain with one of his apps the developer has to specify which access group to use.

Figure 7. KeyChain Viewer

The way to read and write to keychain elements of other apps is to figure out which entitlement is required for access and sign an app with this access group. Unfortunately, the name of the required entitlement is stored unencrypted within the KeyChain's SQLite database file and can be extracted easily. As jailbroken devices have "disabled" the verification of digital signatures the only thing an attacker now need to do, is to sign the app himself.

An easy to use tool is KeyChainViewer from Jean-Baptiste Bédrune and Jean Sigwald, which is running as an app on jailbroken iPhones (see Figure 7. KeyChain Viewer)

Decrypting Files

The last encryption item I want to write about is the encryption of files on the iPhone using NSFileProtectionKey from the API. As described earlier, the file encryption has to be used from the developer of an app, by calling the appropriate API calls. However, if a file is encrypted, the encryption key is based on the pass code. As in most cases, users are using a four digit secret instead of a complex pass code, this is an ideal candidate for a brute force attack. In average, a successful brute force attack against a 4 digit pass code, which tries all 10000 possible combinations, takes only a couple of minutes on an iPhone. This attack by the way will not trigger the local wipe of the device, although the maximum number of failed logins is exceeded. Jean-Baptiste Bédrune and Jean Sigwald also provide a tool to conduct a brute force attack against iPhones[8].

Jailbreak

Jail breaking is the process of removing the limitations imposed by Apple on devices running the iOS operating system [9]. On a jailbreaked iPhone, a user has root access on the operating system and can beside many other tasks, install apps from stores others than the Apple Store. Easy to use tools like redsn0w, Spirit, Pwnage Tool and JailbreakMe are publicly available, including videos on how to use. There are two types of Jailbreaks available, tethered and untethered. A tethered Jailbreak is not persistent and requires the iPhone

to be connected to a Computer, each time it reboots, in order to boot into the jailbreaked operating system. Untethered Jailbreaks do not have this requirement, and stay jailbreaked after each reboot.

One method to jailbreak an iPhone is the Boot-ROM Attack. The boot-ROM is the first piece of code that is running, when starting the iPhone and is a read only memory. Therefore a security vulnerability found within the boot-ROM is something like the holy kraal for vulnerability researcher, as Apple can't simply fix it by supplying a patch and has to issue a new hardware revision.

Boot-ROM attacks can use the Device Firmware Upgrade (DFU) mode of the iPhone, which is accessible by pressing the home and power button in a certain sequence. Once an iPhone has booted into DFU-mode, an application can send a new firmware image to the device via USB. Hackers are using this mode to smuggle in code that is patching the devices operating system. By design, firmware files have to be digitally signed by Apple, in order to be accepted by the iPhone. But hackers already discovered flaws within the boot-ROM, allowing them to circumvent this restriction.

From a support and security standpoint, a company wants to avoid that users are using jailbreaked devices. Beside the fact that Apple understandably does not support jailbreaked iPhones, it also dramatically decreases the resilience of the iPhone against mobile security threats. Users can now install apps from even untrusted sources, which do not need to be digitally signed. This opens new possibilities for malicious apps. In addition, users could modify the configuration profiles, deployed by the company, thus making this device not policy compliant.

Many mobile device management solutions, like Symantec Mobile Management, have functionality to detect jail broken devices. They are using different approaches, like looking for Cydia, an app which by default is installed by the jail breaking tool Redsn0w. Or they try to write data outside of an applications' sandbox. Since Cydia is not available on the Apple Store, and writing outside of the sandbox is prohibited by an unhacked iOS, both methods are good indicators for a jail broken device.

Summary

The iOS security model seems to be well designed and thus has proven largely resistant to attack. It offers strong protection against traditional malware, primarily due to the sandboxing approach and Apple's rigorous app certification process and their developer certification process, which vets the identity of each software author and weeds out attackers.

Once an attacker has physical access to an iPhone, he has good chances to access the information on the device. Enforcing the usage of a complex pass code will increase the devices resilience against brute force attack, as already the usage of 6 characters instead of 4 digits will extend the time required for a successful attack, from 5 minutes to more than a year. Companies developing in house apps should take advantage of the file encryption capabilities which are provided by the iOS API.

In order to avoid data loss by lost or stolen iPhones, it is recommended to control the confidential information being stored on the device and prevent it from being send to external locations. Therefore integration with the corporate data loss prevention solution is recommended. Also the activation of the devices *kill-switch* may be a good idea, allowing conducting a remote wipe in case of loss. Companies should also make use of the extended security features provided by iOS and configure their devices accordingly. And last but not least, a solid mobile device management solution will help to manage large numbers of devices and to keep them secure.

Sources

[1] Egele, Kruegel, Kirda and Vigna, PiOS: Detecting Privacy Leaks in iOS Applications

[2] Symantec, A window Into Mobile Device Security

[3] Symantec, Motivations of Recent Android Malware http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/motivations_of_recent_android_malware.pdf?om_ext_cid=biz_socmed_twitter_facebook_marketwire_linkedin_2011Oct__androidmalwarewhitepap

[4] http://www.threatexpert.com

[5] Apple iOS FreeType vulnerabilities http://www.securityfocus.com/bid/50643/

[6] Miller http://www.youtube.com/watch?v=ynTtuwQYNmk

[7] Heider, Boll (Fraunhofer SIT)

[8] http://code.google.com/p/iphone-dataprotection/

[9] http://en.wikipedia.org/wiki/IOS_jailbreaking

About Symantec

Symantec is a global leader in providing security, storage, and systems management solutions to help consumers and organizations secure and manage their information-driven world. Our software and services protect against more risks at more points, more completely and efficiently, enabling confidence wherever information is used or stored. Headquartered in Mountain View, Calif., Symantec has operations in 40 countries. More information is available at www.symantec.com.

For specific country offices and contact numbers, please visit our website.

Symantec World Headquarters
350 Ellis St.
Mountain View, CA 94043 USA
+1 (650) 527 8000
1 (800) 721 3934
www.symantec.com

Symantec helps organizations secure and manage their information-driven world with **security management**, **endpoint security**, **messaging security**, and **application security** solutions.

4/2012