Securely Integrating iOS Devices into the Business Environment

GIAC (GSEC) Gold Certification

Author: Joshua Brower, Josh@DefensiveDepth.com
Advisor: Aman Hardikar

Abstract

Driven primarily by the end user, iOS devices continue to inundate businesses at an ever-increasing rate. Because these devices are housing sensitive organizational data, it is imperative that it is understood what risks to the organization are involved in allowing users to utilize these devices for business. Ascertaining what the risks are, and what the compensating controls would be, should be a critical component of any business risk assessment. The security features of the device itself, how applications are utilized on the device, and the actual usage of the device needs to be evaluated. Beyond the aforementioned areas, a major consideration that needs to be taken into account is whether the device is personally owned or business owned, as well as how it is managed, as these will be the primary factors by which controls are evaluated to manage the incurred risk. Finally, users need to be made aware of the risks, and trained in what their responsibility is to reduce the risk to an acceptable level.

1. Introduction and History of iOS

iOS is the name of Apple's mobile operating system. Previous to June 2010, it was known as the iPhone OS. Released in the summer of 2007, the first generation iPhone OS was a spectacular hit in the blackberry-dominated smartphone market. With the release of iPhone OS 2.0 the next year, Apple continued to improve on its original design with such features as support for enterprise use, remote wiping, and most importantly, the Apple App Store, the worlds first centralized software distribution platform for mobile devices. Version 3.0, released in the summer of 2009, brought more useful features: copy and paste text and photos, 3G tethering, and much more. (Dwivedi, Clark, & Thiel, 2010) By the time June 2010 came, and the wait for iPhone OS 4.0 was almost over, it was a quite a different world then 2007. Apple had released a number of other mobile devices, which had all been very successful: the iPod, iTouch, and iPhone all ran some form of the iPhone OS. So it made sense that the next version of iPhone OS would be renamed to the more generic "iOS," and still have the next version number of the iPhone OS (4.0). With the release of iOS 4.2 in November of 2010, all current generation Apple mobile devices run iOS4 or later, from the iPhone to the iTouch, to the iPad.

In early 2011, Apple announced that it had sold its one hundred millionth iPhone, and that in 2010, the first generation iPad sold fifteen million units. (Coldewey, 2011) With this kind of explosive growth of iOS devices came growing pains, and iOS has seen its fair share; the elusive white iPhone and the AT&T and iPad data breach are just a couple.

Now, with the rapid adoption of the iPhone and iPad into business environments comes much unknown and unforeseen risks. This paper will start scratching the surface of these risks and bring out some considerations that must be taken into account when an organization decides to allow their employees to use iOS devices for business. Understandably, some of the risks and considerations that will be brought out are common to all such mobile devices, but this paper will focus on managing those risks specifically for iOS devices.

2. Device Ownership & Management

2.1. End User Ownership of iOS device

The first area that needs to be considered is that of ownership of the iOS device. Indeed, this decision will color every control that might be considered to manage the particular risk that needs to be regulated. If it is a large organization, than more than likely there are already policies in place that either allow or prohibit the use of personal equipment for business use.

2.1.1. Business Owned

In this scenario, the business owns the iOS device, and allows the employee to use it for work and/or for personal use.

2.1.2. Personally Owned

In this scenario, the employee owns the iOS device, and they use it for both personal and business use, whether sanctioned by the organization or not.

2.2. How the iOS device will be managed

The second area that needs to be considered is that of how the iOS device will be managed. Like the previous issue (ownership of device), the decision made here will color every step of the device's lifecycle and management style.

2.2.1. Mobile Device Management

Using a Mobile Device Management solution (MDM) is ideal for medium to large organizations that are managing a couple dozen iOS devices up to tens of thousands. A MDM solution leverages the APIs that Apple has exposed in iOS to manage the iOS device's entire lifecycle, from initial deployment, to ongoing maintenance, to end of life.

2.2.2. Individually Managed

Those smaller organizations that cannot afford a MDM solution will have to manage their iOS devices individually. Through software like the iPhone Configuration Utility, which is released and maintained by Apple, organizations can manage their iOS devices more efficiently than ever, but will be a at major disadvantage as the quantity of iOS devices that they manage increases.

3. Risks and Controls: Device Management

The first major area that will be considered is general device management of iOS devices. First, let us look at the background a bit.

Historically, Apple has not marketed their products at the enterprise, preferring more of the consumer market. This is seen most acutely in a comparison between RIM's Blackberry smartphones and the iPhone. From the introduction of the Blackberry smartphone in 1999, RIM has provided enterprise management tools for businesses to manage the mobile devices they give to their employees. (vl, 2008) With this being the case, and with no other major business-minded smartphone in view, RIM has had the lion's share of the business smartphone market until now, when the iPhone & Android devices have started taking market share away from the Blackberry smartphones.

When the iPhone was initially released, it did not even have support for Microsoft Exchange, much less any kind of enterprise management support. It wasn't until iPhone OS 2.0 that some semblance of an enterprise management feature-set was introduced. As iPhone OS/iOS has matured, Apple has continued to expand on enterprise management features, but they have taken a very interesting approach to it.

Rather than release Apple-written software solutions to manage iOS devices (like RIM's Blackberry Enterprise Server), Apple has chosen to open up certain APIs to developers, so that the third-party market can develop their own management solutions. Apple has done this while providing a small subset of these management features to consumers, for instance, Remote Wipe via MobileMe.

This has given rise to a number of mobile device management (MDM) vendors that are currently vying for dominance in this nascent niche of mobile device management.

With this background established, the first area to consider is the initial deployment of the iOS device.

3.1. Initial Deployment

As with all the next issues discussed, the way the initial deployment of the device is handled depends on whether or not the device is managed by a MDM solution or individually managed.

3.1.1. MDM Solution

A MDM solution would allow the iOS device to be initially provisioned in a couple different ways:

-End User Self-Service: In this scenario, the end user can do the initial setup over the air, without any IT interaction.

-Individual Device Setup: With this type of setup, IT would interactively provision each device, whether through a physical cable connection or over the air.

3.1.2. Individually Managed

In this management scenario, the end user would initially activate and sync the iOS device to their personal computer & iTunes account. They would then bring the iOS device into the workplace, and either:

-Manually connect to the business WiFi (if that is provided), manually setup their business email, and not have any policies applied to their device.

-Or, using the iPhone Configuration Utility provided by Apple, IT could connect to the device via USB, and push out a configuration profile that would automatically setup their business WiFi connection, business email, and enforce any device-specific organizational policies.

3.1.3 Security Implications & Considerations

Between the two individually managed options, from a security perspective, option two is better as it allows for the following:

-A pre-configured standard configuration profile, which means less manual entry, which equals less mistakes, which ultimately leads to less security issues related to misconfigurations.

-The enforcement of organizational policies.

3.2 OS Updates

Keeping iOS updated is an important part of managing the risks of the device. OS updates patch vulnerabilities in iOS, provide stability fixes, and add new features.

3.2.1. MDM & Individually Managed

Whether managed individually or through a MDM solution, the updating of iOS is the same: To update an iOS device, the user would connect their device to the computer & iTunes account that the device syncs with. If there is an update available, iTunes will prompt the user to download the update, backup the device, and install the update.

3.2.2. Security Implications & Considerations

Because an iOS device individually or MDM managed cannot be forced to upgrade the OS, the implication is that if Apple releases a critical patch for iOS, IT will not be able to push the patch to the iOS device and force an upgrade. The best alternative is that of the carrot and stick mentality:

Most MDM solutions have the ability to deny users access to organizational resources (email, calendar, and the like), if the iOS version is not version x or above. So the idea would be to alert users to upgrade iOS, and then x number of days later, disable organizational resources for those users who did not update their device.

Unfortunately, there is no such alternative for non-MDM managed devices, except during the initial provisioning, provided the iPhone Configuration Utility is used.

3.3. Disaster Recovery

Disaster Recovery is the process of recovering and restoring the organization back to normal operations after a disaster has occurred. Usually large organizations have a formalized Disaster Recovery plan as part of their Business Continuity Plan. This plan would include policies and procedures of what to do in an event of a catastrophic fire that destroys the data center. It would also include what to do when equipment has been stolen, or when data is destroyed.

We will focus on the following two disaster recovery scenarios: Theft and Data Loss.

3.3.1. MDM: Theft

For the MDM-managed device, IT can do any of the following from the MDM console:
-Locate the device with the built-in GPS (if applicable)
-Display a message or play a sound on the device
-Remotely wipe the device of all personal and corporate data

3.3.2. Individually Managed: Theft

For the individually managed device there are a couple different options to choose from in the event of an iOS device being stolen.

First, if the iOS device has been setup beforehand with Apple's free MobileMe "*Find my iPhone/iPod Touch/iPad"* app, the user can login to the app and do any number of things: locate the device with built-in GPS; Display a message or play a sound; Set a passcode remotely, and if needed, remotely wipe the device of all personal data.

Secondly, if the device has an ActiveSync account setup on it, as long as the ActiveSync administrator has the functionality setup and enabled, either the user or the ActiveSync administrator can remotely wipe the iOS device.

3.3.3. MDM & Individually Managed: Data Loss

If the device has suffered a catastrophic failure whereby all the data on the device has been lost or corrupted, the only way for a device to recover and restore the data is to connect it to the primarily-synced computer and iTunes, and to restore the last backup.

Unfortunately, unless the user syncs the device regularly, (in which case a backup will automatically be made), by default, no other backup will be made. It also needs to be mentioned that this method relies on the primarily-synced iTunes computer to store backups. This computer is more than likely not up to the same security and redundancy standards that a business fileserver would be.

For a fee, there are apps that can be installed that will backup certain types of data on the device. Unfortunately, the quality of the apps varies widely, and so a careful choice must be made when going down this road.

3.3.4. Security Implications & Considerations

One of the key things to keep in mind when considering a disaster recover plan for iOS devices is that the device backups reside on the computer where the iOS devices syncs, which might possibly the user's home computer. This has serious implications for sensitive data loss, especially when there is currently software on the open market that will recover encrypted keychains from a password-protected backup. ("Recover password-protected blackberry,")

3.4. End of Life

In a business environment, when a device is getting older, there is usually a standard end of life procedure to follow when the device is x number of years old. For example, part of the end of life process would be wiping the device of any sensitive data, and disposing of it through a yearly auction.

Depending on the device, Apple tends to release a new version yearly. Based on the past history of iOS, Apple does not seem to fully support more than two previous versions. Current industry conventional wisdom would expect the device to last for two to three years.

3.4.1. Business Owned

If the iOS device is owned by the organization, depending on the business-approved lifecycle timeline of the device, the business would replace the end of life device, and follow the end of life procedure for the specific device. This procedure would most likely include the following:

-Do factory reset on device
-Scrub device disk
-Verify that no sensitive data remains
-Dispose of device (auction, trash, etc.)

3.4.2. Personally Owned

Other the other hand, if the device is personally owned, more than likely the employee will keep it much longer than what business-owned equipment will be kept for. For example, it is industry standard best practice for a business to have a three to four year lifecycle for desktop computers. Conventional wisdom would suggest that the average age of the personally owned computer is much higher, potentially even five to six years.

Applying this mentality to personally owned iOS devices, and it is possible that the employee will use their iOS device beyond what the business would allow if it were owned by the business. This can lead to a number of end of life issues:

-If the version of iOS and apps installed on the device become end of life, security updates will not be provided anymore.

-Since the employee potentially will run the device into the ground, the risk of catastrophic device failure becomes much greater. With this in mind, there is no chance to put the device out of its misery before a disaster recovery scenario happens.

4. Risks and Controls: Specific Security Issues of Device and OS

This next section will detail specific security-related issues with the iOS device. Because these security issues are not specific to being MDM-managed or individually managed, there will be no need to break out each section.

4.1. Access Control

Access control is the ability to be able to control access to the device to authorized personnel only. iOS version 4 and above has the ability to set a simple four character pin or a complex password. A user will have to authenticate with one of these methods before access is granted to interact with the device, with the caveat being that the iPhone can be set to allow answering incoming calls without having to authenticate the user, among other similar settings.

For iPhones, there is an additional layer of access control provided by PUKs (PIN Unlock Key). A PUK is a passcode that can be set to authenticate a user before cellular phone and cellular data functions are used. ("iphone, ipad, ipod,")

Unfortunately, neither of these access control mechanisms are turned on by default.

MDM-managed devices, as well as individually managed devices that are configured with the iPhone Configuration Utility, can enforce a policy that turns on the initial passcode or force the use of complex passwords, though enforcing a PUK through this method is not currently possible.

4.1.1. Security Concerns & Implications

If a user has a four-character pin only, what about passcode guessing attacks? The way Apple mitigates this risk is with the following mechanisms:

-As the passcode is continually input incorrectly, the device will be temporarily disabled for longer and longer periods of time, which makes a passcode guessing attack much more infeasible.

-Allowing the option that the iOS device can be set to automatically securely wipe itself if the passcode has been entered incorrectly ten consecutive times.

In recent years, USA state supreme courts have ruled in opposite of each other when it comes to the legality of police officers being able to search mobile phones on arrested individuals. Most recently, the California Supreme Court ruled that this is legal. (The People v. Gregory Diaz, 2011) At this point in US law, coercing an individual to divulge a password is not a viable legal method. With this being the case, along with the fact that iOS data encryption is tied to the user's password, there is a legitimate case that can be made that an organization should always have their user's devices password-protected, hence, encrypted. (Radia, n.d.)

Finally, it would be remiss not to remind that this is just another layer in a defense in depth strategy; access control on iOS devices have had some major flaws in the past, and cannot be relied upon as the only layer of defense that an organization is hoping will keep their data safe.[1]

4.2. Buffer Overflow

A buffer overflow is a type of coding error that can be exploited to run arbitrary code. This arbitrary code can potentially be used to gain unauthorized access to a system. All modern day operating systems are susceptible to this type of error, though many have incorporated a number of controls to mitigate the risk of this type of error being exploited.

[1] For example: http://www.wired.com/gadgetlab/2008/08/massive-iphone/

As the underlying programming language that iOS is written in, C has a couple of well-known functions that are easily abused in this fashion: strcat and strcpy. Unfortunately, these functions are sometimes still used in iOS apps to this day. For developers, the best way to avoid this type of vulnerability is to not engage in manual memory management, but instead to use Cocoa objects such as NSString for string manipulation. (Dwivedi, Clark, & Thiel, 2010) According to Apple documentation, if C-style string manipulation is needed, the developer should use the strl family of functions instead. (Apple, 2010)

Beyond recommending how to avoid buffer overflow vulnerabilities, Apple has set up their iOS product line so that the last couple revisions have included support for the NX bit.[2]

The NX (No eXecute) bit is technology used by processors that allow the OS to dub certain sections of memory as non-executable, which allows for greater protection against arbitrary code being run because of a buffer overflow. With this extra hardware feature in place for the device, iOS is able to use it to enforce a tighter security posture, specifically when it comes to buffer overflows. (Dwivedi, Clark, & Thiel, 2010)

4.3. Web Application Attacks on iOS Devices

Because the devices that iOS runs on have much smaller screens and much more limited system resources than their laptop and desktop counterparts, when a user is browsing the Web, they will more than likely be browsing “mobile” versions of the website. For instance, Google’s mobile website is m.google.com versus google.com. Mobile HTML websites usually do not support all the bells and whistles that their big brothers do. For example, Mobile HTML websites usually have very limited support for cookies. Between the smaller screen size, mobile browsers, and the mobile version of the websites, this brings a new element of risk to web browsing on an iOS device that must be addressed.

[2] Though Apple’s documentation refers to it as the NX bit, since the processor in the most recent iOS devices is a Cortex-A8 or A9, which are processors implementing ARM v7, it should technically be referenced as XN (eXecute Never). See http://www.arm.com/miscPDFs/14128.pdf for more details

4.3.1. XSS & CSRF

Even though limited storage space on iOS devices may keep cookie storage to a minimum, Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CRSF) really only target a current session cookie. Coupled with the fact that most browsers on iOS support CSS, JavaScript and some limited form of xHTML, and it is obvious that these attack classes are still viable on devices that run iOS. (Dwivedi, Clark, & Thiel, 2010)

One interesting element to take note of is the possibility that users might be more susceptible to these types of attacks on an iOS device than a laptop or desktop because it is a bit more work to logout of a session on a mobile device than a desktop. It is more likely that a user will surf seamlessly from their banking website to their social media website, rather than having to laboriously scroll back up the page far enough to click the logout button on the bank website. (Dwivedi, Clark, & Thiel, 2010)

4.3.2. HTTP Redirects & Phishing

Though HTTP Redirect attacks are not as common as XSS or CRSF, the use of mobile browsers and smaller screens makes it worth discussing. This type of attack redirects the victim's browser to an arbitrary page chosen by the attacker, without the user's knowledge or consent. Though it may be vulnerability in the web application that allows the redirect to take place, done right, it would be very difficult for an iOS user to detect. This is because most iOS browsers, because of screen size limitations, do not keep the full URL of the current page in view all the time, unless the user specifically queries for it. This is not so much of an issue with some iOS devices with larger screens. (the iPad for instance) (Dwivedi, Clark, & Thiel, 2010)

Related to HTTP Redirect attacks, Phishing attacks are difficult to mitigate for the same reason as HTTP Redirect attacks: predominately small screen sizes for most iOS devices make it likely that the full URL will not be shown by default. The situation gets even more difficult when it comes to clicking on links in an email message; on their desktop, users are told to hover over the link and verify that it is a legitimate link. For iOS devices, users need to be told how to verify links in an email: Press down and hold it for three seconds until a dialog box pops up that shows what the URL is actually. This works well for short URLs, but long URLs get truncated, and it is possible that an illegitimate URL could be passed off as legitimate because of this. The only other option would be to "Copy" the URL, paste it into a new email, and verify it that way, since the whole URL is able to be seen. Either way, verifying that links in an email are legitimate is much more cumbersome on an iOS device. (Dwivedi, Clark, & Thiel, 2010)

4.3.3. SSL

With the advent of the popularity of session-hijacking tools like Firesheep[3], it has become vogue to add SSL to your web property. 2010 saw a SSL-only version of Gmail, and in early 2011, Facebook started rolling out SSL to its users. The problem is that it is still quite popular to not have SSL-enabled mobile forms, as the computational overhead of the key exchange is too burdensome for underpowered mobile devices. This is becoming less and less of an excuse as each new mobile device continues to become more powerful as each month rolls by.

The implication of using a non-SSL mobile form is that users think that since the non-mobile version of the website is encrypted, then the mobile version is also. (Dwivedi, Clark, & Thiel, 2010)

Himanshu Dwivdei, (2010) speaks to the practical risk inherent in non-SSL mobile forms:
"One could argue that because the ability to sniff on GSM or CDMA network is not as easy, clear-text transmission of credentials is not so big an issue; however, at some point, the communication medium will change GSM/CDMA to Ethernet, usually after the WAP gateway/proxy, thus allowing attackers on the other side of the fence to capture the clear-text credentials. Although the exploit scenario is more difficult, the idea of a username and password (which provide the ability to move money from one entity to another) passing through the network in clear text is less than an ideal situation. Probably the easiest way to check to see if a mobile form is ssl-enabled is checking for https in the URL." (p. 273)

4.3.4. Lack of HTTPOnly Flag Support & Secure Flag support

Some of the traditional methods of mitigating cross-site scripting, such as the HTTPOnly flag, may not be possible and/or supported on mobile browsers. Likewise, if mobile browsers do

[3] See http://codebutler.com/firesheep for details on FireSheep and its ilk

not support the SECURE flag, users could be unknowingly browsing insecurely. (Dwivedi, Clark, & Thiel, 2010)

4.3.5. Handling the Browser Cache

Himanshu Dwivdei , (2010) discusses why handling the browser cache differently on a mobile device is something that needs special consideration, especially for web application developers:

"Most mobile web applications are implemented without any specific client-side components. The risk of data being exposed on a lost phone is low and dependent on the behavior of the web browser on the user's phone. Most web browsers will not cache pages served using HTTPS, which further reduces the risk. Unfortunately, as a performance optimization, some mobile browsers ignore Cache header directives and will cache all pages. If this is the case, make sure our mobile web application has the appropriate mitigation. Options include removing the cache from the phone often and disabling the "back" button feature. Both of these cache-clearing solutions are best efforts and only work on some browsers." (p. 274)

4.3.6. Secure Data Storage

Apple touts the fact that the iPad (Generations 1 and 2) *" ...offers hardware-based encryption. iPad hardware encryption uses AES 256-bit encoding to protect all data on the device. Encryption is always enabled, and cannot be disabled by users."* (Apple, 2010)

But what does that actually mean?

Secure Data Storage was introduced in iOS4, and is the idea of making sure that user data cannot be accessed by the wrong people, and yet is still accessible to the right people. While Apple does not publicly release alot of technical information of how it works, the following is some pertinent information gleaned from what is publicly available.

SDS is a hardware-level encryption scheme that Apple has implemented starting with iOS4. It is geared primarily to protect user data, such as the user credentials that a Facebook app stores on the device, so as not to have to bother the user to login every time they open the app.

SDS is critically important for organizations that have custom-developed apps running on their employee iOS devices that store sensitive organizational information.

SDS is only available for the iPhone 3GS, iPod Touch 3rd generation or later, and the first generation iPad and later. All previous generation iOS devices do not support SDS. As a side note, 3GS users running iOS3, and who wish to take advantage of SDS when they upgrade to iOS4, must do a filesystem-level restore when they upgrade, as SDS does not support the iOS3 filesystem.

The strength of the encryption key is dependent on the passcode/password chosen by the user, which means that enforcing complex passwords should be a default policy set by most organizations.

The way SDS is setup leaves a few key decisions up to the app developers. Practically, this means that it is up to the developers to make security-conscious decisions when implementing SDS for their app. (Vance, 2010)

4.4 Mobile Security Software

4.4.1 Anti-Malware

As can be seen by the last couple years of Pwn2Own contests, as well as the multitude of iOS jailbreaking methods, iOS has had it's fair share of 0 day vulnerabilities. Some of the exploitations of these vulnerabilities are instigated just by visiting a certain webpage using Safari.

If exploiting a vulnerability and exfiltrating the sms database on an iPhone is possible through a 0 day[4], then it would not be a leap of logic to see the possibility of the downloading a malicious piece of software that would do more than just exfiltrate the sms database: how about recording calls and transferring the recording to a server overseas?

[4]Details here: http://zdnet.com/blog/security/pwn2own-2010-iphone-hacked-sms-database-hijacked/5836

Another large vector of potential iOS malware is through the Apple App store, though this vector seems to be much more of an issue with the wide-open Android Market, compared to the human-curated Apple App store.

Whatever the vector, the question remains the same: Is there any kind of anti-malware software that would be able to help mitigate the risk of malware on an iOS device?

Unfortunately, the lack of true multitasking on iOS4 makes a traditional anti-malware product impossible. There are some products that purport to be able to scan iOS devices for known malware, though the iOS device has to be directly connected to your computer for it to work. (P, 2010)

One of the most practical ways to mitigate the risk of malware on iOS devices is to not jailbreak. Because jailbreaking an iOS device takes away some of the built-in protections of a non-jailbroken device (Disallowing of unsigned code to run, sandboxing of all apps, etc.), users that have jailbroken devices are at a much higher risk of their device being infected by some type of malware.

The other practical way to protect against malware is to always keep iOS updated to the latest software revision, as most updates contain some type of security patch.

4.4.2. Firewall

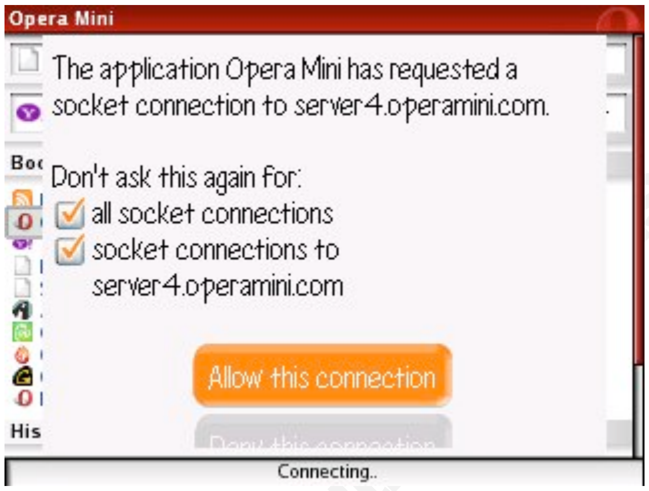
As with the previous security software (Anti-Malware), because iOS 4 lacks true multi-tasking, a traditional, non-Apple developed firewall is not possible.

However, with a jailbroken device, Firewall iP[5] is a well-known firewall that supports all current iOS devices.

[5]http://yllier.webs.com/firewall.html

Having a user-customizable firewall would be a boon for security on iOS devices, and it is something that some of Apple's competitors have had for quite a long time now.

Blackberry Firewall


Opera Mini

The application Opera Mini has requested a socket connection to server4.operamini.com.

Don't ask this again for:
- [✓] all socket connections
- [✓] socket connections to server4.operamini.com

4.4.3. Data Loss Prevention

As iOS devices continue to become more ubiquitous, especially with C-level executives, more organizations are wanting to find some sort of a DLP solution for iOS devices.

At this time, there is no known DLP product available for any iOS device, though there has been alot of interest this type of a product.

5. Risks and Controls: Application Management & Security

We now turn our attention to arguably the most important feature of iOS devices: the ability to install applications from Apple's App Store. This feature is the crux of an iOS device, as there are currently over three hundred and fifty thousand apps that can be downloaded and installed for an iOS device. Some of these apps extend the functionality of an iOS device beyond what Apple had ever dreamed of: from apps that help those that are colorblind

Joshua Brower, Josh@DefensiveDepth.com

differentiate between colors[6], to apps that help people with autism to communicate more effectively.[7]

As always, with increased functionality, comes increased risk.

In late 2010, ViaForensics released a report detailing severe vulnerabilities found in six major financial institution's iOS apps. These vulnerabilities could allow an attacker to gain access to the user's personal financial information. One of the most egregious issues was that of one app failing to encrypt passwords when they were transmitted as part of the user authentication process. (Schwartz, 2010)

Another pertinent example is that of the revelation by the WSJ of just how much personally identifiable information was/is being exfiltrated by popular apps. From the WSJ: *"An examination of 101 popular smartphone "apps"—games and other software applications for iPhone and Android phones—showed that 56 transmitted the phone's unique device ID to other companies without users' awareness or consent. Forty-seven apps transmitted the phone's location in some way. Five sent age, gender and other personal details to outsiders."* (Thurm, & Kane, 2010)

As an organization, how do you manage these risks?

5.1. Policy Generation & Enforcement

One of the first ways to manage these types of risks is that of developing a cohesive organizational policy that defines acceptable use of an iOS device that is used for business, and even more specifically, what type of apps are allowed to be installed on a device that is used for business.

[6] See here for a list of apps for those who are color blind: http://www.colblindor.com/2010/12/13/20-iphone-apps-for-the-color-blind/
[7] See here for a list of apps for those with autism: http://www.autismspeaks.org/community/resources/apps.php

Unfortunately, at this point in time, the APIs that Apple has released for the management of this aspect of iOS are very anemic, which means that the policy that needs to be generated will out of necessity, have to be very simple.

The following are the options available:

-Deny installing of apps from the official Apple App Store

-You can still push out your own in-house app to your employees with this option turned on

-Allow install of apps from the official Apple App store

-Set allowed install of apps up to the specified Content Rating

-Allow or Deny in-app purchases

-Allow or Deny the use of the iTunes Music Store

-Set allowed downloading and viewing of media up to the specified Content Rating

-Allow or Deny Explicit music and podcasts

-Allow or Deny the use of the YouTube app

-Allow or Deny multi-player gaming

-Allow or Deny adding Game Center friends

-Allow or Deny the use of Safari

Taking the above options into consideration from a technical perspective, it is clear that it will be very difficult to enforce some type of middle ground with employees—a middle ground, for example, which would allow some apps, and blacklist others. When it comes to regulating this aspect of iOS, it will have to be almost all or nothing.

This leaves the option that has already been discussed: that of the carrot and the stick.

Most MDM solutions will allow the viewing of an inventory of what apps are installed on an iOS device. With this in mind, one way to take a more balanced approach would be to compile a list of blacklisted apps that IT does not want on the iOS device, and to clearly communicate this policy to the iOS users. If a blacklisted app is installed, IT would get an alert, and IT would then cut the user off from organizational resources until the app has been removed from the device. This type of scenario does create more managerial overhead, though it allows more freedom for the iOS users.

Just to be clear, these options are available for iOS devices managed by a MDM solution, which allows the policies to be enforced at the time of initial configuration, as well as allowing the policies to be changed at will, and pushed out to managed devices over the air.

For iOS devices managed individually, if the iPhone Configuration Utility is used, these policies can be applied at the time of initial configuration, but to update a policy, the device will have to be physically connected to a computer that is running the iPhone Configuration Utility.

5.2. Initial Install

The initial install of an iOS app will happen in one of two ways:

1) Normal install via the Apple App Store; the most common way for organizations to deal with the needed iTunes account for the app install is to allow the user to use their personal iTunes account.

If there is a need to use the company account (for example, to download a company-purchased app for the user), then it is a “simple” login as the organizational iTunes account, download the app, and logout and login for the personal iTunes account again. Unfortunately, at this point, there is not a way to do this any easier.

2) In-house apps can be distributed with a MDM solution, usually with a MDM-specific App store that is installed on the iOS device when it is initially provisioned.

Option two is obviously not available for individually managed devices.

What about requiring a baseline of certain types of apps be installed on every iOS device? Unfortunately, there is not an easy way to require this. MDM solutions have the ability to "recommend" certain apps for users to install, but beyond the carrot and stick mentality, there is no other way to enforce the install of a baseline of apps, with one caveat: there is the ability to push out Web Clips, which would allow an organization to enforce certain links to web content (or an organizational web application) on every organizationally managed iOS device.

5.3. Application Updating

Once again, there are only two methods of updating an iOS app:

1) Normal updates via the Apple App Store.

2) In-house apps can be updated with a MDM solution, usually with a MDM-specific App store that is installed on the iOS device when it is initially provisioned.

Option two is obviously not available for individually managed devices.

5.3.1. Security Implications & Considerations

Consider the following (likely) scenario: A critical vulnerability is found in a popular app, and the app has been updated in the App Store. The problem is that neither individually managed nor MDM managed iOS devices can be forced to update the app, so an organization will be at risk until the user updates the app. Once again, the only viable solution would be the carrot and stick approach.

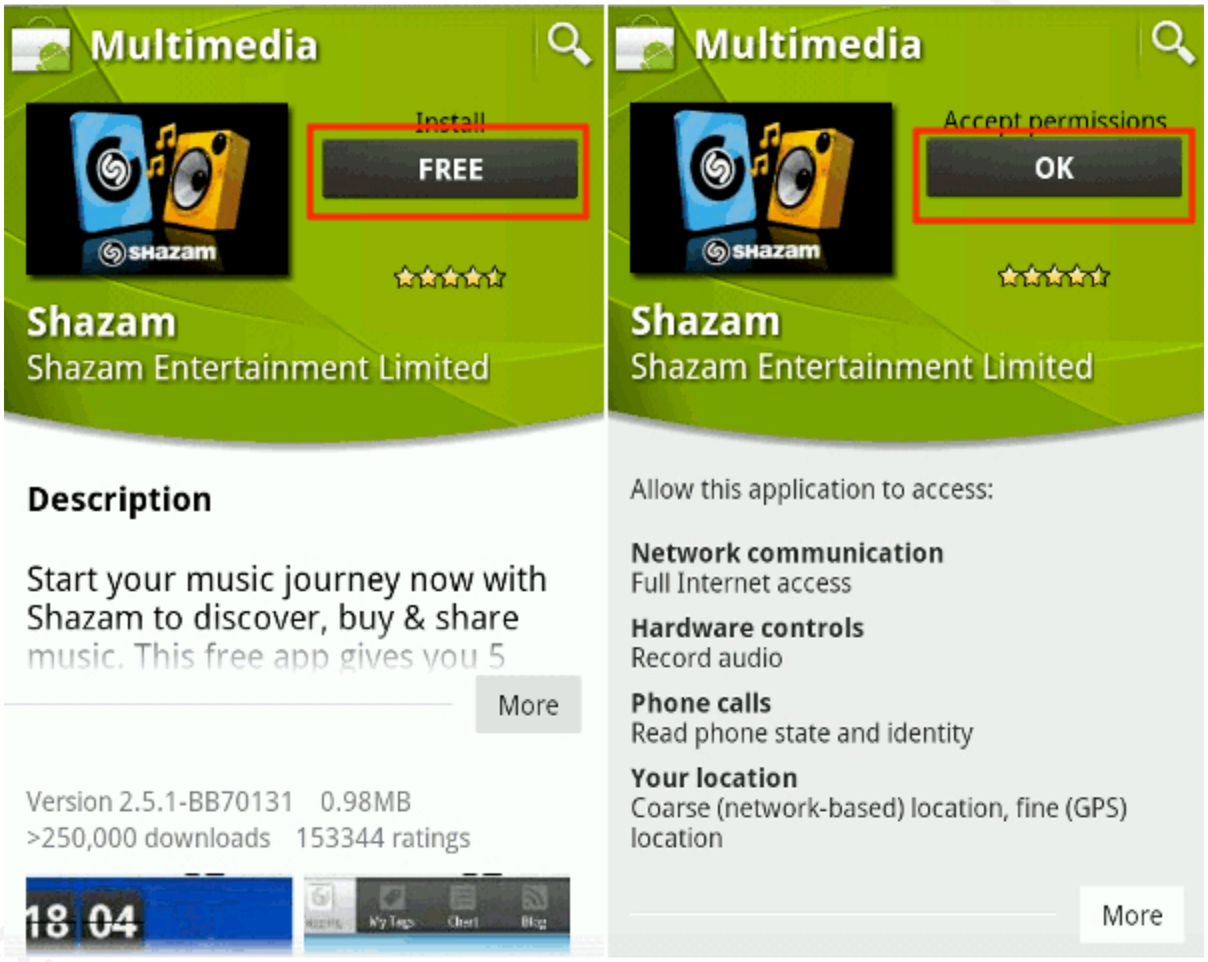
5.4. Application-Specific Security Issues

The focus will now be on some specific security issues related to apps.

5.4.1. Application Permissions

Through the official App store, when an iOS app is queued to be installed, if the user has not authenticated to iTunes in the previous fifteen minutes, the user is asked to authenticate. After authentication, and if no content restrictions apply, the app is downloaded and installed.

One major shortcoming of the current install process is that the user is not made aware what user data the app will have access to. Android-based devices, on the other hand, do:



Start your music journey now with Shazam to discover, buy & share music. This free app gives you 5

Version 2.5.1-BB70131 0.98MB
>250,000 downloads 153344 ratings

Allow this application to access:

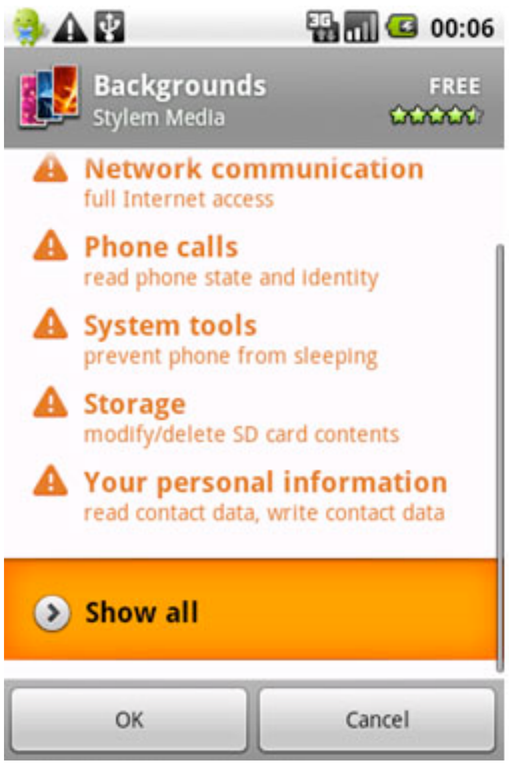
Network communication
Full Internet access

Hardware controls
Record audio

Phone calls
Read phone state and identity

Your location
Coarse (network-based) location, fine (GPS) location

Android also allows the user to view what currently installed apps have access to:



00:06

Backgrounds
Stylem Media
FREE

Network communication
full Internet access

Phone calls
read phone state and identity

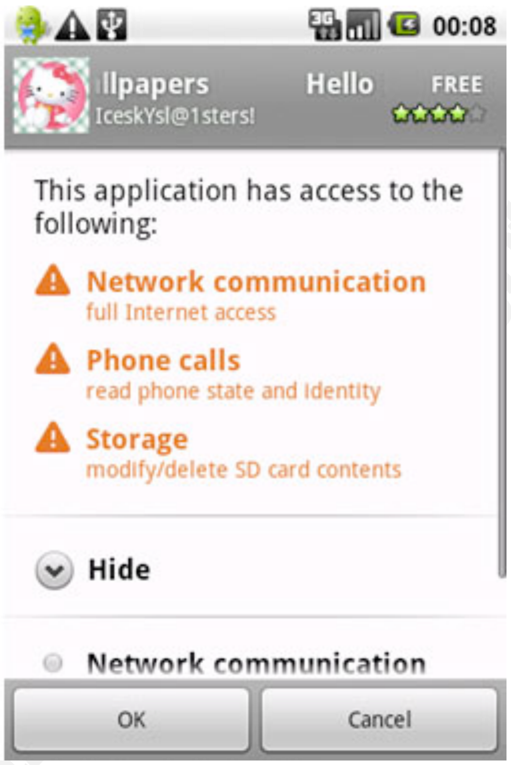
System tools
prevent phone from sleeping

Storage
modify/delete SD card contents

Your personal information
read contact data, write contact data

Show all

OK Cancel



00:08

llpapers Hello
IceskYsl@1sters!
FREE

This application has access to the following:

Network communication
full Internet access

Phone calls
read phone state and identity

Storage
modify/delete SD card contents

Hide

Network communication

OK Cancel

This type of permission system allows the user to decide if the app should really be installed. For instance, if an app that downloads new backgrounds every day wants to be able to record audio, this should send warning flags to the user that this app may not be legitimate.

Giving organizations the ability to make these types of informed decisions on what apps should be allowed on iOS devices that house sensitive organizational data is a much needed feature for iOS.

A practical example of this issue is that of the iOS app Dragon Dictation. With the release of an iOS version of Dragon Dictation, (a dictation app), it was discovered that the app was uploading the names of all contacts in the address book to its servers. This was a "feature," as it (according to Dragon Dictation) allowed easier recognition of names of people that might be dictated. There was enough of a public outcry that the developers of Dragon Dictation quickly added the option to allow or deny the uploading of contacts to its servers. If there was a better permissions system in place for iOS apps, (along the lines of the Android system), it would have been clearer from the beginning that Dragon Dictation might be doing something like this, as the app would have asked for permission to access the address book. (Michaels, 2009)

iOS does impose a certain (anemic) level of restriction to each app by default, and then in the course of operation, explicitly asks the user for additional permission for certain functions, for example, allowing the app to use the camera. Interestingly enough, by default, access to the Address Book and Photos are explicitly allowed, as is the ability to record audio. (Dwivedi, Clark, & Thiel, 2010)

5.4.2. InterProcess Communication & Isolation

iOS implements Mandatory Access Control (MAC) as its framework for enforcing app permissions. Each app is isolated by a permission system based on the TrustedBSD framework. (The actual name of the kernel sandboxing function is *Seatbelt*) This system works by enforcing policies that dictate what the app can and can't do. This allows iOS to partition apps from each other, and disallows malicious or compromised apps from modifying the kernel or from gaining access to other app's data. (Dwivedi, Clark, & Thiel, 2010)

As a side note, most jailbreaking routines disable the sandbox on an iOS device.

6. Risks and Controls: Use of iOS Device

We now turn our attention to the risks that are incurred with general use of an iOS device.

6.1. Multiple Users Scenario

Because iOS devices tend to be fairly expensive, organizations may consider asking multiple users to share one device.

Unfortunately, iOS devices are not designed to be used among multiple users; they are primarily a single user device. This can be seen predominately in the access control area: there is not a way to setup multiple profiles for unique users, as there is one passcode, and one user profile. An iOS device can only sync with one computer/instance of iTunes at a time, and from the iOS level, there is no concept of restricting use of an app to a specific user, though some apps do have the in-app ability to have multiple unique accounts.

With this being the case, organizations need to take this into account and be very careful of asking users to share iOS devices, especially if there is sensitive organizational data stored on the device.

If a user owns the iOS device, organizations must think through the possibility that it is very likely that the user will let their children play a game on the device, or allow a friend to use their cellphone for a call, potentially exposing sensitive organizational data to unauthorized personnel.

6.2. Policy Enforcement & Auditing

Another area that needs to be considered is that of enforcing corporate policies and auditing the adherence to those policies. For example, most organizations have an Acceptable Use Policy that lays out what kind of web content is allowed to be accessed over the network, or accessed with organizationally-owned hardware.

Depending on how the organization manages and filters web content, a client or some sort of configuration package may be pushed out to end users in order to access the web on-site. These configuration packages may not work for iOS devices-Will the organization have IT override the policy and allow iOS devices to bypass the content filter?

Along the same lines, some organizations use a client installed on the user's laptop to manage content even off-site. As of yet, the content management vendor may not have a client that will work with an iOS device.

If either of the above scenarios comes into play, then existing policies may need to be edited to take this lack of control and auditing into account.

6.3. Jailbreaking

Jailbreaking is the process of removing the iOS limitations imposed by Apple, specifically by gaining root access to the device. Jailbreaking allows the user to install apps from outside the jurisdiction of the official App Store, as well as modify the iOS theme and extensions. Jailbreaking is legal under USA law, as well as a number of other countries. (Anderson, 2010)

So what are the risks and concerns of jailbreaking an iOS device, especially when it comes to jailbroken devices being used for business? Unfortunately, they are numerous.

-Though legal in many countries, jailbreaking an iOS device voids its warranty

-Many security features that Apple has included in iOS are broken when an iOS device is jailbroken. This includes:

-The ability to run unsigned code is now available

-Installing apps that have not been reviewed and approved by Apple or the organization is now possible

-Jailbreaking removes the protections enforced by Seatbelt, the iOS sandboxing mechanism

-When a device is being jailbroken, essentially what is happening is that a vulnerability in iOS is being exploited to gain root access. An independent hacker or a hacker group usually writes the software that is used. Because they usually don't release the source code of the jailbreaking tool, how is it known that nothing else was done during the jailbreaking process? How can it ever be verified that there is no rootkit now running on the jailbroken iOS device? The same iOS device that accesses and possibly even houses sensitive organizational data…

As can be seen, the risks of allowing jail-broken iOS devices to access organizational data are overwhelming.

6.4. Secure Usage Awareness

There are many issues that crop up when an organization allows iOS devices to be used. End users need to be aware of these issues, and trained on how to reduce the risk of sensitive data loss and misuse of their device. The following are some key areas that users need to be trained in, when it comes to the usage of an iOS device.

6.4.1. Use a strong passcode, and don't share it

Users need to be reminded that they need to pick a strong passcode (not the last four digits of their social security number), and that they should not share it.

6.4.2. Don't allow unauthorized people to use your device

This is a difficult one, especially if the iOS device is personally owned. Users need to be reminded that their device may house sensitive organizational information, and they cannot just allow their children to play games on it.

6.4.3. At all times, be aware of where your device is and have a travel plan

Finally, users need to be reminded to always be aware of where their device is located, and how they will protect it from being lost or stolen when they travel. Nobody wants to be the next employee that accidentally leaves their prototype model at the bar![8]

7. Real-World Scenario

7.1. Introduction

Taking all the information from the previous sections, let us now see what it might look like in a real-world scenario.

Moten Hospital is a medium-size hospital with five hundred patient beds. Because they deal with sensitive patient health-care data, and are under many legal obligations, their overall risk appetite is very low, which drives a culture of tight security. The CIO has recently been

[8] More details here: http://gizmodo.com/#!5520438/how-apple-lost-the-next-iphone

approached for his input into the specifics of a planned iPad rollout to every doctor in the hospital. These iPads will be used for a corporate app that is being developed that will allow the doctor to access and edit patient records on their iPad.

After much research and planning, the CIO presents the following scenario for approval:

Each doctor will be allocated an iPad. The iPads will not be shared, as they are not designed for multiple users. The iPads will be provisioned with a MDM solution that will allow IT to fully manage the devices.

7.2. Lifecycle - Initial Provisioning

Each iPad will be initially provisioned by IT, and then given to the doctor. The following policy will be applied to each device when it is provisioned:

-iTunes, Safari, YouTube, and installing apps will be disabled.
-Mail will be setup with the doctor's Exchange account, so that they can send and receive their email and work with their calendar on the iPad.
-The hospital's patient records app will be installed.
-A simple four-character passcode will be enforced, and if it is entered ten times incorrectly, the iPad will be automatically wiped. The device will auto-lock after five minutes of inactivity.
-WiFi network settings will be installed and configured.

7.3. Lifecycle - OS & Application Updates

-When iOS updates are needed, IT will test the update, then perform rolling updates to the iPads as the doctors go off the clock.
-When needed, the hospital's patient information app installed on the iPads will be updated over the air through the MDM console

7.4. Lifecycle - End of life

Each iPad will be retired after three years in service. Each retired iPad will go through the following procedure:

-Factory Default Secure Wipe
-Verification of secure wipe
-Removal of asset tags
-Sold at next organizational auction

7.5. Lifecyle - Disaster Recovery

7.5.1. Theft

Each iPad is to be checked in to the specified storage space before the doctor leaves. If a device ever goes missing, IT will use the MDM solution to see if they can track it via GPS. If it cannot be immediately picked up or tracked, IT will set it to securely wipe itself next time it is able to be contacted. Because the patient data app doesn't actually store long-term information on the device itself, there is not much of a risk of sensitive data being lost through theft of the iPad, except for any sensitive data in the doctor’s Exchange account.

7.5.2. Catastrophic Data Loss

If an iPad suffers some type of catastrophic data loss, the device will be securely wiped and re-provisioned, if possible. Neither the patient data app nor the Exchange account store (unrecoverable) long-term information on the device itself, so there is a very low risk of actually losing any critical data if the device suffers a failure. The only inconvenience is to re-provision the replacement device.

7.6. Specific Security Issues

-The MDM solution will deny rouge iOS devices from gaining access to doctors Exchange accounts, thereby ensuring that only hospital-managed iOS devices house corporate data.
-The doctors will be trained on what the acceptable use of their iPad is. Fortunately, as the device is completely locked down, there will be less chance of any misuse.

Joshua Brower. Josh@DefensiveDepth.com

8. Conclusion

With the continued upward explosion of iOS devices in the mobile device portfolio of business users, comes a continued need for practical guidance on what the risks of using iOS devices are for organizations, and how to regulate and mitigate those risks.

This paper has considered what some of those risks are, and what are some practical ways to mitigate those risks. From the management of the iOS device itself, to the management of the apps on the device, we have seen what are the major pitfalls, and how to avoid them.

One of the key points that has been woven throughout this paper is that of the carrot and stick mentality. Because Apple has not provided a robust mechanism to truly manage what is allowed on the iOS device and what isn't, organizations are unfortunately stuck with three strategies:

-Allow everything
-Deny everything
-Allow some things, and play whack-a-mole with the rest

With this being the case, what strategy is chosen will depend on the organizational culture and the legal demands that the organization must adhere to.

The good news is that this niche is finally starting to see some real innovation, and in the coming iOS versions, it is expected that Apple will start dealing with these shortcomings.

In the meantime, organizations must clearly think through the risks that they are incurring by allowing their employees to use iOS devices for business, and be able to manage the risks in an organizationally appropriate manner.

Joshua Brower, Josh@DefensiveDepth.com

9. References

Anderson, N. (2010, March). *Apple loses big in drm ruling: jailbreaks are "fair use"*. Retrieved from http://arstechnica.com/tech-policy/news/2010/07/apple-loses-big-in-drm-ruling-jailbreaks-are-fair-use.ars

Coldewey, Devin. (2011, March 2). *Apple announces 100 million iphones, 15 million ipads sold*. Retrieved from http://www.crunchgear.com/2011/03/02/apple-announces-100-million-iphones-15-million-ipads-sold/

Dwivedi, H, Clark, C, & Thiel, D. (2010). *Mobile application security*. USA: McGraw-Hill Osborne Media.

ipad security overview. (2010). Retrieved from images.apple.com/ipad/business/pdf/iPad_Security_Overview.pdf

Michaels, P. (2009, December 16). *Dragon dictation adds privacy setting*. Retrieved from http://www.pcworld.com/article/184875/dragon_dictation_adds_privacy_setting.html

P, R.P. (2010, April 10). *Virusbarrier x6 – first anti-virus for ipad*. Retrieved from http://www.intego.com/news/intego-updates-virusbarrier-x6-to-scan-the-apple-ipad.asp

Radia, Ryan. (n.d.). Why you should always encrypt your smartphone. Retrieved from http://arstechnica.com/gadgets/guides/2011/01/why-you-should-always-encrypt-your-smartphone.ars

Recover password-protected blackberry and apple backups. (n.d.). Retrieved from http://www.elcomsoft.com/eppb.html

Schwartz, M.J. (2010, November 5). inshare permalink rss vulnerabilities found in banking apps. Retrieved from http://www.informationweek.com/news/security/vulnerabilities/showArticle.jhtml?articleID=228200291

Secure coding guide: avoiding buffer overflows. (2010, February 12). Retrieved from http://developer.apple.com/library/mac/#documentation/Security/Conceptual/SecureCodingGuide/Articles/BufferOverflows.html

The People v. Gregory Diaz, Super. Ct. No. 2007015733 (Ventura Country, CA 2011).

Thurm, S., & Kane, Y.I. (2010, December 17). *Your apps are watching you*. Retrieved from http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html

Vance, A. (2010, June 24). *Limitations of data protection in ios 4*. Retrieved from http://anthonyvance.com/blog/forensics/ios4_data_protection

vl, w. (2008, August 25). *Blackberry enterprise server release history*. Retrieved from http://wiki.itrezzo.com/Default.aspx?Page=BlackBerry_Enterprise_Server_History